

# **Standard for Information Technology— Portable Operating System Interface (POSIX®)**

## **Base Specifications, Issue 7**

Prepared by the Austin Group ([www.opengroup.org/austin](http://www.opengroup.org/austin)).

IMPORTANT NOTICE: This standard is not intended to assure safety, security, health, or environmental protection in all circumstances. Implementors of the standard are responsible for determining appropriate safety, security, environmental, and health practices or regulatory requirements.

This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading “Important Notice” or “Important Notices and Disclaimers Concerning IEEE Documents”. They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.



# Contents

<b>Volume</b>	<b>1</b>	<b>Base Definitions, Issue 7.....</b>	<b>1</b>
<b>Chapter</b>	<b>1</b>	<b>Introduction.....</b>	<b>3</b>
	1.1	Scope .....	3
	1.2	Conformance.....	4
	1.3	Normative References .....	4
	1.4	Change History .....	5
	1.5	Terminology .....	5
	1.6	Definitions and Concepts.....	6
	1.7	Portability .....	6
	1.7.1	Codes .....	7
	1.7.2	Margin Code Notation .....	13
<b>Chapter</b>	<b>2</b>	<b>Conformance.....</b>	<b>15</b>
	2.1	Implementation Conformance .....	15
	2.1.1	Requirements .....	15
	2.1.2	Documentation .....	16
	2.1.3	POSIX Conformance.....	17
	2.1.4	XSI Conformance .....	19
	2.1.5	Option Groups.....	20
	2.1.6	Options .....	26
	2.2	Application Conformance.....	29
	2.2.1	Strictly Conforming POSIX Application.....	29
	2.2.2	Conforming POSIX Application .....	30
	2.2.3	Conforming POSIX Application Using Extensions.....	30
	2.2.4	Strictly Conforming XSI Application .....	30
	2.2.5	Conforming XSI Application Using Extensions .....	31
	2.3	Language-Dependent Services for the C Programming Language .....	31
	2.4	Other Language-Related Specifications.....	31
<b>Chapter</b>	<b>3</b>	<b>Definitions.....</b>	<b>33</b>
	3.1	Abortive Release.....	33
	3.2	Absolute Pathname.....	33
	3.3	Access Mode .....	33
	3.4	Additional File Access Control Mechanism.....	33
	3.5	Address Space.....	33
	3.6	Advisory Information.....	34
	3.7	Affirmative Response .....	34
	3.8	Alert .....	34
	3.9	Alert Character (<alert>).....	34
	3.10	Alias Name.....	34
	3.11	Alignment .....	35
	3.12	Alternate File Access Control Mechanism .....	35

3.13	Alternate Signal Stack.....	35
3.14	Ancillary Data.....	35
3.15	Angle Brackets.....	35
3.16	Apostrophe Character (<apostrophe>).....	35
3.17	Application.....	35
3.18	Application Address.....	36
3.19	Application Program Interface (API) .....	36
3.20	Appropriate Privileges .....	36
3.21	Argument .....	36
3.22	Arm (a Timer) .....	36
3.23	Asterisk Character (<asterisk>) .....	36
3.24	Async-Cancel-Safe Function.....	37
3.25	Asynchronous Events.....	37
3.26	Asynchronous Input and Output .....	37
3.27	Async-Signal-Safe Function.....	37
3.28	Asynchronously-Generated Signal.....	37
3.29	Asynchronous I/O Completion.....	37
3.30	Asynchronous I/O Operation .....	37
3.31	Authentication .....	38
3.32	Authorization .....	38
3.33	Background Job .....	38
3.34	Background Process.....	38
3.35	Background Process Group (or Background Job).....	38
3.36	Backquote Character.....	38
3.37	Backslash Character (<backslash>) .....	38
3.38	Backspace Character (<backspace>) .....	39
3.39	Barrier .....	39
3.40	Basename.....	39
3.41	Basic Regular Expression (BRE).....	39
3.42	Batch Access List .....	39
3.43	Batch Administrator .....	39
3.44	Batch Client.....	40
3.45	Batch Destination .....	40
3.46	Batch Destination Identifier.....	40
3.47	Batch Directive.....	40
3.48	Batch Job .....	40
3.49	Batch Job Attribute.....	41
3.50	Batch Job Identifier.....	41
3.51	Batch Job Name .....	41
3.52	Batch Job Owner.....	41
3.53	Batch Job Priority .....	41
3.54	Batch Job State .....	41
3.55	Batch Name Service .....	41
3.56	Batch Name Space.....	42
3.57	Batch Node.....	42
3.58	Batch Operator.....	42
3.59	Batch Queue.....	42
3.60	Batch Queue Attribute.....	42
3.61	Batch Queue Position .....	42
3.62	Batch Queue Priority .....	43
3.63	Batch Rerunability.....	43
3.64	Batch Restart .....	43



3.65	Batch Server .....	43
3.66	Batch Server Name.....	43
3.67	Batch Service .....	43
3.68	Batch Service Request.....	43
3.69	Batch Submission .....	44
3.70	Batch System .....	44
3.71	Batch Target User .....	44
3.72	Batch User .....	44
3.73	Bind .....	44
3.74	Blank Character (<blank>).....	44
3.75	Blank Line.....	44
3.76	Blocked Process (or Thread) .....	45
3.77	Blocking .....	45
3.78	Block-Mode Terminal .....	45
3.79	Block Special File.....	45
3.80	Braces .....	45
3.81	Brackets.....	46
3.82	Broadcast .....	46
3.83	Built-In Utility (or Built-In).....	46
3.84	Byte.....	46
3.85	Byte Input/Output Functions .....	46
3.86	Carriage-Return Character (<carriage-return>) .....	46
3.87	Character .....	47
3.88	Character Array .....	47
3.89	Character Class.....	47
3.90	Character Set .....	47
3.91	Character Special File .....	48
3.92	Character String.....	48
3.93	Child Process .....	48
3.94	Circumflex Character (<circumflex>).....	48
3.95	Clock .....	48
3.96	Clock Jump.....	48
3.97	Clock Tick.....	48
3.98	Coded Character Set .....	49
3.99	Codeset .....	49
3.100	Collating Element.....	49
3.101	Collation .....	49
3.102	Collation Sequence.....	49
3.103	Column Position.....	50
3.104	Command.....	50
3.105	Command Language Interpreter .....	50
3.106	Composite Graphic Symbol.....	50
3.107	Condition Variable .....	50
3.108	Connected Socket .....	51
3.109	Connection .....	51
3.110	Connection Mode.....	51
3.111	Connectionless Mode .....	51
3.112	Control Character.....	51
3.113	Control Operator .....	51
3.114	Controlling Process.....	51
3.115	Controlling Terminal .....	52
3.116	Conversion Descriptor .....	52

3.117	Core File.....	52
3.118	CPU Time (Execution Time) .....	52
3.119	CPU-Time Clock.....	52
3.120	CPU-Time Timer.....	52
3.121	Current Job.....	52
3.122	Current Working Directory.....	53
3.123	Cursor Position.....	53
3.124	Datagram.....	53
3.125	Data Segment.....	53
3.126	Deferred Batch Service .....	53
3.127	Device .....	53
3.128	Device ID .....	53
3.129	Directory .....	53
3.130	Directory Entry (or Link) .....	53
3.131	Directory Stream .....	54
3.132	Disarm (a Timer) .....	54
3.133	Display .....	54
3.134	Display Line.....	54
3.135	Dollar-Sign Character (<dollar-sign>) .....	54
3.136	Dot.....	54
3.137	Dot-Dot.....	55
3.138	Double-Quote Character.....	55
3.139	Downshifting.....	55
3.140	Driver .....	55
3.141	Effective Group ID .....	55
3.142	Effective User ID .....	55
3.143	Eight-Bit Transparency .....	55
3.144	Empty Directory.....	56
3.145	Empty Line.....	56
3.146	Empty String (or Null String).....	56
3.147	Empty Wide-Character String.....	56
3.148	Encoding Rule .....	56
3.149	Entire Regular Expression.....	56
3.150	Epoch .....	57
3.151	Equivalence Class.....	57
3.152	Era.....	57
3.153	Event Management.....	57
3.154	Executable File.....	57
3.155	Execute.....	58
3.156	Execution Time .....	58
3.157	Execution Time Monitoring.....	58
3.158	Expand.....	58
3.159	Extended Regular Expression (ERE) .....	58
3.160	Extended Security Controls.....	58
3.161	Feature Test Macro .....	59
3.162	Field.....	59
3.163	FIFO Special File (or FIFO) .....	59
3.164	File .....	59
3.165	File Description .....	59
3.166	File Descriptor .....	60
3.167	File Group Class .....	60
3.168	File Mode.....	60

3.169	File Mode Bits .....	60
3.170	Filename .....	60
3.171	Filename String.....	61
3.172	File Offset .....	61
3.173	File Other Class .....	61
3.174	File Owner Class .....	61
3.175	File Permission Bits.....	61
3.176	File Serial Number .....	61
3.177	File System .....	61
3.178	File Type .....	62
3.179	Filter .....	62
3.180	First Open (of a File) .....	62
3.181	Flow Control .....	62
3.182	Foreground Job.....	62
3.183	Foreground Process .....	62
3.184	Foreground Process Group (or Foreground Job).....	62
3.185	Foreground Process Group ID .....	62
3.186	Form-Feed Character (<form-feed>).....	63
3.187	Graphic Character.....	63
3.188	Group Database.....	63
3.189	Group ID.....	63
3.190	Group Name .....	63
3.191	Hard Limit.....	64
3.192	Hard Link .....	64
3.193	Home Directory .....	64
3.194	Host Byte Order.....	64
3.195	Incomplete Line.....	64
3.196	Inf.....	64
3.197	Instrumented Application .....	64
3.198	Interactive Shell .....	65
3.199	Internationalization .....	65
3.200	Interprocess Communication .....	65
3.201	Invoke .....	65
3.202	Job.....	65
3.203	Job Control .....	65
3.204	Job Control Job ID .....	66
3.205	Last Close (of a File).....	66
3.206	Line.....	66
3.207	Linger .....	66
3.208	Link .....	66
3.209	Link Count .....	66
3.210	Live Process.....	66
3.211	Local Customs .....	67
3.212	Local Interprocess Communication (Local IPC).....	67
3.213	Locale .....	67
3.214	Localization.....	67
3.215	Login .....	67
3.216	Login Name .....	67
3.217	Map .....	67
3.218	Marked Message .....	68
3.219	Matched .....	68
3.220	Memory Mapped Files .....	68

3.221	Memory Object .....	68
3.222	Memory-Resident .....	68
3.223	Message .....	69
3.224	Message Catalog .....	69
3.225	Message Catalog Descriptor .....	69
3.226	Message Queue .....	69
3.227	Mode .....	69
3.228	Monotonic Clock .....	69
3.229	Mount Point .....	70
3.230	Multi-Character Collating Element .....	70
3.231	Multi-Threaded Library .....	70
3.232	Multi-Threaded Process .....	70
3.233	Multi-Threaded Program .....	70
3.234	Mutex .....	70
3.235	Name .....	71
3.236	Named STREAM .....	71
3.237	NaN (Not a Number) .....	71
3.238	Native Language .....	71
3.239	Negative Response .....	71
3.240	Network .....	71
3.241	Network Address .....	71
3.242	Network Byte Order .....	72
3.243	Newline Character (<newline>) .....	72
3.244	Nice Value .....	72
3.245	Non-Blocking .....	72
3.246	Non-Spacing Characters .....	72
3.247	NUL .....	73
3.248	Null Byte .....	73
3.249	Null Pointer .....	73
3.250	Null String .....	73
3.251	Null Wide-Character Code .....	73
3.252	Number-Sign Character (<number-sign>) .....	73
3.253	Object File .....	73
3.254	Octet .....	73
3.255	Offset Maximum .....	74
3.256	Opaque Address .....	74
3.257	Open File .....	74
3.258	Open File Description .....	74
3.259	Operand .....	74
3.260	Operator .....	74
3.261	Option .....	75
3.262	Option-Argument .....	75
3.263	Orientation .....	75
3.264	Orphaned Process Group .....	75
3.265	Page .....	75
3.266	Page Size .....	75
3.267	Parameter .....	76
3.268	Parent Directory .....	76
3.269	Parent Process .....	76
3.270	Parent Process ID .....	76
3.271	Pathname .....	76
3.272	Pathname Component .....	77

3.273	Path Prefix .....	77
3.274	Pattern.....	77
3.275	Period Character (<period>) .....	77
3.276	Permissions .....	78
3.277	Persistence.....	78
3.278	Pipe.....	78
3.279	Polling.....	78
3.280	Portable Character Set .....	78
3.281	Portable Filename.....	79
3.282	Portable Filename Character Set.....	79
3.283	Positional Parameter.....	79
3.284	Preallocation .....	79
3.285	Preempted Process (or Thread).....	79
3.286	Previous Job .....	79
3.287	Printable Character .....	80
3.288	Printable File .....	80
3.289	Priority .....	80
3.290	Priority Band.....	80
3.291	Priority Inversion .....	80
3.292	Priority Scheduling.....	80
3.293	Priority-Based Scheduling .....	80
3.294	Privilege.....	81
3.295	Process .....	81
3.296	Process Group.....	81
3.297	Process Group ID .....	81
3.298	Process Group Leader.....	81
3.299	Process Group Lifetime .....	81
3.300	Process ID.....	82
3.301	Process Lifetime.....	82
3.302	Process Memory Locking.....	82
3.303	Process Termination.....	82
3.304	Process-To-Process Communication .....	82
3.305	Process Virtual Time .....	83
3.306	Program .....	83
3.307	Protocol.....	83
3.308	Pseudo-Terminal .....	83
3.309	Radix Character .....	83
3.310	Read-Only File System .....	83
3.311	Read-Write Lock.....	83
3.312	Real Group ID.....	84
3.313	Real Time .....	84
3.314	Realtime Signal Extension .....	84
3.315	Real User ID .....	84
3.316	Record .....	84
3.317	Redirection .....	84
3.318	Redirection Operator .....	85
3.319	Referenced Shared Memory Object.....	85
3.320	Refresh .....	85
3.321	Regular Expression .....	85
3.322	Region .....	85
3.323	Regular File .....	85
3.324	Relative Pathname .....	86

3.325	Relocatable File.....	86
3.326	Relocation.....	86
3.327	Requested Batch Service .....	86
3.328	(Time) Resolution .....	86
3.329	Robust Mutex.....	86
3.330	Root Directory .....	86
3.331	Runnable Process (or Thread) .....	86
3.332	Running Process (or Thread).....	87
3.333	Saved Resource Limits .....	87
3.334	Saved Set-Group-ID.....	87
3.335	Saved Set-User-ID .....	87
3.336	Scheduling.....	87
3.337	Scheduling Allocation Domain .....	87
3.338	Scheduling Contention Scope .....	87
3.339	Scheduling Policy.....	88
3.340	Screen .....	88
3.341	Scroll.....	88
3.342	Semaphore.....	88
3.343	Session .....	89
3.344	Session Leader .....	89
3.345	Session Lifetime.....	89
3.346	Shared Memory Object.....	89
3.347	Shell.....	89
3.348	Shell, the .....	89
3.349	Shell Script.....	90
3.350	Signal.....	90
3.351	Signal Stack .....	90
3.352	Single-Quote Character .....	90
3.353	Single-Threaded Process .....	90
3.354	Single-Threaded Program.....	90
3.355	Slash Character (<slash>).....	91
3.356	Socket .....	91
3.357	Socket Address .....	91
3.358	Soft Limit .....	91
3.359	Source Code .....	91
3.360	Space Character (<space>).....	92
3.361	Spawn .....	92
3.362	Special Built-In.....	92
3.363	Special Parameter.....	92
3.364	Spin Lock.....	92
3.365	Sporadic Server.....	92
3.366	Standard Error .....	92
3.367	Standard Input.....	92
3.368	Standard Output .....	93
3.369	Standard Utilities .....	93
3.370	Stream .....	93
3.371	STREAM .....	93
3.372	STREAM End.....	93
3.373	STREAM Head .....	93
3.374	STREAMS Multiplexor.....	94
3.375	String.....	94
3.376	Subshell.....	94

3.377	Successfully Transferred .....	94
3.378	Supplementary Group ID .....	94
3.379	Suspended Job .....	94
3.380	Symbolic Constant .....	95
3.381	Symbolic Link .....	95
3.382	Synchronized Input and Output.....	95
3.383	Synchronized I/O Completion .....	95
3.384	Synchronized I/O Data Integrity Completion.....	95
3.385	Synchronized I/O File Integrity Completion .....	96
3.386	Synchronized I/O Operation .....	96
3.387	Synchronous I/O Operation.....	96
3.388	Synchronously-Generated Signal .....	96
3.389	System.....	96
3.390	System Boot.....	96
3.391	System Clock.....	96
3.392	System Console .....	97
3.393	System Crash .....	97
3.394	System Databases.....	97
3.395	System Documentation .....	97
3.396	System Process.....	97
3.397	System Reboot .....	97
3.398	System Trace Event .....	97
3.399	System-Wide .....	98
3.400	Tab Character (<tab>).....	98
3.401	Terminal (or Terminal Device) .....	98
3.402	Text Column.....	98
3.403	Text File.....	98
3.404	Thread .....	99
3.405	Thread ID .....	99
3.406	Thread List .....	99
3.407	Thread-Safe .....	99
3.408	Thread-Specific Data Key.....	99
3.409	Tilde Character (<tilde>).....	99
3.410	Timeouts .....	100
3.411	Timer .....	100
3.412	Timer Overrun .....	100
3.413	Token.....	100
3.414	Trace Analyzer Process.....	100
3.415	Trace Controller Process.....	100
3.416	Trace Event.....	100
3.417	Trace Event Type .....	100
3.418	Trace Event Type Mapping .....	101
3.419	Trace Filter.....	101
3.420	Trace Generation Version .....	101
3.421	Trace Log .....	101
3.422	Trace Point.....	101
3.423	Trace Stream.....	101
3.424	Trace Stream Identifier .....	101
3.425	Trace System .....	101
3.426	Traced Process.....	102
3.427	Tracing Status of a Trace Stream .....	102
3.428	Typed Memory Name Space .....	102

3.429	Typed Memory Object .....	102
3.430	Typed Memory Pool .....	102
3.431	Typed Memory Port .....	102
3.432	Unbind .....	102
3.433	Unit Data .....	102
3.434	Upshifting .....	103
3.435	User Database .....	103
3.436	User ID .....	103
3.437	User Name .....	103
3.438	User Trace Event .....	103
3.439	Utility .....	104
3.440	Variable .....	104
3.441	Vertical-Tab Character (<vertical-tab>) .....	104
3.442	White Space .....	104
3.443	Wide-Character Code (C Language) .....	104
3.444	Wide-Character Input/Output Functions .....	105
3.445	Wide-Character String .....	105
3.446	Word .....	105
3.447	Working Directory (or Current Working Directory) .....	105
3.448	Worldwide Portability Interface .....	105
3.449	Write .....	105
3.450	XSI .....	105
3.451	XSI-Conformant .....	106
3.452	Zombie Process .....	106
3.453	±0 .....	106
<b>Chapter 4</b>	<b>General Concepts .....</b>	<b>107</b>
4.1	Concurrent Execution .....	107
4.2	Default Initialization .....	107
4.3	Directory Protection .....	108
4.4	Extended Security Controls .....	108
4.5	File Access Permissions .....	108
4.6	File Hierarchy .....	109
4.7	Filenames .....	109
4.8	Filename Portability .....	109
4.9	File Times Update .....	109
4.10	Host and Network Byte Orders .....	110
4.11	Measurement of Execution Time .....	110
4.12	Memory Synchronization .....	111
4.13	Pathname Resolution .....	111
4.14	Process ID Reuse .....	113
4.15	Scheduling Policy .....	113
4.16	Seconds Since the Epoch .....	113
4.17	Semaphore .....	114
4.18	Thread-Safety .....	114
4.19	Tracing .....	115
4.20	Treatment of Error Conditions for Mathematical Functions .....	117
4.20.1	Domain Error .....	117
4.20.2	Pole Error .....	117
4.20.3	Range Error .....	118
4.21	Treatment of NaN Arguments for the Mathematical	



		Functions .....	118
	4.22	Utility .....	119
	4.23	Variable Assignment.....	119
<b>Chapter</b>	<b>5</b>	<b>File Format Notation.....</b>	<b>121</b>
<b>Chapter</b>	<b>6</b>	<b>Character Set .....</b>	<b>125</b>
	6.1	Portable Character Set .....	125
	6.2	Character Encoding .....	128
	6.3	C Language Wide-Character Codes .....	129
	6.4	Character Set Description File.....	129
	6.4.1	State-Dependent Character Encodings .....	133
<b>Chapter</b>	<b>7</b>	<b>Locale .....</b>	<b>135</b>
	7.1	General.....	135
	7.2	POSIX Locale .....	136
	7.3	Locale Definition .....	136
	7.3.1	LC_CTYPE .....	139
	7.3.2	LC_COLLATE.....	147
	7.3.3	LC_MONETARY .....	155
	7.3.4	LC_NUMERIC.....	158
	7.3.5	LC_TIME .....	159
	7.3.6	LC_MESSAGES .....	165
	7.4	Locale Definition Grammar .....	166
	7.4.1	Locale Lexical Conventions.....	166
	7.4.2	Locale Grammar.....	167
<b>Chapter</b>	<b>8</b>	<b>Environment Variables.....</b>	<b>173</b>
	8.1	Environment Variable Definition.....	173
	8.2	Internationalization Variables .....	174
	8.3	Other Environment Variables.....	177
<b>Chapter</b>	<b>9</b>	<b>Regular Expressions.....</b>	<b>181</b>
	9.1	Regular Expression Definitions.....	181
	9.2	Regular Expression General Requirements.....	182
	9.3	Basic Regular Expressions .....	183
	9.3.1	BREs Matching a Single Character or Collating Element .....	183
	9.3.2	BRE Ordinary Characters.....	183
	9.3.3	BRE Special Characters .....	183
	9.3.4	Periods in BREs .....	184
	9.3.5	RE Bracket Expression.....	184
	9.3.6	BREs Matching Multiple Characters .....	186
	9.3.7	BRE Precedence .....	187
	9.3.8	BRE Expression Anchoring.....	188
	9.4	Extended Regular Expressions.....	188
	9.4.1	EREs Matching a Single Character or Collating Element .....	188
	9.4.2	ERE Ordinary Characters.....	188
	9.4.3	ERE Special Characters .....	189
	9.4.4	Periods in EREs .....	189
	9.4.5	ERE Bracket Expression .....	189

	9.4.6	EREs Matching Multiple Characters .....	190
	9.4.7	ERE Alternation.....	191
	9.4.8	ERE Precedence .....	191
	9.4.9	ERE Expression Anchoring.....	191
	9.5	Regular Expression Grammar .....	192
	9.5.1	BRE/ERE Grammar Lexical Conventions.....	192
	9.5.2	RE and Bracket Expression Grammar.....	193
	9.5.3	ERE Grammar .....	195
<b>Chapter</b>	<b>10</b>	<b>Directory Structure and Devices .....</b>	<b>197</b>
	10.1	Directory Structure and Files.....	197
	10.2	Output Devices and Terminal Types.....	198
<b>Chapter</b>	<b>11</b>	<b>General Terminal Interface .....</b>	<b>199</b>
	11.1	Interface Characteristics .....	199
	11.1.1	Opening a Terminal Device File.....	199
	11.1.2	Process Groups .....	200
	11.1.3	The Controlling Terminal.....	200
	11.1.4	Terminal Access Control .....	201
	11.1.5	Input Processing and Reading Data .....	201
	11.1.6	Canonical Mode Input Processing.....	202
	11.1.7	Non-Canonical Mode Input Processing.....	202
	11.1.8	Writing Data and Output Processing .....	203
	11.1.9	Special Characters .....	204
	11.1.10	Modem Disconnect .....	205
	11.1.11	Closing a Terminal Device File.....	205
	11.2	Parameters that Can be Set .....	205
	11.2.1	The termios Structure .....	205
	11.2.2	Input Modes.....	206
	11.2.3	Output Modes.....	207
	11.2.4	Control Modes .....	209
	11.2.5	Local Modes.....	210
	11.2.6	Special Control Characters.....	212
<b>Chapter</b>	<b>12</b>	<b>Utility Conventions .....</b>	<b>213</b>
	12.1	Utility Argument Syntax.....	213
	12.2	Utility Syntax Guidelines.....	216
<b>Chapter</b>	<b>13</b>	<b>Headers .....</b>	<b>219</b>
<b>Volume</b>	<b>2</b>	<b>System Interfaces, Issue 7.....</b>	<b>467</b>
<b>Chapter</b>	<b>1</b>	<b>Introduction.....</b>	<b>469</b>
	1.1	Relationship to Other Formal Standards.....	469
	1.2	Format of Entries.....	469
<b>Chapter</b>	<b>2</b>	<b>General Information .....</b>	<b>471</b>
	2.1	Use and Implementation of Interfaces .....	471
	2.1.1	Use and Implementation of Functions.....	471
	2.1.2	Use and Implementation of Macros .....	472
	2.2	The Compilation Environment .....	472

2.2.1	POSIX.1 Symbols.....	472
2.2.2	The Name Space.....	473
2.3	Error Numbers.....	481
2.3.1	Additional Error Numbers .....	488
2.4	Signal Concepts .....	488
2.4.1	Signal Generation and Delivery.....	488
2.4.2	Realtime Signal Generation and Delivery .....	489
2.4.3	Signal Actions .....	490
2.4.4	Signal Effects on Other Functions.....	495
2.5	Standard I/O Streams .....	495
2.5.1	Interaction of File Descriptors and Standard I/O Streams .....	497
2.5.2	Stream Orientation and Encoding Rules .....	498
2.6	STREAMS.....	500
2.6.1	Accessing STREAMS .....	501
2.7	XSI Interprocess Communication .....	501
2.7.1	IPC General Description .....	502
2.8	Realtime.....	503
2.8.1	Realtime Signals .....	503
2.8.2	Asynchronous I/O.....	503
2.8.3	Memory Management.....	505
2.8.4	Process Scheduling.....	506
2.8.5	Clocks and Timers.....	511
2.9	Threads .....	512
2.9.1	Thread-Safety.....	513
2.9.2	Thread IDs.....	513
2.9.3	Thread Mutexes.....	514
2.9.4	Thread Scheduling .....	515
2.9.5	Thread Cancellation.....	517
2.9.6	Thread Read-Write Locks.....	521
2.9.7	Thread Interactions with Regular File Operations.....	522
2.9.8	Use of Application-Managed Thread Stacks.....	522
2.9.9	Synchronization Object Copies and Alternative Mappings.....	523
2.10	Sockets .....	523
2.10.1	Address Families .....	523
2.10.2	Addressing .....	524
2.10.3	Protocols .....	524
2.10.4	Routing .....	524
2.10.5	Interfaces .....	524
2.10.6	Socket Types.....	524
2.10.7	Socket I/O Mode.....	525
2.10.8	Socket Owner.....	526
2.10.9	Socket Queue Limits.....	526
2.10.10	Pending Error .....	526
2.10.11	Socket Receive Queue.....	526
2.10.12	Socket Out-of-Band Data State.....	527
2.10.13	Connection Indication Queue .....	527
2.10.14	Signals.....	527
2.10.15	Asynchronous Errors.....	527
2.10.16	Use of Options .....	528
2.10.17	Use of Sockets for Local UNIX Connections .....	531

	2.10.18	Use of Sockets over Internet Protocols.....	532
	2.10.19	Use of Sockets over Internet Protocols Based on IPv4.....	532
	2.10.20	Use of Sockets over Internet Protocols Based on IPv6.....	532
	2.11	Tracing .....	536
	2.11.1	Tracing Data Definitions .....	538
	2.11.2	Trace Event Type Definitions.....	542
	2.11.3	Trace Functions.....	546
	2.12	Data Types.....	547
	2.12.1	Defined Types .....	547
	2.12.2	The char Type.....	548
	2.13	Status Information .....	548
	2.14	File Descriptor Allocation .....	549
<b>Chapter</b>	<b>3</b>	<b>System Interfaces.....</b>	<b>551</b>
<b>Volume</b>	<b>3</b>	<b>Shell and Utilities, Issue 7 .....</b>	<b>2315</b>
<b>Chapter</b>	<b>1</b>	<b>Introduction.....</b>	<b>2317</b>
	1.1	Relationship to Other Documents .....	2317
	1.1.1	System Interfaces.....	2317
	1.1.2	Concepts Derived from the ISO C Standard .....	2321
	1.2	Utility Limits.....	2323
	1.3	Grammar Conventions.....	2325
	1.4	Utility Description Defaults.....	2326
	1.5	Considerations for Utilities in Support of Files of Arbitrary Size.....	2333
	1.6	Built-In Utilities .....	2334
<b>Chapter</b>	<b>2</b>	<b>Shell Command Language .....</b>	<b>2335</b>
	2.1	Shell Introduction.....	2335
	2.2	Quoting.....	2336
	2.2.1	Escape Character (Backslash).....	2336
	2.2.2	Single-Quotes.....	2336
	2.2.3	Double-Quotes.....	2336
	2.3	Token Recognition.....	2337
	2.3.1	Alias Substitution.....	2338
	2.4	Reserved Words.....	2339
	2.5	Parameters and Variables.....	2339
	2.5.1	Positional Parameters .....	2339
	2.5.2	Special Parameters .....	2340
	2.5.3	Shell Variables.....	2341
	2.6	Word Expansions .....	2343
	2.6.1	Tilde Expansion .....	2344
	2.6.2	Parameter Expansion.....	2344
	2.6.3	Command Substitution .....	2347
	2.6.4	Arithmetic Expansion.....	2348
	2.6.5	Field Splitting .....	2349
	2.6.6	Pathname Expansion .....	2350
	2.6.7	Quote Removal.....	2350

2.7	Redirection .....	2350
2.7.1	Redirecting Input .....	2351
2.7.2	Redirecting Output .....	2351
2.7.3	Appending Redirected Output .....	2351
2.7.4	Here-Document .....	2352
2.7.5	Duplicating an Input File Descriptor .....	2353
2.7.6	Duplicating an Output File Descriptor .....	2353
2.7.7	Open File Descriptors for Reading and Writing .....	2353
2.8	Exit Status and Errors .....	2353
2.8.1	Consequences of Shell Errors .....	2353
2.8.2	Exit Status for Commands .....	2354
2.9	Shell Commands .....	2355
2.9.1	Simple Commands .....	2355
2.9.2	Pipelines .....	2358
2.9.3	Lists .....	2359
2.9.4	Compound Commands .....	2361
2.9.5	Function Definition Command .....	2364
2.10	Shell Grammar .....	2365
2.10.1	Shell Grammar Lexical Conventions .....	2365
2.10.2	Shell Grammar Rules .....	2365
2.11	Signals and Error Handling .....	2371
2.12	Shell Execution Environment .....	2371
2.13	Pattern Matching Notation .....	2372
2.13.1	Patterns Matching a Single Character .....	2372
2.13.2	Patterns Matching Multiple Characters .....	2373
2.13.3	Patterns Used for Filename Expansion .....	2373
2.14	Special Built-In Utilities .....	2374
<b>Chapter 3</b>	<b>Batch Environment Services .....</b>	<b>2417</b>
3.1	General Concepts .....	2417
3.1.1	Batch Client-Server Interaction .....	2417
3.1.2	Batch Queues .....	2418
3.1.3	Batch Job Creation .....	2418
3.1.4	Batch Job Tracking .....	2418
3.1.5	Batch Job Routing .....	2419
3.1.6	Batch Job Execution .....	2419
3.1.7	Batch Job Exit .....	2420
3.1.8	Batch Job Abort .....	2420
3.1.9	Batch Authorization .....	2420
3.1.10	Batch Administration .....	2420
3.1.11	Batch Notification .....	2421
3.2	Batch Services .....	2421
3.2.1	Batch Job States .....	2422
3.2.2	Deferred Batch Services .....	2423
3.2.3	Requested Batch Services .....	2432
3.3	Common Behavior for Batch Environment Utilities .....	2439
3.3.1	Batch Job Identifier .....	2439
3.3.2	Destination .....	2440
3.3.3	Multiple Keyword-Value Pairs .....	2441
<b>Chapter 4</b>	<b>Utilities .....</b>	<b>2443</b>

<b>Volume</b>	<b>4</b>	<b>Rationale (Informative), Issue 7.....</b>	<b>3463</b>
<b>Part</b>	<b>A</b>	<b>Base Definitions .....</b>	<b>3465</b>
<b>Appendix</b>	<b>A</b>	<b>Rationale for Base Definitions.....</b>	<b>3467</b>
	A.1	Introduction .....	3467
	A.1.1	Scope .....	3467
	A.1.2	Conformance.....	3470
	A.1.3	Normative References .....	3470
	A.1.4	Change History .....	3470
	A.1.5	Terminology .....	3470
	A.1.6	Definitions and Concepts.....	3472
	A.1.7	Portability .....	3472
	A.2	Conformance.....	3473
	A.2.1	Implementation Conformance .....	3473
	A.2.2	Application Conformance.....	3477
	A.2.3	Language-Dependent Services for the C Programming Language.....	3477
	A.2.4	Other Language-Related Specifications.....	3478
	A.3	Definitions .....	3478
	A.4	General Concepts .....	3500
	A.4.1	Concurrent Execution.....	3500
	A.4.2	Directory Protection.....	3501
	A.4.3	Extended Security Controls.....	3501
	A.4.4	File Access Permissions.....	3501
	A.4.5	File Hierarchy .....	3501
	A.4.6	Filenames.....	3502
	A.4.7	Filename Portability.....	3503
	A.4.8	File Times Update .....	3503
	A.4.9	Host and Network Byte Order.....	3504
	A.4.10	Measurement of Execution Time .....	3504
	A.4.11	Memory Synchronization .....	3504
	A.4.12	Pathname Resolution.....	3506
	A.4.13	Process ID Reuse .....	3507
	A.4.14	Scheduling Policy.....	3507
	A.4.15	Seconds Since the Epoch .....	3507
	A.4.16	Semaphore.....	3509
	A.4.17	Thread-Safety.....	3509
	A.4.18	Tracing .....	3509
	A.4.19	Treatment of Error Conditions for Mathematical Functions .....	3509
	A.4.20	Treatment of NaN Arguments for Mathematical Functions .....	3509
	A.4.21	Utility .....	3509
	A.4.22	Variable Assignment.....	3509
	A.5	File Format Notation .....	3510
	A.6	Character Set.....	3510
	A.6.1	Portable Character Set .....	3510
	A.6.2	Character Encoding .....	3511
	A.6.3	C Language Wide-Character Codes .....	3511
	A.6.4	Character Set Description File.....	3511

A.7	Locale .....	3514
A.7.1	General.....	3514
A.7.2	POSIX Locale .....	3514
A.7.3	Locale Definition .....	3514
A.7.4	Locale Definition Grammar.....	3521
A.7.5	Locale Definition Example.....	3522
A.8	Environment Variables .....	3525
A.8.1	Environment Variable Definition.....	3525
A.8.2	Internationalization Variables .....	3526
A.8.3	Other Environment Variables.....	3527
A.9	Regular Expressions .....	3528
A.9.1	Regular Expression Definitions.....	3529
A.9.2	Regular Expression General Requirements.....	3529
A.9.3	Basic Regular Expressions .....	3530
A.9.4	Extended Regular Expressions.....	3533
A.9.5	Regular Expression Grammar.....	3535
A.10	Directory Structure and Devices.....	3536
A.10.1	Directory Structure and Files.....	3536
A.10.2	Output Devices and Terminal Types.....	3536
A.11	General Terminal Interface .....	3536
A.11.1	Interface Characteristics .....	3537
A.11.2	Parameters that Can be Set .....	3541
A.12	Utility Conventions.....	3543
A.12.1	Utility Argument Syntax.....	3543
A.12.2	Utility Syntax Guidelines.....	3544
A.13	Headers.....	3546
A.13.1	Format of Entries.....	3546
A.13.2	Removed Headers in Issue 7 .....	3547
<b>Part</b>	<b>B</b>	<b>System Interfaces..... 3549</b>
<b>Appendix</b>	<b>B</b>	<b>Rationale for System Interfaces..... 3551</b>
B.1	Introduction .....	3551
B.1.1	Change History .....	3551
B.1.2	Relationship to Other Formal Standards.....	3554
B.1.3	Format of Entries.....	3554
B.2	General Information .....	3555
B.2.1	Use and Implementation of Interfaces.....	3555
B.2.2	The Compilation Environment .....	3556
B.2.3	Error Numbers.....	3561
B.2.4	Signal Concepts .....	3565
B.2.5	Standard I/O Streams .....	3575
B.2.6	STREAMS.....	3575
B.2.7	XSI Interprocess Communication.....	3576
B.2.8	Realtime .....	3577
B.2.9	Threads .....	3622
B.2.10	Sockets .....	3650
B.2.11	Tracing .....	3652
B.2.12	Data Types.....	3677
B.2.13	Status Information .....	3679
B.2.14	File Descriptor Allocation.....	3679

	B.3	System Interfaces.....	3679
	B.3.1	System Interfaces Removed in this Version .....	3679
	B.3.2	System Interfaces Removed in the Previous Version.....	3682
	B.3.3	Examples for Spawn .....	3682
<b>Part</b>	<b>C</b>	<b>Shell and Utilities .....</b>	<b>3693</b>
<b>Appendix</b>	<b>C</b>	<b>Rationale for Shell and Utilities.....</b>	<b>3695</b>
	C.1	Introduction .....	3695
	C.1.1	Change History .....	3695
	C.1.2	Relationship to Other Documents .....	3696
	C.1.3	Utility Limits.....	3697
	C.1.4	Grammar Conventions.....	3700
	C.1.5	Utility Description Defaults.....	3700
	C.1.6	Considerations for Utilities in Support of Files of Arbitrary Size .....	3704
	C.1.7	Built-In Utilities .....	3704
	C.2	Shell Command Language .....	3706
	C.2.1	Shell Introduction.....	3706
	C.2.2	Quoting.....	3706
	C.2.3	Token Recognition.....	3708
	C.2.4	Reserved Words.....	3709
	C.2.5	Parameters and Variables.....	3709
	C.2.6	Word Expansions .....	3715
	C.2.7	Redirection .....	3723
	C.2.8	Exit Status and Errors .....	3725
	C.2.9	Shell Commands .....	3725
	C.2.10	Shell Grammar.....	3733
	C.2.11	Signals and Error Handling.....	3734
	C.2.12	Shell Execution Environment.....	3734
	C.2.13	Pattern Matching Notation .....	3735
	C.2.14	Special Built-In Utilities.....	3736
	C.3	Batch Environment Services and Utilities .....	3736
	C.3.1	Batch General Concepts .....	3739
	C.3.2	Batch Services .....	3741
	C.3.3	Common Behavior for Batch Environment Utilities.....	3742
	C.4	Utilities.....	3742
	C.4.1	Utilities Removed in this Version .....	3742
	C.4.2	Utilities Removed in the Previous Version.....	3743
	C.4.3	Exclusion of Utilities.....	3743
<b>Part</b>	<b>D</b>	<b>Portability Considerations.....</b>	<b>3747</b>
<b>Appendix</b>	<b>D</b>	<b>Portability Considerations (Informative).....</b>	<b>3749</b>
	D.1	User Requirements.....	3749
	D.1.1	Configuration Interrogation .....	3750
	D.1.2	Process Management .....	3750
	D.1.3	Access to Data.....	3750
	D.1.4	Access to the Environment .....	3750
	D.1.5	Access to Determinism and Performance Enhancements.....	3750



D.1.6	Operating System-Dependent Profile .....	3751
D.1.7	I/O Interaction .....	3751
D.1.8	Internationalization Interaction .....	3751
D.1.9	C-Language Extensions.....	3751
D.1.10	Command Language .....	3751
D.1.11	Interactive Facilities .....	3751
D.1.12	Accomplish Multiple Tasks Simultaneously .....	3751
D.1.13	Complex Data Manipulation.....	3752
D.1.14	File Hierarchy Manipulation .....	3752
D.1.15	Locale Configuration .....	3752
D.1.16	Inter-User Communication.....	3752
D.1.17	System Environment .....	3752
D.1.18	Printing .....	3752
D.1.19	Software Development.....	3752
D.2	Portability Capabilities.....	3753
D.2.1	Configuration Interrogation .....	3753
D.2.2	Process Management .....	3754
D.2.3	Access to Data.....	3754
D.2.4	Access to the Environment .....	3755
D.2.5	Bounded (Realtime) Response .....	3756
D.2.6	Operating System-Dependent Profile .....	3756
D.2.7	I/O Interaction .....	3756
D.2.8	Internationalization Interaction .....	3757
D.2.9	C-Language Extensions.....	3757
D.2.10	Command Language .....	3757
D.2.11	Interactive Facilities .....	3758
D.2.12	Accomplish Multiple Tasks Simultaneously .....	3758
D.2.13	Complex Data Manipulation.....	3758
D.2.14	File Hierarchy Manipulation .....	3759
D.2.15	Locale Configuration .....	3759
D.2.16	Inter-User Communication.....	3759
D.2.17	System Environment .....	3760
D.2.18	Printing .....	3760
D.2.19	Software Development.....	3760
D.2.20	Future Growth .....	3760
D.3	Profiling Considerations .....	3761
D.3.1	Configuration Options .....	3761
D.3.2	Configuration Options (Shell and Utilities) .....	3761
D.3.3	Configurable Limits .....	3763
D.3.4	Configuration Options (System Interfaces).....	3763
D.3.5	Configurable Limits .....	3768
D.3.6	Optional Behavior .....	3771
<b>Part E</b>	<b>Subprofiling Considerations.....</b>	<b>3773</b>
<b>Appendix E</b>	<b>Subprofiling Considerations (Informative) .....</b>	<b>3775</b>
E.1	Subprofiling Option Groups.....	3775
	<b>Index .....</b>	<b>3781</b>

**List of Figures**

B-1	Example of a System with Typed Memory .....	3595
B-2	Trace System Overview: for Offline Analysis .....	3657
B-3	Trace System Overview: for Online Analysis .....	3659
B-4	Trace System Overview: States of a Trace Stream .....	3660
B-5	Trace Another Process .....	3670
B-6	Trace Name Space Overview: With Third-Party Library .....	3671

**List of Tables**

3-1	Job Control Job ID Formats.....	66
5-1	Escape Sequences and Associated Actions .....	121
6-1	Portable Character Set .....	125
6-2	Non-Portable Control Characters .....	130
7-1	Valid Character Class Combinations.....	142
10-1	Control Character Names .....	198
2-1	Value of Level for Socket Options.....	528
2-2	Socket-Level Options.....	529
2-3	Trace Option: System Trace Events.....	544
2-4	Trace and Trace Event Filter Options: System Trace Events.....	544
2-5	Trace and Trace Log Options: System Trace Events.....	545
2-6	Trace, Trace Log, and Trace Event Filter Options: System Trace Events .....	545
2-7	Trace Option: User Trace Event.....	546
1-1	Actions when Creating a File that Already Exists.....	2319
1-2	Selected ISO C Standard Operators and Control Flow Keywords .....	2322
1-3	Utility Limit Minimum Values .....	2323
1-4	Symbolic Utility Limits .....	2324
1-5	Regular Built-In Utilities .....	2334
3-1	Batch Utilities.....	2417
3-2	Environment Variable Summary .....	2421
3-3	Next State Table .....	2423
3-4	Results/Output Table .....	2425
3-5	Batch Services Summary .....	2432
A-1	Historical Practice for Symbolic Links.....	3497

# Trademarks

The following information is given for the convenience of users of POSIX.1-2008 and does not constitute an endorsement by the IEEE or The Open Group of these products. Equivalent products may be used if they can be shown to lead to the same results.

There may be other products mentioned in the text that might be covered by trademark protection and readers are advised to verify them independently.

AIX<sup>®</sup> and IBM<sup>®</sup> are registered trademarks of International Business Machines Corporation.

ArchiMate<sup>®</sup>, Jericho Forum<sup>®</sup>, Making Standards Work<sup>®</sup>, The Open Group<sup>®</sup>, TOGAF<sup>®</sup>, UNIX<sup>®</sup>, and the “X”<sup>®</sup> device are registered trademarks and Boundaryless Information Flow<sup>™</sup>, DirecNet<sup>™</sup>, FACE<sup>™</sup>, and The Open Group Certification Mark<sup>™</sup> are trademarks of The Open Group.

AT&T<sup>®</sup> is a registered trademark of AT&T in the USA and other countries.

BSD<sup>™</sup> is a trademark of the University of California, Berkeley, USA.

Hewlett-Packard<sup>®</sup>, HP<sup>®</sup>, and HP-UX<sup>®</sup> are registered trademarks of Hewlett-Packard Company.

IEEE<sup>®</sup> and POSIX<sup>®</sup> are registered trademarks, and 754<sup>™</sup>, 854<sup>™</sup>, 1003.0<sup>™</sup>, 1003.1<sup>™</sup>, 1003.1d<sup>™</sup>, 1003.1g<sup>™</sup>, 1003.1j<sup>™</sup>, 1003.1q<sup>™</sup>, 1003.2<sup>™</sup>, 1003.2a<sup>™</sup>, 1003.2d<sup>™</sup>, 1003.9<sup>™</sup>, and 1003.13<sup>™</sup> are trademarks of The Institute of Electrical and Electronic Engineers, Inc.

Linux<sup>®</sup> is a registered trademark of Linus Torvalds.

Sun<sup>®</sup> and Sun Microsystems<sup>®</sup> are registered trademarks of Oracle America, Inc.

/usr/group<sup>®</sup> is a registered trademark of UniForum, the International Network of UNIX System Users.

# *Acknowledgements*

The contributions of the following organizations to the development of POSIX.1-2008 are gratefully acknowledged:

- AT&T for permission to reproduce portions of its copyrighted System V Interface Definition (SVID) and material from the UNIX System V Release 2.0 documentation
- Hewlett-Packard Company, International Business Machines Corporation, Novell Inc., The Open Software Foundation, and Sun Microsystems Inc. for permission to reproduce portions of their copyrighted documentation
- ISO/IEC JTC 1/SC 22/WG 14 C Language Committee
- Red Hat Inc. for permission to reproduce portions of its copyrighted documentation

POSIX.1-2008 was prepared by the Austin Group, a joint working group of the IEEE, The Open Group, and ISO/IEC JTC 1/SC 22.

# Referenced Documents

## Normative References

Normative references for POSIX.1-2008 are defined in [Section 1.3](#) (on page 4).

## Informative References

The following documents are referenced in POSIX.1-2008:

1984 /usr/group Standard

/usr/group Standards Committee, Santa Clara, CA, UniForum 1984.

Almasi and Gottlieb

George S. Almasi and Allan Gottlieb, *Highly Parallel Computing*, The Benjamin/Cummings Publishing Company, Inc., 1989, ISBN: 0-8053-0177-1.

ANSI C

American National Standard for Information Systems: Standard X3.159-1989, Programming Language C.

ANSI X3.226-1994

American National Standard for Information Systems: Standard X3.226-1994, Programming Language Common LISP.

Brawer

Steven Brawer, *Introduction to Parallel Programming*, Academic Press, 1989, ISBN: 0-12-128470-0.

DeRemer and Pennello Article

DeRemer, Frank and Pennello, Thomas J., *Efficient Computation of LALR(1) Look-Ahead Sets*, SigPlan Notices, Volume 15, No. 8, August 1979.

Draft ANSI X3J11.1

IEEE Floating Point draft report of ANSI X3J11.1 (NCEG).

FIPS 151-1

Federal Information Procurement Standard (FIPS) 151-1. Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language].

FIPS 151-2

Federal Information Procurement Standards (FIPS) 151-2, Portable Operating System Interface (POSIX)—Part 1: System Application Program Interface (API) [C Language].

HP-UX Manual

Hewlett-Packard HP-UX Release 9.0 Reference Manual, Third Edition, August 1992.

IEC 60559: 1989

IEC 60559: 1989, Binary Floating-Point Arithmetic for Microprocessor Systems (previously designated IEC 559: 1989).

IEEE Standards Terms

IEEE 100, The Authoritative Dictionary of IEEE Standards Terms, Seventh Edition.

- IEEE Std 754™-1985  
IEEE Std 754-1985 (Reaff 1990), IEEE Standard for Binary Floating-Point Arithmetic.
- IEEE Std 854™-1987  
IEEE Std 854-1987, IEEE Standard for Radix-Independent Floating-Point Arithmetic.
- IEEE Std 1003.9™-1992  
IEEE Std 1003.9-1992, IEEE Standard for Information Technology — POSIX FORTRAN 77 Language Interfaces — Part 1: Binding for System Application Program Interface API.
- IETF RFC 791  
Internet Protocol, Version 4 (IPv4), September 1981 (available at: [www.ietf.org/rfc/rfc0791.txt](http://www.ietf.org/rfc/rfc0791.txt)).
- IETF RFC 819  
The Domain Naming Convention for Internet User Applications, Z. Su, J. Postel, August 1982 (available at: [www.ietf.org/rfc/rfc0819.txt](http://www.ietf.org/rfc/rfc0819.txt)).
- IETF RFC 822  
Standard for the Format of ARPA Internet Text Messages, D.H. Crocker, August 1982 (available at: [www.ietf.org/rfc/rfc0822.txt](http://www.ietf.org/rfc/rfc0822.txt)).
- IETF RFC 919  
Broadcasting Internet Datagrams, J. Mogul, October 1984 (available at: [www.ietf.org/rfc/rfc0919.txt](http://www.ietf.org/rfc/rfc0919.txt)).
- IETF RFC 920  
Domain Requirements, J. Postel, J. Reynolds, October 1984 (available at: [www.ietf.org/rfc/rfc0920.txt](http://www.ietf.org/rfc/rfc0920.txt)).
- IETF RFC 921  
Domain Name System Implementation Schedule, J. Postel, October 1984 (available at: [www.ietf.org/rfc/rfc0921.txt](http://www.ietf.org/rfc/rfc0921.txt)).
- IETF RFC 922  
Broadcasting Internet Datagrams in the Presence of Subnets, J. Mogul, October 1984 (available at: [www.ietf.org/rfc/rfc0922.txt](http://www.ietf.org/rfc/rfc0922.txt)).
- IETF RFC 1034  
Domain Names — Concepts and Facilities, P. Mockapetris, November 1987 (available at: [www.ietf.org/rfc/rfc1034.txt](http://www.ietf.org/rfc/rfc1034.txt)).
- IETF RFC 1035  
Domain Names — Implementation and Specification, P. Mockapetris, November 1987 (available at: [www.ietf.org/rfc/rfc1035.txt](http://www.ietf.org/rfc/rfc1035.txt)).
- IETF RFC 1123  
Requirements for Internet Hosts — Application and Support, R. Braden, October 1989 (available at: [www.ietf.org/rfc/rfc1123.txt](http://www.ietf.org/rfc/rfc1123.txt)).
- IETF RFC 1886  
DNS Extensions to Support Internet Protocol, Version 6 (IPv6), C. Huitema, S. Thomson, December 1995 (available at: [www.ietf.org/rfc/rfc1886.txt](http://www.ietf.org/rfc/rfc1886.txt)).
- IETF RFC 2045  
Multipurpose Internet Mail Extensions (MIME), Part 1: Format of Internet Message Bodies, N. Freed, N. Borenstein, November 1996 (available at: [www.ietf.org/rfc/rfc2045.txt](http://www.ietf.org/rfc/rfc2045.txt)).

*Referenced Documents*

IETF RFC 2181

Clarifications to the DNS Specification, R. Elz, R. Bush, July 1997 (available at: [www.ietf.org/rfc/rfc2181.txt](http://www.ietf.org/rfc/rfc2181.txt)).

IETF RFC 2373

Internet Protocol, Version 6 (IPv6) Addressing Architecture, S. Deering, R. Hinden, July 1998 (available at: [www.ietf.org/rfc/rfc2373.txt](http://www.ietf.org/rfc/rfc2373.txt)).

IETF RFC 2460

Internet Protocol, Version 6 (IPv6), S. Deering, R. Hinden, December 1998 (available at: [www.ietf.org/rfc/rfc2460.txt](http://www.ietf.org/rfc/rfc2460.txt)).

Internationalisation Guide

Guide, July 1993, Internationalisation Guide, Version 2 (ISBN: 1-859120-02-4, G304), published by The Open Group.

ISO 2375:1985

ISO 2375:1985, Data Processing — Procedure for Registration of Escape Sequences.

ISO 8652:1987

ISO 8652:1987, Programming Languages — Ada (technically identical to ANSI standard 1815A-1983).

ISO/IEC 1539:1991

ISO/IEC 1539:1991, Information Technology — Programming Languages — Fortran (technically identical to the ANSI X3.9-1978 standard [FORTRAN 77]).

ISO/IEC 4873:1991

ISO/IEC 4873:1991, Information Technology — ISO 8-bit Code for Information Interchange — Structure and Rules for Implementation.

ISO/IEC 6429:1992

ISO/IEC 6429:1992, Information Technology — Control Functions for Coded Character Sets.

ISO/IEC 6937:1994

ISO/IEC 6937:1994, Information Technology — Coded Graphic Character Set for Text Communication — Latin Alphabet.

ISO/IEC 8802-3:1996

ISO/IEC 8802-3:1996, Information Technology — Telecommunications and Information Exchange Between Systems — Local and Metropolitan Area Networks — Specific Requirements — Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications.

ISO/IEC 8859

ISO/IEC 8859, Information Technology — 8-Bit Single-Byte Coded Graphic Character Sets:

Part 1: Latin Alphabet No. 1

Part 2: Latin Alphabet No. 2

Part 3: Latin Alphabet No. 3

Part 4: Latin Alphabet No. 4

Part 5: Latin/Cyrillic Alphabet

Part 6: Latin/Arabic Alphabet

Part 7: Latin/Greek Alphabet

Part 8: Latin/Hebrew Alphabet

Part 9: Latin Alphabet No. 5

Part 10: Latin Alphabet No. 6

Part 11: Latin/Thai Alphabet

- Part 13: Latin Alphabet No. 7
- Part 14: Latin Alphabet No. 8 (Celtic)
- Part 15: Latin Alphabet No. 9
- Part 16: Latin Alphabet No. 10

ISO/IEC 9899:1990

ISO/IEC 9899:1990, Programming Languages — C, including Amendment 1:1995 (E), C Integrity (Multibyte Support Extensions (MSE) for ISO C).

ISO POSIX-1:1996

ISO/IEC 9945-1:1996, Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language] (identical to ANSI/IEEE Std 1003.1-1996). Incorporating ANSI/IEEE Stds 1003.1-1990, 1003.1b-1993, 1003.1c-1995, and 1003.1i-1995.

ISO POSIX-2:1993

ISO/IEC 9945-2:1993, Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities (identical to ANSI/IEEE Std 1003.2™-1992, as amended by ANSI/IEEE Std 1003.2a™-1992).

Issue 1

X/Open Portability Guide, July 1985 (ISBN: 0-444-87839-4).

Issue 2

X/Open Portability Guide, January 1987:

- Volume 1: XVS Commands and Utilities (ISBN: 0-444-70174-5)
- Volume 2: XVS System Calls and Libraries (ISBN: 0-444-70175-3)

Issue 3

X/Open Specification, 1988, 1989, February 1992:

- Commands and Utilities, Issue 3 (ISBN: 1-872630-36-7, C211); this specification was formerly X/Open Portability Guide, Issue 3, Volume 1, January 1989, XSI Commands and Utilities (ISBN: 0-13-685835-X, XO/XPG/89/002)
- System Interfaces and Headers, Issue 3 (ISBN: 1-872630-37-5, C212); this specification was formerly X/Open Portability Guide, Issue 3, Volume 2, January 1989, XSI System Interface and Headers (ISBN: 0-13-685843-0, XO/XPG/89/003)
- Curses Interface, Issue 3, contained in Supplementary Definitions, Issue 3 (ISBN: 1-872630-38-3, C213), Chapters 9 to 14 inclusive; this specification was formerly X/Open Portability Guide, Issue 3, Volume 3, January 1989, XSI Supplementary Definitions (ISBN: 0-13-685850-3, XO/XPG/89/004)
- Headers Interface, Issue 3, contained in Supplementary Definitions, Issue 3 (ISBN: 1-872630-38-3, C213), Chapter 19, Cpio and Tar Headers; this specification was formerly X/Open Portability Guide Issue 3, Volume 3, January 1989, XSI Supplementary Definitions (ISBN: 0-13-685850-3, XO/XPG/89/004)

Issue 4

CAE Specification, July 1992, published by The Open Group:

- System Interface Definitions (XBD), Issue 4 (ISBN: 1-872630-46-4, C204)
- Commands and Utilities (XCU), Issue 4 (ISBN: 1-872630-48-0, C203)
- System Interfaces and Headers (XSH), Issue 4 (ISBN: 1-872630-47-2, C202)



## *Referenced Documents*

### Issue 4, Version 2

CAE Specification, August 1994, published by The Open Group:

- System Interface Definitions (XBD), Issue 4, Version 2 (ISBN: 1-85912-036-9, C434)
- Commands and Utilities (XCU), Issue 4, Version 2 (ISBN: 1-85912-034-2, C436)
- System Interfaces and Headers (XSH), Issue 4, Version 2 (ISBN: 1-85912-037-7, C435)

### Issue 5

Technical Standard, February 1997, published by The Open Group:

- System Interface Definitions (XBD), Issue 5 (ISBN: 1-85912-186-1, C605)
- Commands and Utilities (XCU), Issue 5 (ISBN: 1-85912-191-8, C604)
- System Interfaces and Headers (XSH), Issue 5 (ISBN: 1-85912-181-0, C606)

### Issue 6

Technical Standard, April 2004, published by The Open Group:

- Base Definitions (XBD), Issue 6 (ISBN: 1-931624-43-7, C046)
- System Interfaces (XSH), Issue 6 (ISBN: 1-931624-44-5, C047)
- Shell and Utilities (XCU), Issue 6 (ISBN: 1-931624-45-3, C048)

### Knuth Article

Knuth, Donald E., *On the Translation of Languages from Left to Right*, Information and Control, Volume 8, No. 6, October 1965.

### KornShell

Bolsky, Morris I. and Korn, David G., *The New KornShell Command and Programming Language*, March 1995, Prentice Hall.

### MSE Working Draft

Working draft of ISO/IEC 9899:1990/Add3:Draft, Addendum 3 — Multibyte Support Extensions (MSE) as documented in the ISO Working Paper SC22/WG14/N205 dated 31 March 1992.

### POSIX.0: 1995

IEEE Std 1003.0™-1995, IEEE Guide to the POSIX Open System Environment (OSE) (identical to ISO/IEC TR 14252).

### POSIX.1: 1988

IEEE Std 1003.1™-1988, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language].

### POSIX.1: 1990

IEEE Std 1003.1™-1990, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) [C Language].

### POSIX.1a

P1003.1a, Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — (C Language) Amendment.

### POSIX.1d: 1999

IEEE Std 1003.1d™-1999, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) —

- Amendment 4: Additional Realtime Extensions [C Language].
- POSIX.1g: 2000  
IEEE Std 1003.1g™-2000, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — Amendment 6: Protocol-Independent Interfaces (PII).
- POSIX.1j: 2000  
IEEE Std 1003.1j™-2000, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — Amendment 5: Advanced Realtime Extensions [C Language].
- POSIX.1q: 2000  
IEEE Std 1003.1q™-2000, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application Program Interface (API) — Amendment 7: Tracing [C Language].
- POSIX.2: 1992  
IEEE Std 1003.2™-1992, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities.
- POSIX.2b  
P1003.2b, Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities — Amendment.
- POSIX.2d: 1994  
IEEE Std 1003.2d™-1994, IEEE Standard for Information Technology — Portable Operating System Interface (POSIX) — Part 2: Shell and Utilities — Amendment 1: Batch Environment.
- POSIX.13: 1998  
IEEE Std 1003.13™-1998, IEEE Standard for Information Technology — Standardized Application Environment Profile (AEP) — POSIX Realtime Application Support.
- Sarwate Article  
Sarwate, Dilip V., *Computation of Cyclic Redundancy Checks via Table Lookup*, Communications of the ACM, Volume 30, No. 8, August 1988.
- Sprunt, Sha, and Lehoczky  
Sprunt, B., Sha, L., and Lehoczky, J.P., *Aperiodic Task Scheduling for Hard Real-Time Systems*, The Journal of Real-Time Systems, Volume 1, 1989, Pages 27-60.
- SVID, Issue 1  
American Telephone and Telegraph Company, System V Interface Definition (SVID), Issue 1; Morristown, NJ, UNIX Press, 1985.
- SVID, Issue 2  
American Telephone and Telegraph Company, System V Interface Definition (SVID), Issue 2; Morristown, NJ, UNIX Press, 1986.
- SVID, Issue 3  
American Telephone and Telegraph Company, System V Interface Definition (SVID), Issue 3; Morristown, NJ, UNIX Press, 1989.
- The AWK Programming Language  
Aho, Alfred V., Kernighan, Brian W., and Weinberger, Peter J., *The AWK Programming Language*, Reading, MA, Addison-Wesley 1988.
- The C Programming Language  
Kernighan, Brian W. and Ritchie, Dennis M., *The C Programming Language*, Englewood Cliffs, NJ, Prentice Hall, 1st Edition (February 1978) ISBN 0-13-110163-3; 2nd Edition (March

## *Referenced Documents*

- 1988) ISBN 0-13-110362-8.
- UNIX Programmer's Manual  
American Telephone and Telegraph Company, *UNIX Time-Sharing System: UNIX Programmer's Manual*, 7th Edition, Murray Hill, NJ, Bell Telephone Laboratories, January 1979.
- XNS, Issue 4  
CAE Specification, August 1994, Networking Services, Issue 4 (ISBN: 1-85912-049-0, C438), published by The Open Group.
- XNS, Issue 5  
CAE Specification, February 1997, Networking Services, Issue 5 (ISBN: 1-85912-165-9, C523), published by The Open Group.
- XNS, Issue 5.2  
Technical Standard, January 2000, Networking Services (XNS), Issue 5.2 (ISBN: 1-85912-241-8, C808), published by The Open Group.
- X/Open Curses, Issue 4, Version 2  
CAE Specification, May 1996, X/Open Curses, Issue 4, Version 2 (ISBN: 1-85912-171-3, C610), published by The Open Group.
- Yacc  
*Yacc: Yet Another Compiler Compiler*, Stephen C. Johnson, 1978.

## **Source Documents**

Parts of the following documents were used to create the base documents for POSIX.1-2001:

- AIX 3.2 Manual  
AIX Version 3.2 For RISC System/6000, Technical Reference: Base Operating System and Extensions, 1990, 1992 (Part No. SC23-2382-00).
- OSF/1  
OSF/1 Programmer's Reference, Release 1.2 (ISBN: 0-13-020579-6).
- OSF AES  
Application Environment Specification (AES) Operating System Programming Interfaces Volume, Revision A (ISBN: 0-13-043522-8).
- System V Release 2.0  
— UNIX System V Release 2.0 Programmer's Reference Manual (April 1984 - Issue 2).  
— UNIX System V Release 2.0 Programming Guide (April 1984 - Issue 2).
- System V Release 4.2  
Operating System API Reference, UNIX<sup>®</sup> SVR4.2 (1992) (ISBN: 0-13-017658-3).

# **Standard for Information Technology— Portable Operating System Interface (POSIX®)**

## **Base Specifications, Issue 7**

Prepared by the Austin Group ([www.opengroup.org/austin](http://www.opengroup.org/austin)).

IMPORTANT NOTICE: This standard is not intended to assure safety, security, health, or environmental protection in all circumstances. Implementors of the standard are responsible for determining appropriate safety, security, environmental, and health practices or regulatory requirements.

This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading “Important Notice” or “Important Notices and Disclaimers Concerning IEEE Documents”. They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.



1       **Vol. 1:**  
2       **Base Definitions, Issue 7**

3       *The Open Group*  
4       *The Institute of Electrical and Electronics Engineers, Inc.*



# Introduction

## 1.1 Scope

POSIX.1-2008 defines a standard operating system interface and environment, including a command interpreter (or “shell”), and common utility programs to support applications portability at the source code level. It is intended to be used by both application developers and system implementors.

POSIX.1-2008 comprises four major components (each in an associated volume):

1. General terms, concepts, and interfaces common to all volumes of POSIX.1-2008, including utility conventions and C-language header definitions, are included in the Base Definitions volume of POSIX.1-2008.
2. Definitions for system service functions and subroutines, language-specific system services for the C programming language, function issues, including portability, error handling, and error recovery, are included in the System Interfaces volume of POSIX.1-2008.
3. Definitions for a standard source code-level interface to command interpretation services (a “shell”) and common utility programs for application programs are included in the Shell and Utilities volume of POSIX.1-2008.
4. Extended rationale that did not fit well into the rest of the document structure, containing historical information concerning the contents of POSIX.1-2008 and why features were included or discarded by the standard developers, is included in the Rationale (Informative) volume of POSIX.1-2008.

The following areas are outside of the scope of POSIX.1-2008:

- Graphics interfaces
- Database management system interfaces
- Record I/O considerations
- Object or binary code portability
- System configuration and resource availability

POSIX.1-2008 describes the external characteristics and facilities that are of importance to application developers, rather than the internal construction techniques employed to achieve these capabilities. Special emphasis is placed on those functions and facilities that are needed in a wide variety of commercial applications.

The facilities provided in POSIX.1-2008 are drawn from the following base documents:

- IEEE Std 1003.1, 2004 Edition (POSIX-1) (incorporating IEEE Std 1003.1-2001, IEEE Std 1003.1-2001/Cor 1-2002, and IEEE Std 1003.1-2001/Cor 2-2004)

- The Open Group Technical Standard, 2006, Extended API Set Part 1
- The Open Group Technical Standard, 2006, Extended API Set Part 2
- The Open Group Technical Standard, 2006, Extended API Set Part 3
- The Open Group Technical Standard, 2006, Extended API Set Part 4
- ISO/IEC 9899:1999, Programming Languages — C, including ISO/IEC 9899:1999/Cor.1:2001(E), ISO/IEC 9899:1999/Cor.2:2004(E), and ISO/IEC 9899:1999/Cor.3

Emphasis has been placed on standardizing existing practice for existing users, with changes and additions limited to correcting deficiencies in the following areas:

- Issues raised by Austin Group defect reports, IEEE Interpretations against IEEE Std 1003.1, and ISO/IEC defect reports against ISO/IEC 9945
- Issues raised in corrigenda for The Open Group Technical Standards and working group resolutions from The Open Group
- Issues arising from ISO TR 24715:2006, Conflicts between POSIX and the LSB
- Changes to make the text self-consistent with the additional material merged
- Features, marked Legacy or obsolescent in the base documents, have been considered for removal in this version
- A review and reorganization of the options within the standard
- Alignment with the ISO/IEC 9899:1999 standard, including ISO/IEC 9899:1999/Cor.2:2004(E)

## 1.2 Conformance

Conformance requirements for POSIX.1-2008 are defined in [Chapter 2](#) (on page 15).

## 1.3 Normative References

The following standards contain provisions which, through references in POSIX.1-2008, constitute provisions of POSIX.1-2008. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on POSIX.1-2008 are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

ANS X3.9-1978

(Reaffirmed 1989) American National Standard for Information Systems: Standard X3.9-1978, Programming Language FORTRAN.<sup>1</sup>

ISO/IEC 646:1991

ISO/IEC 646:1991, Information Processing — ISO 7-Bit Coded Character Set for Information Interchange.<sup>2</sup>

---

1. ANSI documents can be obtained from the Sales Department, American National Standards Institute, 1430 Broadway, New York, NY 10018, USA.



- ISO 4217:2001  
ISO 4217:2001, Codes for the Representation of Currencies and Funds.
- ISO 8601:2004  
ISO 8601:2004, Data Elements and Interchange Formats — Information Interchange — Representation of Dates and Times.
- ISO C (1999)  
ISO/IEC 9899:1999, Programming Languages — C, including ISO/IEC 9899:1999/Cor.1:2001(E), ISO/IEC 9899:1999/Cor.2:2004(E), and ISO/IEC 9899:1999/Cor.3.
- ISO/IEC 10646-1:2000  
ISO/IEC 10646-1:2000, Information Technology — Universal Multiple-Octet Coded Character Set (UCS) — Part 1: Architecture and Basic Multilingual Plane.

## 1.4 Change History

Change history is described in the Rationale (Informative) volume of POSIX.1-2008, and in the CHANGE HISTORY section of reference pages.

## 1.5 Terminology

For the purposes of POSIX.1-2008, the following terminology definitions apply:

### **can**

Describes a permissible optional feature or behavior available to the user or application. The feature or behavior is mandatory for an implementation that conforms to POSIX.1-2008. An application can rely on the existence of the feature or behavior.

### **implementation-defined**

Describes a value or behavior that is not defined by POSIX.1-2008 but is selected by an implementor. The value or behavior may vary among implementations that conform to POSIX.1-2008. An application should not rely on the existence of the value or behavior. An application that relies on such a value or behavior cannot be assured to be portable across conforming implementations.

The implementor shall document such a value or behavior so that it can be used correctly by an application.

### **legacy**

Describes a feature or behavior that is being retained for compatibility with older applications, but which has limitations which make it inappropriate for developing portable applications. New applications should use alternative means of obtaining equivalent functionality.

### **may**

Describes a feature or behavior that is optional for an implementation that conforms to POSIX.1-2008. An application should not rely on the existence of the feature or behavior. An application that relies on such a feature or behavior cannot be assured to be portable across conforming implementations.

---

2. ISO/IEC documents can be obtained from the ISO office: 1 Rue de Varembe, Case Postale 56, CH-1211, Genève 20, Switzerland/Suisse

To avoid ambiguity, the opposite of *may* is expressed as *need not*, instead of *may not*.

#### **shall**

For an implementation that conforms to POSIX.1-2008, describes a feature or behavior that is mandatory. An application can rely on the existence of the feature or behavior.

For an application or user, describes a behavior that is mandatory.

#### **should**

For an implementation that conforms to POSIX.1-2008, describes a feature or behavior that is recommended but not mandatory. An application should not rely on the existence of the feature or behavior. An application that relies on such a feature or behavior cannot be assured to be portable across conforming implementations.

For an application, describes a feature or behavior that is recommended programming practice for optimum portability.

#### **undefined**

Describes the nature of a value or behavior not defined by POSIX.1-2008 which results from use of an invalid program construct or invalid data input.

The value or behavior may vary among implementations that conform to POSIX.1-2008. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

#### **unspecified**

Describes the nature of a value or behavior not specified by POSIX.1-2008 which results from use of a valid program construct or valid data input.

The value or behavior may vary among implementations that conform to POSIX.1-2008. An application should not rely on the existence or validity of the value or behavior. An application that relies on any particular value or behavior cannot be assured to be portable across conforming implementations.

## **1.6 Definitions and Concepts**

Definitions and concepts are defined in [Chapter 3](#) (on page 33) and [Chapter 4](#) (on page 107).

## **1.7 Portability**

Some of the utilities in the Shell and Utilities volume of POSIX.1-2008 and functions in the System Interfaces volume of POSIX.1-2008 describe functionality that might not be fully portable to systems meeting the requirements for POSIX conformance (see [Chapter 2](#), on page 15).

Where optional, enhanced, or reduced functionality is specified, the text is shaded and a code in the margin identifies the nature of the option, extension, or warning (see [Section 1.7.1](#), on page 7). For maximum portability, an application should avoid such functionality.

Unless the primary task of a utility is to produce textual material on its standard output, application developers should not rely on the format or content of any such material that may be produced. Where the primary task *is* to provide such material, but the output format is incompletely specified, the description is marked with the OF margin code and shading. Application developers are warned not to expect that the output of such an interface on one

system is any guide to its behavior on another system.

### 1.7.1 Codes

The codes and their meanings are as follows. See also [Section 1.7.2](#) (on page 13).

#### ADV **Advisory Information**

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the ADV margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the ADV margin legend.

#### BE **Batch Environment Services and Utilities**

The functionality described is optional.

Where applicable, utilities are marked with the BE margin legend in the SYNOPSIS section. Where additional semantics apply to a utility, the material is identified by use of the BE margin legend.

#### CD **C-Language Development Utilities**

The functionality described is optional.

Where applicable, utilities are marked with the CD margin legend in the SYNOPSIS section. Where additional semantics apply to a utility, the material is identified by use of the CD margin legend.

#### CPT **Process CPU-Time Clocks**

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the CPT margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the CPT margin legend.

#### CX **Extension to the ISO C standard**

The functionality described is an extension to the ISO C standard. Application developers may make use of an extension as it is supported on all POSIX.1-2008-conforming systems.

With each function or header from the ISO C standard, a statement to the effect that “any conflict is unintentional” is included. That is intended to refer to a direct conflict. POSIX.1-2008 acts in part as a profile of the ISO C standard, and it may choose to further constrain behaviors allowed to vary by the ISO C standard. Such limitations and other compatible differences are not considered conflicts, even if a CX mark is missing. The markings are for information only.

Where additional semantics apply to a function or header, the material is identified by use of the CX margin legend.

#### FD **FORTRAN Development Utilities**

The functionality described is optional.

Where applicable, utilities are marked with the FD margin legend in the SYNOPSIS section. Where additional semantics apply to a utility, the material is identified by use of the FD margin legend.

#### FR **FORTRAN Runtime Utilities**

The functionality described is optional.

Where applicable, utilities are marked with the FR margin legend in the SYNOPSIS section.

200		Where additional semantics apply to a utility, the material is identified by use of the FR margin legend.
201		
202	FSC	<b>File Synchronization</b>
203		The functionality described is optional. The functionality described is also an extension to the
204		ISO C standard.
205		Where applicable, functions are marked with the FSC margin legend in the SYNOPSIS section.
206		Where additional semantics apply to a function, the material is identified by use of the FSC
207		margin legend.
208	IP6	<b>IPV6</b>
209		The functionality described is optional. The functionality described is also an extension to the
210		ISO C standard.
211		Where applicable, functions are marked with the IP6 margin legend in the SYNOPSIS section.
212		Where additional semantics apply to a function, the material is identified by use of the IP6
213		margin legend.
214	MC1	<b>Non-Robust Mutex Priority Protection or Non-Robust Mutex Priority Inheritance or Robust</b>
215		<b>Mutex Priority Protection or Robust Mutex Priority Inheritance</b>
216		The functionality described is optional. The functionality described is also an extension to the
217		ISO C standard.
218		This is a shorthand notation for combinations of multiple option codes.
219		Where applicable, functions are marked with the MC1 margin legend in the SYNOPSIS section.
220		Where additional semantics apply to a function, the material is identified by use of the MC1
221		margin legend.
222		Refer to <a href="#">Section 1.7.2</a> (on page 13).
223	ML	<b>Process Memory Locking</b>
224		The functionality described is optional. The functionality described is also an extension to the
225		ISO C standard.
226		Where applicable, functions are marked with the ML margin legend in the SYNOPSIS section.
227		Where additional semantics apply to a function, the material is identified by use of the ML
228		margin legend.
229	MLR	<b>Range Memory Locking</b>
230		The functionality described is optional. The functionality described is also an extension to the
231		ISO C standard.
232		Where applicable, functions are marked with the MLR margin legend in the SYNOPSIS section.
233		Where additional semantics apply to a function, the material is identified by use of the MLR
234		margin legend.
235	MON	<b>Monotonic Clock</b>
236		The functionality described is optional. The functionality described is also an extension to the
237		ISO C standard.
238		Where applicable, functions are marked with the MON margin legend in the SYNOPSIS section.
239		Where additional semantics apply to a function, the material is identified by use of the MON
240		margin legend.
241	MSG	<b>Message Passing</b>
242		The functionality described is optional. The functionality described is also an extension to the
243		ISO C standard.
244		Where applicable, functions are marked with the MSG margin legend in the SYNOPSIS section.

Where additional semantics apply to a function, the material is identified by use of the MSG margin legend.

#### MX IEC 60559 Floating-Point

The functionality described is optional. The functionality described is mandated by the ISO C standard only for implementations that define `__STDC_IEC_559__`.

#### MXX IEC 60559 Floating-Point Extension

The functionality described is optional. The functionality described is part of the IEC 60559 Floating-Point option, but is an extension to the ISO C standard.

#### OB Obsolescent

The functionality described may be removed in a future version of this volume of POSIX.1-2008. Strictly Conforming POSIX Applications and Strictly Conforming XSI Applications shall not use obsolescent features.

Where applicable, the material is identified by use of the OB margin legend.

#### OF Output Format Incompletely Specified

The functionality described is an XSI extension. The format of the output produced by the utility is not fully specified. It is therefore not possible to post-process this output in a consistent fashion. Typical problems include unknown length of strings and unspecified field delimiters.

Where applicable, the material is identified by use of the OF margin legend.

#### OH Optional Header

In the SYNOPSIS section of some interfaces in the System Interfaces volume of POSIX.1-2008 an included header is marked as in the following example:

```
#include <sys/types.h>
#include <fcntl.h>
struct open(const char *path, int oflag, ...);
```

The OH margin legend indicates that the optional header defines constants that will be needed if the function is called with certain flag arguments; thus it may be required for some of the functionality described, but is not needed otherwise.

#### PIO Prioritized Input and Output

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the PIO margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the PIO margin legend.

#### PS Process Scheduling

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the PS margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the PS margin legend.

#### RPI Robust Mutex Priority Inheritance

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the RPI margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the RPI margin legend.

290	RPP	<b>Robust Mutex Priority Protection</b>
291		The functionality described is optional. The functionality described is also an extension to the
292		ISO C standard.
293		Where applicable, functions are marked with the RPP margin legend in the SYNOPSIS section.
294		Where additional semantics apply to a function, the material is identified by use of the RPP
295		margin legend.
296	RS	<b>Raw Sockets</b>
297		The functionality described is optional. The functionality described is also an extension to the
298		ISO C standard.
299		Where applicable, functions are marked with the RS margin legend in the SYNOPSIS section.
300		Where additional semantics apply to a function, the material is identified by use of the RS
301		margin legend.
302	SD	<b>Software Development Utilities</b>
303		The functionality described is optional.
304		Where applicable, utilities are marked with the SD margin legend in the SYNOPSIS section.
305		Where additional semantics apply to a utility, the material is identified by use of the SD margin
306		legend.
307	SHM	<b>Shared Memory Objects</b>
308		The functionality described is optional. The functionality described is also an extension to the
309		ISO C standard.
310		Where applicable, functions are marked with the SHM margin legend in the SYNOPSIS section.
311		Where additional semantics apply to a function, the material is identified by use of the SHM
312		margin legend.
313	SIO	<b>Synchronized Input and Output</b>
314		The functionality described is optional. The functionality described is also an extension to the
315		ISO C standard.
316		Where applicable, functions are marked with the SIO margin legend in the SYNOPSIS section.
317		Where additional semantics apply to a function, the material is identified by use of the SIO
318		margin legend.
319	SPN	<b>Spawn</b>
320		The functionality described is optional. The functionality described is also an extension to the
321		ISO C standard.
322		Where applicable, functions are marked with the SPN margin legend in the SYNOPSIS section.
323		Where additional semantics apply to a function, the material is identified by use of the SPN
324		margin legend.
325	SS	<b>Process Sporadic Server</b>
326		The functionality described is optional. The functionality described is also an extension to the
327		ISO C standard.
328		Where applicable, functions are marked with the SS margin legend in the SYNOPSIS section.
329		Where additional semantics apply to a function, the material is identified by use of the SS
330		margin legend.
331	TCT	<b>Thread CPU-Time Clocks</b>
332		The functionality described is optional. The functionality described is also an extension to the
333		ISO C standard.
334		Where applicable, functions are marked with the TCT margin legend in the SYNOPSIS section.



Where additional semantics apply to a function, the material is identified by use of the TCT margin legend.

#### Trace Event Filter

The functionality described is optional. This functionality is dependent on support for the Trace option. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the TEF margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the TEF margin legend.

#### Non-Robust Mutex Priority Inheritance

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the TPI margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the TPI margin legend.

#### Non-Robust Mutex Priority Protection

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the TPP margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the TPP margin legend.

#### Thread Execution Scheduling

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the TPS margin legend for the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the TPS margin legend.

#### Trace

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the TRC margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the TRC margin legend.

#### Trace Inherit

The functionality described is optional. This functionality is dependent on support for the Trace option. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the TRI margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the TRI margin legend.

#### Trace Log

The functionality described is optional. This functionality is dependent on support for the Trace option. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the TRL margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the TRL margin legend.

379	TSA	<b>Thread Stack Address Attribute</b>
380		The functionality described is optional. The functionality described is also an extension to the
381		ISO C standard.
382		Where applicable, functions are marked with the TSA margin legend for the SYNOPSIS section.
383		Where additional semantics apply to a function, the material is identified by use of the TSA
384		margin legend.
385	TSH	<b>Thread Process-Shared Synchronization</b>
386		The functionality described is optional. The functionality described is also an extension to the
387		ISO C standard.
388		Where applicable, functions are marked with the TSH margin legend in the SYNOPSIS section.
389		Where additional semantics apply to a function, the material is identified by use of the TSH
390		margin legend.
391	TSP	<b>Thread Sporadic Server</b>
392		The functionality described is optional. The functionality described is also an extension to the
393		ISO C standard.
394		Where applicable, functions are marked with the TSP margin legend in the SYNOPSIS section.
395		Where additional semantics apply to a function, the material is identified by use of the TSP
396		margin legend.
397	TSS	<b>Thread Stack Size Attribute</b>
398		The functionality described is optional. The functionality described is also an extension to the
399		ISO C standard.
400		Where applicable, functions are marked with the TSS margin legend in the SYNOPSIS section.
401		Where additional semantics apply to a function, the material is identified by use of the TSS
402		margin legend.
403	TYM	<b>Typed Memory Objects</b>
404		The functionality described is optional. The functionality described is also an extension to the
405		ISO C standard.
406		Where applicable, functions are marked with the TYM margin legend in the SYNOPSIS section.
407		Where additional semantics apply to a function, the material is identified by use of the TYM
408		margin legend.
409	UP	<b>User Portability Utilities</b>
410		The functionality described is optional.
411		Where applicable, utilities are marked with the UP margin legend in the SYNOPSIS section.
412		Where additional semantics apply to a utility, the material is identified by use of the UP margin
413		legend.
414	UU	<b>UUCP Utilities</b>
415		The functionality described is optional. The functionality described is also an extension to the
416		ISO C standard.
417		Where applicable, functions are marked with the UU margin legend in the SYNOPSIS section.
418		Where additional semantics apply to a function, the material is identified by use of the UU
419		margin legend.
420	XSI	<b>X/Open System Interfaces</b>
421		The functionality described is part of the X/Open Systems Interfaces option. Functionality
422		marked XSI is an extension to the ISO C standard. Application developers may confidently
423		make use of such extensions on all systems supporting the X/Open System Interfaces option.



If an entire SYNOPSIS section is shaded and marked XSI, all the functionality described in that reference page is an extension. See [Section 2.1.4](#) (on page 19).

XSR

**XSI STREAMS**

The functionality described is optional. The functionality described is also an extension to the ISO C standard.

Where applicable, functions are marked with the XSR margin legend in the SYNOPSIS section. Where additional semantics apply to a function, the material is identified by use of the XSR margin legend.

**1.7.2 Margin Code Notation**

Some of the functionality described in POSIX.1-2008 depends on support of more than one option, or independently may depend on several options. The following notation for margin codes is used to denote the following cases.

**A Feature Dependent on One or Two Options**

In this case, margin codes have a <space> separator; for example:

SHM

This feature requires support for only the Shared Memory Objects option.

SHM TYM

This feature requires support for both the Shared Memory Objects option and the Typed Memory Objects option; that is, an application which uses this feature is portable only between implementations that provide both options.

**A Feature Dependent on Either of the Options Denoted**

In this case, margin codes have a ' | ' separator to denote the logical OR; for example:

SHM | TYM

This feature is dependent on support for either the Shared Memory Objects option or the Typed Memory Objects option; that is, an application which uses this feature is portable between implementations that provide any (or all) of the options.

**A Feature Dependent on More than Two Options**

The following shorthand notations are used:

MC1

The MC1 margin code is shorthand for TPP | TPI | RPP | RPI. Features which are shaded with this margin code require support of either the Non-Robust Mutex Priority Protection option or the Non-Robust Mutex Priority Inheritance option or the Robust Mutex Priority Protection option or the Robust Mutex Priority Inheritance option.

**Large Sections Dependent on an Option**

Where large sections of text are dependent on support for an option, a lead-in text block is provided and shaded accordingly; for example:

XSI

This section describes extensions to support interprocess communication. The functionality described in this section shall be provided on implementations that support the XSI option (and the rest of this section is not further shaded).



## 2.1 Implementation Conformance

For the purposes of POSIX.1-2008, the implementation conformance requirements given in this section apply.

### 2.1.1 Requirements

A *conforming implementation* shall meet all of the following criteria:

1. The system shall support all utilities, functions, and facilities defined within POSIX.1-2008 that are required for POSIX conformance (see [Section 2.1.3](#), on page 17). These interfaces shall support the functional behavior described herein.
2. The system may support the X/Open System Interfaces (XSI) option as described in [Section 2.1.4](#) (on page 19).
3. The system may support one or more options as described under [Section 2.1.5](#) (on page 20). When an implementation claims that an option is supported, all of its constituent parts shall be provided.
4. The system may provide non-standard extensions. These are features not required by POSIX.1-2008 and may include, but are not limited to:
  - Additional functions
  - Additional headers
  - Additional symbols in standard headers
  - Additional utilities
  - Additional options for standard utilities
  - Additional environment variables
  - Additional file types
  - Non-conforming file systems (for example, legacy file systems for which `_POSIX_NO_TRUNC` is false, case-insensitive file systems, or network file systems)
  - Dynamically populated file systems (for example, `/proc`)
  - Additional character special files with special properties (for example, `/dev/stdin`, `/dev/stdout`, and `/dev/stderr`)

Non-standard extensions of the utilities, functions, or facilities specified in POSIX.1-2008 should be identified as such in the system documentation. Non-standard extensions, when used, may change the behavior of utilities, functions, or facilities defined by POSIX.1-2008. The conformance document shall define an environment in which an application can be run with the behavior specified by POSIX.1-2008. In no case shall such

an environment require modification of a Strictly Conforming POSIX Application (see Section 2.2.1, on page 29).

**Note:** If the documented method of setting up a conforming environment includes the need to set one or more environment variables, then the values of those environment variables cannot include any <space> characters, since the *confstr()* function must be able to return them in a <space>-separated list of variable=value pairs. See XSH *confstr()* (on page 696).

## 2.1.2 Documentation

A conformance document with the following information shall be available for an implementation claiming conformance to POSIX.1-2008. The conformance document shall have the same structure as POSIX.1-2008, with the information presented in the appropriate sections and subsections. Sections and subsections that consist solely of subordinate section titles, with no other information, are not required. The conformance document shall not contain information about extended facilities or capabilities outside the scope of POSIX.1-2008.

The conformance document shall contain a statement that indicates the full name, number, and date of the standard that applies. The conformance document may also list international software standards that are available for use by a Conforming POSIX Application. Applicable characteristics where documentation is required by one of these standards, or by standards of government bodies, may also be included.

The conformance document shall describe the limit values found in the headers *<limits.h>* (on page 270) and *<unistd.h>* (on page 433), stating values, the conditions under which those values may change, and the limits of such variations, if any.

The conformance document shall describe the behavior of the implementation for all implementation-defined features defined in POSIX.1-2008. This requirement shall be met by listing these features and providing either a specific reference to the system documentation or providing full syntax and semantics of these features. When the value or behavior in the implementation is designed to be variable or customized on each instantiation of the system, the implementation provider shall document the nature and permissible ranges of this variation.

The conformance document may specify the behavior of the implementation for those features where POSIX.1-2008 states that implementations may vary or where features are identified as undefined or unspecified.

The conformance document shall not contain documentation other than that specified in the preceding paragraphs except where such documentation is specifically allowed or required by other provisions of POSIX.1-2008.

The phrases “shall document” or “shall be documented” in POSIX.1-2008 mean that documentation of the feature shall appear in the conformance document, as described previously, unless there is an explicit reference in the conformance document to show where the information can be found in the system documentation.

The system documentation should also contain the information found in the conformance document.

### 2.1.3 POSIX Conformance

A conforming implementation shall meet the following criteria for POSIX conformance.

#### 2.1.3.1 POSIX System Interfaces

The following requirements apply to the system interfaces (functions and headers):

- The system shall support all the mandatory functions and headers defined in POSIX.1-2008, and shall set the symbolic constant `_POSIX_VERSION` to the value 200809L.
- Although all implementations conforming to POSIX.1-2008 support all the features described below, there may be system-dependent or file system-dependent configuration procedures that can remove or modify any or all of these features. Such configurations should not be made if strict compliance is required.

The following symbolic constants shall be defined with a value other than `-1`. If a constant is defined with the value zero, applications should use the `sysconf()`, `pathconf()`, or `fpathconf()` functions, or the `getconf` utility, to determine which features are present on the system at that time or for the particular pathname in question.

— `_POSIX_CHOWN_RESTRICTED`

The use of `chown()` is restricted to a process with appropriate privileges, and to changing the group ID of a file only to the effective group ID of the process or to one of its supplementary group IDs.

— `_POSIX_NO_TRUNC`

Pathname components longer than `{NAME_MAX}` generate an error.

- The following symbolic constants shall be defined by the implementation as follows:

— Symbolic constants defined with the value 200809L:

```

_POSIX_ASYNCHRONOUS_IO
_POSIX_BARRIERS
_POSIX_CLOCK_SELECTION
_POSIX_MAPPED_FILES
_POSIX_MEMORY_PROTECTION
_POSIX_READER_WRITER_LOCKS
_POSIX_REALTIME_SIGNALS
_POSIX_SEMAPHORES
_POSIX_SPIN_LOCKS
_POSIX_THREAD_SAFE_FUNCTIONS
_POSIX_THREADS
_POSIX_TIMEOUTS
_POSIX_TIMERS
_POSIX2_C_BIND

```

+

— Symbolic constants defined with a value greater than zero:

```

_POSIX_JOB_CONTROL
_POSIX_REGEX
_POSIX_SAVED_IDS
_POSIX_SHELL

```

+

+

— Symbolic constants defined with a value other than `-1`.

`_POSIX_VDISABLE`

**Note:** The symbols above represent historical options that are no longer allowed as options, but are retained here for backwards-compatibility of applications.

- The system may support one or more options (see [Section 2.1.6](#), on page 26) denoted by the following symbolic constants:

`_POSIX_ADVISORY_INFO`  
`_POSIX_CPUTIME`  
`_POSIX_FSYNC`  
`_POSIX_IPV6`  
`_POSIX_MEMLOCK`  
`_POSIX_MEMLOCK_RANGE`  
`_POSIX_MESSAGE_PASSING`  
`_POSIX_MONOTONIC_CLOCK`  
`_POSIX_PRIORITIZED_IO`  
`_POSIX_PRIORITY_SCHEDULING`  
`_POSIX_RAW_SOCKETS`  
`_POSIX_SHARED_MEMORY_OBJECTS`  
`_POSIX_SPAWN`  
`_POSIX_SPORADIC_SERVER`  
`_POSIX_SYNCHRONIZED_IO`  
`_POSIX_THREAD_ATTR_STACKADDR`  
`_POSIX_THREAD_CPUTIME`  
`_POSIX_THREAD_ATTR_STACKSIZE`  
`_POSIX_THREAD_PRIO_INHERIT`  
`_POSIX_THREAD_PRIO_PROTECT`  
`_POSIX_THREAD_PRIORITY_SCHEDULING`  
`_POSIX_THREAD_PROCESS_SHARED`  
`_POSIX_THREAD_SPAWNING`  
`_POSIX_TRACE`  
`_POSIX_TRACE_EVENT_FILTER`  
`_POSIX_TRACE_INHERIT`  
`_POSIX_TRACE_LOG`  
`_POSIX_TYPED_MEMORY_OBJECTS`  
`_XOPEN_CRYPT`  
`_XOPEN_REALTIME`  
`_XOPEN_REALTIME_THREADS`  
`_XOPEN_STREAMS`  
`_XOPEN_UNIX`

If any of the symbolic constants `_POSIX_TRACE_EVENT_FILTER`, `_POSIX_TRACE_LOG`, or `_POSIX_TRACE_INHERIT` is defined to have a value other than `-1`, then the symbolic constant `_POSIX_TRACE` shall also be defined to have a value other than `-1`.

If the Advisory Information option is supported, there shall be at least one file system that supports the functionality.

2.1.3.2 *POSIX Shell and Utilities*

The following requirements apply to the shell and utilities:

- The system shall provide all the mandatory utilities in the Shell and Utilities volume of POSIX.1-2008 with all the functional behavior described therein.
- The system shall support the Large File capabilities described in the Shell and Utilities volume of POSIX.1-2008.
- The system may support one or more options (see [Section 2.1.6](#), on page 26) denoted by the following symbolic constants. (The literal names below apply to the *getconf* utility.)

```

POSIX2_C_DEV
POSIX2_CHAR_TERM
POSIX2_FORT_DEV
POSIX2_FORT_RUN
POSIX2_LOCALEDEF
POSIX2_PBS
POSIX2_PBS_ACCOUNTING
POSIX2_PBS_LOCATE
POSIX2_PBS_MESSAGE
POSIX2_PBS_TRACK
POSIX2_SW_DEV
POSIX2_UPE
XOPEN_UNIX
XOPEN_UUCP

```

Additional language bindings and development utility options may be provided in other related standards or in a future version of this standard. In the former case, additional symbolic constants of the same general form as shown in this subsection should be defined by the related standard document and made available to the application without requiring POSIX.1-2008 to be updated.

2.1.4 **XSI Conformance**

XSI This section describes the criteria for implementations providing conformance to the X/Open System Interfaces (XSI) option (see [Section 3.450](#), on page 105). The functionality described in this section shall be provided on implementations that support the XSI option (and the rest of this section is not further shaded).

POSIX.1-2008 describes utilities, functions, and facilities offered to application programs by the X/Open System Interfaces (XSI) option. An XSI-conforming implementation shall meet the criteria for POSIX conformance and the following requirements listed in this section.

XSI-conforming implementations shall set the symbolic constant `_XOPEN_UNIX` to a value other than `-1` and shall set the symbolic constant `_XOPEN_VERSION` to the value `700`.

2.1.4.1 *XSI System Interfaces*

The following requirements apply to the system interfaces when the XSI option is supported:

- The system shall support all the functions and headers defined in POSIX.1-2008 as part of the XSI option denoted by the XSI marking in the SYNOPSIS section, and any extensions marked with the XSI option marking (see [Section 1.7.1](#), on page 7) within the text.

- The system shall support the following options defined within POSIX.1-2008 (see [Section 2.1.6](#), on page 26):

```
_POSIX_FSYNC
_POSIX_THREAD_ATTR_STACKADDR
_POSIX_THREAD_ATTR_STACKSIZE
_POSIX_THREAD_PROCESS_SHARED
```

- The system may support the following XSI Option Groups (see [Section 2.1.5.2](#), on page 22) defined within POSIX.1-2008:

```
— Encryption
— Realtime
— Advanced Realtime
— Realtime Threads
— Advanced Realtime Threads
— Tracing
— XSI STREAMS
```

#### 2.1.4.2 XSI Shell and Utilities Conformance

The following requirements apply to the shell and utilities when the XSI option is supported:

- The system shall support all the utilities defined in the Shell and Utilities volume of POSIX.1-2008 as part of the XSI option denoted by the XSI marking in the SYNOPSIS section, and any extensions marked with the XSI option marking (see [Section 1.7.1](#), on page 7) within the text.
- The system shall support the User Portability Utilities option and the Terminal Characteristics option.
- The system shall support creation of locales (see [Chapter 7](#), on page 135).
- The C-language Development utility *c99* shall be supported.
- The XSI Development Utilities option may be supported. It consists of the following software development utilities:

```
admin  delta  rmdel  val
cflow  get    sact   what
ctags  nm     sccs
cxref  prs    unget
```

### 2.1.5 Option Groups

An Option Group is a group of related functions or options defined within the System Interfaces volume of POSIX.1-2008.

If an implementation supports an Option Group, then the system shall support the functional behavior described herein.

If an implementation does not support an Option Group, then the system need not support the functional behavior described herein.



## 2.1.5.1 Subprofiling Considerations

Profiling standards supporting functional requirements less than that required in POSIX.1-2008 may subset both mandatory and optional functionality required for POSIX Conformance (see [Section 2.1.3](#), on page 17) or XSI Conformance (see [Section 2.1.4](#), on page 19). Such profiles shall organize the subsets into Subprofiling Option Groups.

X RAT [Appendix E](#) (on page 3775) describes a representative set of such Subprofiling Option Groups for use by profiles applicable to specialized realtime systems. POSIX.1-2008 does not require that the presence of Subprofiling Option Groups be testable at compile-time (as symbols defined in any header) or at runtime (via *sysconf()* or *getconf()*).

A Subprofiling Option Group may provide basic system functionality that other Subprofiling Option Groups and other options depend upon.<sup>3</sup> If a profile of POSIX.1-2008 does not require an implementation to provide a Subprofiling Option Group that provides features utilized by a required Subprofiling Option Group (or option),<sup>4</sup> the profile shall specify<sup>5</sup> all of the following:

- Restricted or altered behavior of interfaces defined in POSIX.1-2008 that may differ on an implementation of the profile
- Additional behaviors that may produce undefined or unspecified results
- Additional implementation-defined behavior that implementations shall be required to document in the profile's conformance document

if any of the above is a result of the profile not requiring an interface required by POSIX.1-2008.

The following additional rules shall apply to all profiles of POSIX.1-2008:

- Any application that conforms to that profile shall also conform to POSIX.1-2008, unless the application depends on the definition of a profile support indicator macro in **<unistd.h>** (that is, a profile shall not require restricted, altered, or extended behaviors of an implementation of POSIX.1-2008).
- Profiles are permitted to require the definition of a *profile support indicator macro* with a name beginning **\_POSIX\_AEP\_** in **<unistd.h>**.
- Profiles shall require the definition of the macro **\_POSIX\_SUBPROFILE** in **<unistd.h>** on implementations that do not meet all of the requirements of a POSIX.1-conforming implementation.
- Profiles are permitted to add additional requirements to the limits defined in **<limits.h>** and **<stdint.h>**, subject to the following:

For the limits in **<limits.h>** and **<stdint.h>**:

- 
3. As an example, the File System profiling option group provides underlying support for pathname resolution and file creation which are needed by any interface in POSIX.1-2008 that parses a *path* argument. If a profile requires support for the Device Input and Output profiling option group but does not require support for the File System profiling option group, the profile must specify how pathname resolution is to behave in that profile, how the **O\_CREAT** flag to *open()* is to be handled (and the use of the character 'a' in the *mode* argument of *fopen()* when a pathname argument names a file that does not exist), and specify lots of other details.
  4. As an example, POSIX.1-2008 requires that implementations claiming to support the Range Memory Locking option also support the Process Memory Locking option. A profile could require that the Range Memory Locking option had to be supplied without requiring that the Process Memory Locking option be supplied as long as the profile specifies everything an application developer or system implementor would have to know to build an application or implementation conforming to the profile.
  5. Note that the profile could just specify that any use of the features not specified by the profile would produce undefined or unspecified results.

- If the limit is specified as having a fixed value, it shall not be changed by a profile.
- If a limit is specified as having a minimum or maximum acceptable value, it may be changed by a profile as follows:
  - A profile may increase a minimum acceptable value, but shall not make a minimum acceptable value smaller.
  - A profile may reduce a maximum acceptable value, but shall not make a maximum acceptable value larger.
- A profile shall not change a limit specified as having a minimum or maximum value into a limit specified as having a fixed value.
- A profile shall not create new limits.
- Any implementation that conforms to POSIX.1-2008 (including all options and extended limits required by the profile) shall also conform to that profile, except for the possible omission from `<unistd.h>` of a profile support indicator macro required by the profile.

#### 2.1.5.2 XSI Option Groups

XSI This section describes Option Groups to support the definition of XSI conformance within the System Interfaces volume of POSIX.1-2008. The functionality described in this section shall be provided on implementations that support the XSI option and the appropriate Option Group (and the rest of this section is not further shaded).

The following Option Groups are defined.

##### Encryption

The Encryption Option Group is denoted by the symbolic constant `_XOPEN_CRYPT`. It includes the following functions:

`crypt()`, `encrypt()`, `setkey()`

These functions are marked CRYPT.

Due to export restrictions on the decoding algorithm in some countries, implementations may be restricted in making these functions available. All the functions in the Encryption Option Group may therefore return `[ENOSYS]` or, alternatively, `encrypt()` shall return `[ENOSYS]` for the decryption operation.

An implementation that claims conformance to this Option Group shall set `_XOPEN_CRYPT` to a value other than `-1`.

##### Realtime

The Realtime Option Group is denoted by the symbolic constant `_XOPEN_REALTIME`.

This Option Group includes a set of realtime functions drawn from options within POSIX.1-2008 (see [Section 2.1.6](#), on page 26).

Where entire functions are included in the Option Group, the NAME section is marked with REALTIME. Where additional semantics have been added to existing pages, the new material is identified by use of the appropriate margin legend for the underlying option defined within POSIX.1-2008.

An implementation that claims conformance to this Option Group shall set `_XOPEN_REALTIME` to a value other than `-1`.

This Option Group consists of the set of the following options from within POSIX.1-2008 (see [Section 2.1.6](#), on page 26):

```
_POSIX_FSYNC
_POSIX_MEMLOCK
_POSIX_MEMLOCK_RANGE
_POSIX_MESSAGE_PASSING
_POSIX_PRIORITIZED_IO
_POSIX_PRIORITY_SCHEDULING
_POSIX_SHARED_MEMORY_OBJECTS
_POSIX_SYNCHRONIZED_IO
```

If the symbolic constant `_XOPEN_REALTIME` is defined to have a value other than `-1`, then the following symbolic constants shall be defined by the implementation to have the value 200809L:

```
_POSIX_MEMLOCK
_POSIX_MEMLOCK_RANGE
_POSIX_MESSAGE_PASSING
_POSIX_PRIORITY_SCHEDULING
_POSIX_SHARED_MEMORY_OBJECTS
_POSIX_SYNCHRONIZED_IO
```

The functionality associated with `_POSIX_FSYNC` shall always be supported on XSI-conformant systems.

Support of `_POSIX_PRIORITIZED_IO` on XSI-conformant systems is optional. If `_POSIX_PRIORITIZED_IO` is supported, then asynchronous I/O operations performed by `aio_read()`, `aio_write()`, and `lio_listio()` shall be submitted at a priority equal to the scheduling priority equal to a base scheduling priority minus `aiocbp->aio_reqprio`. If Thread Execution Scheduling is not supported, then the base scheduling priority is that of the calling process; otherwise, the base scheduling priority is that of the calling thread. The implementation shall also document for which files I/O prioritization is supported.

### Advanced Realtime

An implementation that claims conformance to this Option Group shall also support the Realtime Option Group.

Where entire functions are included in the Option Group, the NAME section is marked with **ADVANCED REALTIME**. Where additional semantics have been added to existing pages, the new material is identified by use of the appropriate margin legend for the underlying option defined within POSIX.1-2008.

This Option Group consists of the set of the following options from within POSIX.1-2008 (see [Section 2.1.6](#), on page 26):

```
_POSIX_ADVISORY_INFO
_POSIX_CPUTIME
_POSIX_MONOTONIC_CLOCK
_POSIX_SPAWN
_POSIX_SPORADIC_SERVER
_POSIX_TYPED_MEMORY_OBJECTS
```

If the implementation supports the Advanced Realtime Option Group, then the following symbolic constants shall be defined by the implementation to have the value 200809L:

824        \_POSIX\_ADVISORY\_INFO  
 825        \_POSIX\_CPUTIME  
 826        \_POSIX\_MONOTONIC\_CLOCK  
 827        \_POSIX\_SPAWN  
 828        \_POSIX\_SPORADIC\_SERVER  
 829        \_POSIX\_TYPED\_MEMORY\_OBJECTS

830        If the symbolic constant \_POSIX\_SPORADIC\_SERVER is defined, then the symbolic constant  
 831        \_POSIX\_PRIORITY\_SCHEDULING shall also be defined by the implementation to have the  
 832        value 200809L.

### 833        **Realtime Threads**

834        The Realtime Threads Option Group is denoted by the symbolic constant  
 835        \_XOPEN\_REALTIME\_THREADS.

836        This Option Group consists of the set of the following options from within POSIX.1-2008 (see  
 837        [Section 2.1.6](#), on page 26):

838        \_POSIX\_THREAD\_PRIO\_INHERIT  
 839        \_POSIX\_THREAD\_PRIO\_PROTECT  
 840        \_POSIX\_THREAD\_PRIORITY\_SCHEDULING  
 841        \_POSIX\_THREAD\_ROBUST\_PRIO\_INHERIT  
 842        \_POSIX\_THREAD\_ROBUST\_PRIO\_PROTECT

843        Where applicable, whole pages are marked REALTIME THREADS, together with the  
 844        appropriate option margin legend for the SYNOPSIS section (see [Section 1.7.1](#), on page 7).

845        An implementation that claims conformance to this Option Group shall set  
 846        \_XOPEN\_REALTIME\_THREADS to a value other than -1.

847        If the symbol \_XOPEN\_REALTIME\_THREADS is defined to have a value other than -1, then the  
 848        following options shall also be defined by the implementation to have the value 200809L:

849        \_POSIX\_THREAD\_PRIO\_INHERIT  
 850        \_POSIX\_THREAD\_PRIO\_PROTECT  
 851        \_POSIX\_THREAD\_PRIORITY\_SCHEDULING  
 852        \_POSIX\_THREAD\_ROBUST\_PRIO\_INHERIT  
 853        \_POSIX\_THREAD\_ROBUST\_PRIO\_PROTECT

### 854        **Advanced Realtime Threads**

855        An implementation that claims conformance to this Option Group shall also support the  
 856        Realtime Threads Option Group.

857        Where entire functions are included in the Option Group, the NAME section is marked with  
 858        ADVANCED REALTIME THREADS. Where additional semantics have been added to existing  
 859        pages, the new material is identified by use of the appropriate margin legend for the underlying  
 860        option defined within POSIX.1-2008.

861        This Option Group consists of the set of the following options from within POSIX.1-2008 (see  
 862        [Section 2.1.6](#), on page 26):

863        \_POSIX\_THREAD\_CPUTIME  
 864        \_POSIX\_THREAD\_SPORADIC\_SERVER

865        If the symbolic constant \_POSIX\_THREAD\_SPORADIC\_SERVER is defined to have the value

200809L, then the symbolic constant `_POSIX_THREAD_PRIORITY_SCHEDULING` shall also be defined by the implementation to have the value 200809L.

If the implementation supports the Advanced Realtime Threads Option Group, then the following symbolic constants shall be defined by the implementation to have the value 200809L:

```
_POSIX_THREAD_CPUTIME
_POSIX_THREAD_SPORADIC_SERVER
```

## Tracing

This Option Group includes a set of tracing functions drawn from options within POSIX.1-2008 (see [Section 2.1.6](#), on page 26).

Where entire functions are included in the Option Group, the NAME section is marked with TRACING. Where additional semantics have been added to existing pages, the new material is identified by use of the appropriate margin legend for the underlying option defined within POSIX.1-2008.

This Option Group consists of the set of the following options from within POSIX.1-2008 (see [Section 2.1.6](#), on page 26):

```
_POSIX_TRACE
_POSIX_TRACE_EVENT_FILTER
_POSIX_TRACE_LOG
_POSIX_TRACE_INHERIT
```

If the implementation supports the Tracing Option Group, then the following symbolic constants shall be defined by the implementation to have the value 200809L:

```
_POSIX_TRACE
_POSIX_TRACE_EVENT_FILTER
_POSIX_TRACE_LOG
_POSIX_TRACE_INHERIT
```

## XSI STREAMS

OB XSR This section describes the XSI STREAMS Option Group, denoted by the symbolic constant `_XOPEN_STREAMS`. The functionality described in this section shall be provided on implementations that support the XSI STREAMS option (and the rest of this section is not further shaded).

This Option Group includes functionality related to STREAMS, a uniform mechanism for implementing networking services and other character-based I/O as described in XSH [Section 2.6](#) (on page 500).

It includes the following functions:

```
fattach()      ioctl()
fdetach()      isastream()
getmsg()       putmsg()
getpmsg()      putpmsg()
```

and the `<stropts.h>` header.

Where applicable, whole pages are marked STREAMS, together with the appropriate option margin legend for the SYNOPSIS section (see [Section 1.7.1](#), on page 7). Where additional semantics have been added to existing pages, the new material is identified by use of the

appropriate margin legend for the underlying option defined within POSIX.1-2008.

An implementation that claims conformance to this Option Group shall set `_XOPEN_STREAMS` to a value other than `-1`.

## 2.1.6 Options

The symbolic constants defined in `<unistd.h>`, [Constants for Options and Option Groups](#) (on page 433) reflect implementation options for POSIX.1-2008. These symbols can be used by the application to determine which of three categories of support for optional facilities are provided by the implementation.

### 1. Option not supported for compilation.

The implementation advertises at compile time (by defining the constant in `<unistd.h>` with value `-1`, or by leaving it undefined) that the option is not supported for compilation and, at the time of compilation, is not supported for runtime use. In this case, the headers, data types, function interfaces, and utilities required only for the option need not be present. A later runtime check using the `fpathconf()`, `pathconf()`, or `sysconf` functions defined in the System Interfaces volume of POSIX.1-2008 or the `getconf` utility defined in the Shell and Utilities volume of POSIX.1-2008 can in some circumstances indicate that the option is supported at runtime. (For example, an old application binary might be run on a newer implementation to which support for the option has been added.)

### 2. Option always supported.

The implementation advertises at compile time (by defining the constant in `<unistd.h>` with a value greater than zero) that the option is supported both for compilation and for use at runtime. In this case, all headers, data types, function interfaces, and utilities required only for the option shall be available and shall operate as specified. Runtime checks with `fpathconf()`, `pathconf()`, or `sysconf` shall indicate that the option is supported.

### 3. Option might or might not be supported at runtime.

The implementation advertises at compile time (by defining the constant in `<unistd.h>` with value zero) that the option is supported for compilation and might or might not be supported at runtime. In this case, the `fpathconf()`, `pathconf()`, or `sysconf()` functions defined in the System Interfaces volume of POSIX.1-2008 or the `getconf` utility defined in the Shell and Utilities volume of POSIX.1-2008 can be used to retrieve the value of each symbol on each specific implementation to determine whether the option is supported at runtime. All headers, data types, and function interfaces required to compile and execute applications which use the option at runtime (after checking at runtime that the option is supported) shall be provided, but if the option is not supported at runtime they need not operate as specified. Utilities or other facilities required only for the option, but not needed to compile and execute such applications, need not be present.

If an option is not supported for compilation, an application that attempts to use anything associated only with the option is considered to be requiring an extension. Unless explicitly specified otherwise, the behavior of functions associated with an option that is not supported at runtime is unspecified, and an application that uses such functions without first checking `fpathconf()`, `pathconf()`, or `sysconf` is considered to be requiring an extension.

Margin codes are defined for each option (see [Section 1.7.1](#), on page 7).



## 950 2.1.6.1 System Interfaces

951 Refer to **<unistd.h>**, **Constants for Options and Option Groups** (on page 433) for the list of  
 952 options.

## 953 2.1.6.2 Shell and Utilities

954 Each of these symbols shall be considered valid names by the implementation. Refer to  
 955 **<unistd.h>**, **Constants for Options and Option Groups** (on page 433).

956 The literal names shown below apply only to the *getconf* utility.

## 957 CD POSIX2\_C\_DEV

958 The system supports the C-Language Development Utilities option.

959 The utilities in the C-Language Development Utilities option are used for the development  
 960 of C-language applications, including compilation or translation of C source code and  
 961 complex program generators for simple lexical tasks and processing of context-free  
 962 grammars.

963 The utilities listed below may be provided by a conforming system; however, any system  
 964 claiming conformance to the C-Language Development Utilities option shall provide all of  
 965 the utilities listed.

966 *c99*  
 967 *lex*  
 968 *yacc*

## 969 POSIX2\_CHAR\_TERM

970 The system supports the Terminal Characteristics option. This value need not be present on  
 971 a system not supporting the User Portability Utilities option.

972 Where applicable, the dependency is noted within the description of the utility.

973 This option applies only to systems supporting the User Portability Utilities option. If  
 974 supported, then the system supports at least one terminal type capable of all operations  
 975 described in POSIX.1-2008; see **Section 10.2** (on page 198).

## 976 FD POSIX2\_FORT\_DEV

977 The system supports the FORTRAN Development Utilities option.

978 The *fort77* FORTRAN compiler is the only utility in the FORTRAN Development Utilities  
 979 option. This is used for the development of FORTRAN language applications, including  
 980 compilation or translation of FORTRAN source code.

981 The *fort77* utility may be provided by a conforming system; however, any system claiming  
 982 conformance to the FORTRAN Development Utilities option shall provide the *fort77* utility.

## 983 FR POSIX2\_FORT\_RUN

984 The system supports the FORTRAN Runtime Utilities option.

985 The *asa* utility is the only utility in the FORTRAN Runtime Utilities option.

986 The *asa* utility may be provided by a conforming system; however, any system claiming  
 987 conformance to the FORTRAN Runtime Utilities option shall provide the *asa* utility.

## 988 POSIX2\_LOCALEDEF

989 The system supports the Locale Creation Utilities option.

990 If supported, the system supports the creation of locales as described in the *localedef* utility.

991 The *localedef* utility may be provided by a conforming system; however, any system  
 992 claiming conformance to the Locale Creation Utilities option shall provide the *localedef*  
 993 utility.

#### 994 OB BE POSIX2\_PBS

995 The system supports the Batch Environment Services and Utilities option (see XCU [Chapter](#)  
 996 [3](#), on page 2417).

997 **Note:** The Batch Environment Services and Utilities option is a combination of mandatory and  
 998 optional batch services and utilities. The POSIX2\_PBS symbolic constant implies the system  
 999 supports all the mandatory batch services and utilities.

#### 1000 POSIX2\_PBS\_ACCOUNTING

1001 The system supports the Batch Accounting option.

#### 1002 POSIX2\_PBS\_CHECKPOINT

1003 The system supports the Batch Checkpoint/Restart option.

#### 1004 POSIX2\_PBS\_LOCATE

1005 The system supports the Locate Batch Job Request option.

#### 1006 POSIX2\_PBS\_MESSAGE

1007 The system supports the Batch Job Message Request option.

#### 1008 POSIX2\_PBS\_TRACK

1009 The system supports the Track Batch Job Request option.

#### 1010 SD POSIX2\_SW\_DEV

1011 The system supports the Software Development Utilities option.

1012 The utilities in the Software Development Utilities option are used for the development of  
 1013 applications, including compilation or translation of source code, the creation and  
 1014 maintenance of library archives, and the maintenance of groups of inter-dependent  
 1015 programs.

1016 The utilities listed below may be provided by the conforming system; however, any system  
 1017 claiming conformance to the Software Development Utilities option shall provide all of the  
 1018 utilities listed here.

1019 *ar*  
 1020 *make*  
 1021 *nm*  
 1022 *strip*

#### 1023 UP POSIX2\_UPE

1024 The system supports the User Portability Utilities option.

1025 The utilities in the User Portability Utilities option shall be implemented on all systems that  
 1026 claim conformance to this option, except for the *vi* utility which is noted as having features  
 1027 that cannot be implemented on all terminal types; if the POSIX2\_CHAR\_TERM option is  
 1028 supported, the system shall support all such features on at least one terminal type; see  
 1029 [Section 10.2](#) (on page 198).

1030 The list of utilities in the User Portability Utilities option is as follows:

1031 *bg* *fc* *jobs* *talk*  
 1032 *ex* *fg* *more* *vi*



1033	XSI	XOPEN_UNIX
1034		The system supports the X/Open System Interfaces (XSI) option (see <a href="#">Section 2.1.4</a> , on page
1035		19).
1036	UU	XOPEN_UUCP
1037		The system supports the UUCP Utilities option.
1038		The list of utilities in the UUCP Utilities option is as follows:
1039		<i>uucp</i>
1040		<i>uustat</i>
1041		<i>uux</i>

## 1042 2.2 Application Conformance

1043 For the purposes of POSIX.1-2008, the application conformance requirements given in this  
1044 section apply.

1045 All applications claiming conformance to POSIX.1-2008 shall use only language-dependent  
1046 services for the C programming language described in [Section 2.3](#) (on page 31), shall use only  
1047 the utilities and facilities defined in the Shell and Utilities volume of POSIX.1-2008, and shall fall  
1048 within one of the following categories.

### 1049 2.2.1 Strictly Conforming POSIX Application

1050 A Strictly Conforming POSIX Application is an application that requires only the facilities  
1051 described in POSIX.1-2008. Such an application:

- 1052 1. Shall accept any implementation behavior that results from actions it takes in areas  
1053 described in POSIX.1-2008 as *implementation-defined* or *unspecified*, or where POSIX.1-2008  
1054 indicates that implementations may vary
- 1055 2. Shall not perform any actions that are described as producing *undefined* results
- 1056 3. For symbolic constants, shall accept any value in the range permitted by POSIX.1-2008,  
1057 but shall not rely on any value in the range being greater than the minimums listed or  
1058 being less than the maximums listed in POSIX.1-2008
- 1059 4. Shall not use facilities designated as *obsolescent*
- 1060 5. Is required to tolerate and permitted to adapt to the presence or absence of optional  
1061 facilities whose availability is indicated by [Section 2.1.3](#) (on page 17)
- 1062 6. For the C programming language, shall not produce any output dependent on any  
1063 behavior described in the ISO/IEC 9899:1999 standard as *unspecified*, *undefined*, or  
1064 *implementation-defined*, unless the System Interfaces volume of POSIX.1-2008 specifies the  
1065 behavior
- 1066 7. For the C programming language, shall not exceed any minimum implementation limit  
1067 defined in the ISO/IEC 9899:1999 standard, unless the System Interfaces volume of  
1068 POSIX.1-2008 specifies a higher minimum implementation limit
- 1069 8. For the C programming language, shall define `_POSIX_C_SOURCE` to be 200809L before  
1070 any header is included

1071 Within POSIX.1-2008, any restrictions placed upon a Conforming POSIX Application shall

restrict a Strictly Conforming POSIX Application.

## 2.2.2 Conforming POSIX Application

### 2.2.2.1 ISO/IEC Conforming POSIX Application

An ISO/IEC Conforming POSIX Application is an application that uses only the facilities described in POSIX.1-2008 and approved Conforming Language bindings for any ISO or IEC standard. Such an application shall include a statement of conformance that documents all options and limit dependencies, and all other ISO or IEC standards used.

### 2.2.2.2 <National Body> Conforming POSIX Application

A <National Body> Conforming POSIX Application differs from an ISO/IEC Conforming POSIX Application in that it also may use specific standards of a single ISO/IEC member body referred to here as <National Body>. Such an application shall include a statement of conformance that documents all options and limit dependencies, and all other <National Body> standards used.

## 2.2.3 Conforming POSIX Application Using Extensions

A Conforming POSIX Application Using Extensions is an application that differs from a Conforming POSIX Application only in that it uses non-standard facilities that are consistent with POSIX.1-2008. Such an application shall fully document its requirements for these extended facilities, in addition to the documentation required of a Conforming POSIX Application. A Conforming POSIX Application Using Extensions shall be either an ISO/IEC Conforming POSIX Application Using Extensions or a <National Body> Conforming POSIX Application Using Extensions (see [Section 2.2.2.1](#) and [Section 2.2.2.2](#)).

## 2.2.4 Strictly Conforming XSI Application

A Strictly Conforming XSI Application is an application that requires only the facilities described in POSIX.1-2008. Such an application:

1. Shall accept any implementation behavior that results from actions it takes in areas described in POSIX.1-2008 as *implementation-defined* or *unspecified*, or where POSIX.1-2008 indicates that implementations may vary
2. Shall not perform any actions that are described as producing *undefined* results
3. For symbolic constants, shall accept any value in the range permitted by POSIX.1-2008, but shall not rely on any value in the range being greater than the minimums listed or being less than the maximums listed in POSIX.1-2008
4. Shall not use facilities designated as *obsolescent*
5. Is required to tolerate and permitted to adapt to the presence or absence of optional facilities whose availability is indicated by [Section 2.1.4](#) (on page 19)

6. For the C programming language, shall not produce any output dependent on any behavior described in the ISO C standard as *unspecified*, *undefined*, or *implementation-defined*, unless the System Interfaces volume of POSIX.1-2008 specifies the behavior
7. For the C programming language, shall not exceed any minimum implementation limit defined in the ISO C standard, unless the System Interfaces volume of POSIX.1-2008 specifies a higher minimum implementation limit
8. For the C programming language, shall define `_XOPEN_SOURCE` to be 700 before any header is included

Within POSIX.1-2008, any restrictions placed upon a Conforming POSIX Application shall restrict a Strictly Conforming XSI Application.

### 2.2.5 Conforming XSI Application Using Extensions

A Conforming XSI Application Using Extensions is an application that differs from a Strictly Conforming XSI Application only in that it uses non-standard facilities that are consistent with POSIX.1-2008. Such an application shall fully document its requirements for these extended facilities, in addition to the documentation required of a Strictly Conforming XSI Application.

## 2.3 Language-Dependent Services for the C Programming Language

Implementors seeking to claim conformance using the ISO C standard shall claim POSIX conformance as described in [Section 2.1.3](#) (on page 17).

## 2.4 Other Language-Related Specifications

POSIX.1-2008 is currently specified in terms of the shell command language and ISO C. Bindings to other programming languages are being developed.

If conformance to POSIX.1-2008 is claimed for implementation of any programming language, the implementation of that language shall support the use of external symbols distinct to at least 31 bytes in length in the source program text. (That is, identifiers that differ at or before the thirty-first byte shall be distinct.) If a national or international standard governing a language defines a maximum length that is less than this value, the language-defined maximum shall be supported. External symbols that differ only by case shall be distinct when the character set in use distinguishes uppercase and lowercase characters and the language permits (or requires) uppercase and lowercase characters to be distinct in external symbols.



Chapter 3

## Definitions

For the purposes of POSIX.1-2008, the following terms and definitions apply. The Authoritative Dictionary of IEEE Standards Terms, Seventh Edition should be referenced for terms not defined in this section.

**Note:** No shading to denote extensions or options occurs in this chapter. Where the terms and definitions given in this chapter are used elsewhere in text related to extensions and options, they are shaded as appropriate.

### 3.1 Abortive Release

An abrupt termination of a network connection that may result in the loss of data.

### 3.2 Absolute Pathname

A pathname beginning with a single or more than two <slash> characters; see also [Section 3.271](#) (on page 76).

**Note:** Pathname Resolution is defined in detail in [Section 4.13](#) (on page 111).

### 3.3 Access Mode

A particular form of access permitted to a file.

### 3.4 Additional File Access Control Mechanism

An implementation-defined mechanism that is layered upon the access control mechanisms defined here, but which do not grant permissions beyond those defined herein, although they may further restrict them.

**Note:** File Access Permissions are defined in detail in [Section 4.5](#) (on page 108).

### 3.5 Address Space

The memory locations that can be referenced by a process or the threads of a process.

### 3.6 Advisory Information

An interface that advises the implementation on (portable) application behavior so that it can optimize the system.

### 3.7 Affirmative Response

An input string that matches one of the responses acceptable to the *LC\_MESSAGES* category keyword **yesexpr**, matching an extended regular expression in the current locale.

**Note:** The *LC\_MESSAGES* category is defined in detail in [Section 7.3.6](#) (on page 165).

### 3.8 Alert

To cause the user's terminal to give some audible or visual indication that an error or some other event has occurred. When the standard output is directed to a terminal device, the method for alerting the terminal user is unspecified. When the standard output is not directed to a terminal device, the alert is accomplished by writing the alert to standard output (unless the utility description indicates that the use of standard output produces undefined results in this case).

### 3.9 Alert Character (<alert>)

A character that in the output stream should cause a terminal to alert its user via a visual or audible notification. It is the character designated by '`\a`' in the C language. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the alert function.

### 3.10 Alias Name

In the shell command language, a word consisting solely of underscores, digits, and alphabetic characters from the portable character set and any of the following characters: '`!`', '`%`', '`'`', '`,`', '`@`'.

Implementations may allow other characters within alias names as an extension.

**Note:** The Portable Character Set is defined in detail in [Section 6.1](#) (on page 125).

### 3.11 Alignment

A requirement that objects of a particular type be located on storage boundaries with addresses that are particular multiples of a byte address.

**Note:** See also the ISO C standard, Section B3.

### 3.12 Alternate File Access Control Mechanism

An implementation-defined mechanism that is independent of the access control mechanisms defined herein, and which if enabled on a file may either restrict or extend the permissions of a given user. POSIX.1-2008 defines when such mechanisms can be enabled and when they are disabled.

**Note:** File Access Permissions are defined in detail in [Section 4.5](#) (on page 108).

### 3.13 Alternate Signal Stack

Memory associated with a thread, established upon request by the implementation for a thread, separate from the thread signal stack, in which signal handlers responding to signals sent to that thread may be executed.

### 3.14 Ancillary Data

Protocol-specific, local system-specific, or optional information. The information can be both local or end-to-end significant, header information, part of a data portion, protocol-specific, and implementation or system-specific.

### 3.15 Angle Brackets

The characters '`<`' (left-angle-bracket) and '`>`' (right-angle-bracket). When used in the phrase "enclosed in angle brackets", the symbol '`<`' immediately precedes the object to be enclosed, and '`>`' immediately follows it. When describing these characters in the portable character set, the names `<less-than-sign>` and `<greater-than-sign>` are used.

### 3.16 Apostrophe Character (<apostrophe>)

The character designated by '`\'`' in the C language, also known as the single-quote character.

### 3.17 Application

A computer program that performs some desired function.

1208 When the User Portability Utilities option is supported, requirements placed on applications +  
 1209 relating to the use of standard utilities shall also apply to the actions of a user who is entering +  
 1210 shell command language statements into an interactive shell.

### 1211 3.18 Application Address

1212 Endpoint address of a specific application.

### 1213 3.19 Application Program Interface (API)

1214 The definition of syntax and semantics for providing computer system services.

### 1215 3.20 Appropriate Privileges

1216 An implementation-defined means of associating privileges with a process with regard to the  
 1217 function calls, function call options, and the commands that need special privileges. There may  
 1218 be zero or more such means. These means (or lack thereof) are described in the conformance  
 1219 document.

1220 **Note:** Function calls are defined in the System Interfaces volume of POSIX.1-2008, and commands are  
 1221 defined in the Shell and Utilities volume of POSIX.1-2008.

### 1222 3.21 Argument

1223 In the shell command language, a parameter passed to a utility as the equivalent of a single  
 1224 string in the *argv* array created by one of the *exec* functions. An argument is one of the options,  
 1225 option-arguments, or operands following the command name.

1226 **Note:** The Utility Argument Syntax is defined in detail in [Section 12.1](#) (on page 213) and XCU [Section](#)  
 1227 [2.9.1.1](#) (on page 2357).

1228 In the C language, an expression in a function call expression or a sequence of preprocessing  
 1229 tokens in a function-like macro invocation.

### 1230 3.22 Arm (a Timer)

1231 To start a timer measuring the passage of time, enabling notifying a process when the specified  
 1232 time or time interval has passed.

### 1233 3.23 Asterisk Character (<asterisk>)

1234 The character ' \* '.



### 3.24 Async-Cancel-Safe Function

A function that may be safely invoked by an application while the asynchronous form of cancellation is enabled. No function is async-cancel-safe unless explicitly described as such.

### 3.25 Asynchronous Events

Events that occur independently of the execution of the application.

### 3.26 Asynchronous Input and Output

A functionality enhancement to allow an application process to queue data input and output commands with asynchronous notification of completion.

### 3.27 Async-Signal-Safe Function

A function that can be called, without restriction, from signal-catching functions. Note that, although there is no restriction on the calls themselves, for certain functions there are restrictions on subsequent behavior after the function is called from a signal-catching function. No function is async-signal-safe unless explicitly described as such.

**Note:** Async-signal-safety is defined in detail in XSH [Section 2.4.3](#) (on page 490).

### 3.28 Asynchronously-Generated Signal

A signal that is not attributable to a specific thread. Examples are signals sent via *kill()*, signals sent from the keyboard, and signals delivered to process groups. Being asynchronous is a property of how the signal was generated and not a property of the signal number. All signals may be generated asynchronously.

**Note:** The *kill()* function is defined in detail in the System Interfaces volume of POSIX.1-2008.

### 3.29 Asynchronous I/O Completion

For an asynchronous read or write operation, when a corresponding synchronous read or write would have completed and when any associated status fields have been updated.

### 3.30 Asynchronous I/O Operation

An I/O operation that does not of itself cause the thread requesting the I/O to be blocked from further use of the processor.

1261 This implies that the process and the I/O operation may be running concurrently.

### 1262 **3.31 Authentication**

1263 The process of validating a user or process to verify that the user or process is not a counterfeit.

### 1264 **3.32 Authorization**

1265 The process of verifying that a user or process has permission to use a resource in the manner  
1266 requested.

1267 To ensure security, the user or process would also need to be authenticated before granting  
1268 access.

### 1269 **3.33 Background Job**

1270 See *Background Process Group* in [Section 3.35](#).

### 1271 **3.34 Background Process**

1272 A process that is a member of a background process group.

### 1273 **3.35 Background Process Group (or Background Job)**

1274 Any process group, other than a foreground process group, that is a member of a session that  
1275 has established a connection with a controlling terminal.

### 1276 **3.36 Backquote Character**

1277 The character ' ` ', also known as <grave-accent>.

### 1278 **3.37 Backslash Character (<backslash>)**

1279 The character designated by ' \\ ' in the C language, also known as reverse solidus.

### 3.38 Backspace Character (<backspace>)

A character that, in the output stream, should cause printing (or displaying) to occur one column position previous to the position about to be printed. If the position about to be printed is at the beginning of the current line, the behavior is unspecified. It is the character designated by '`\b`' in the C language. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the backspace function. The backspace defined here is not necessarily the ERASE special character.

**Note:** Special Characters are defined in detail in [Section 11.1.9](#) (on page 204).

### 3.39 Barrier

A synchronization object that allows multiple threads to synchronize at a particular point in their execution.

### 3.40 Basename

For pathnames containing at least one filename: the final, or only, filename in the pathname. For pathnames consisting only of <slash> characters: either '`/`' or '`/"`' if the pathname consists of exactly two <slash> characters, and '`/`' otherwise.

### 3.41 Basic Regular Expression (BRE)

A regular expression (see [Section 3.321](#), on page 85) used by the majority of utilities that select strings from a set of character strings.

**Note:** Basic Regular Expressions are described in detail in [Section 9.3](#) (on page 183).

### 3.42 Batch Access List

A list of user IDs and group IDs of those users and groups authorized to place batch jobs in a batch queue.

A batch access list is associated with a batch queue. A batch server uses the batch access list of a batch queue as one of the criteria in deciding to put a batch job in a batch queue.

### 3.43 Batch Administrator

A user that is authorized to modify all the attributes of queues and jobs and to change the status of a batch server.

### 3.44 Batch Client

A computational entity that utilizes batch services by making requests of batch servers.

Batch clients often provide the means by which users access batch services, although a batch server may act as a batch client by virtue of making requests of another batch server.

### 3.45 Batch Destination

The batch server in a batch system to which a batch job should be sent for processing.

Acceptance of a batch job at a batch destination is the responsibility of a receiving batch server. A batch destination may consist of a batch server-specific portion, a network-wide portion, or both. The batch server-specific portion is referred to as the “batch queue”. The network-wide portion is referred to as a “batch server name”.

### 3.46 Batch Destination Identifier

A string that identifies a specific batch destination.

A string of characters in the portable character set used to specify a particular batch destination.

**Note:** The Portable Character Set is defined in detail in [Section 6.1](#) (on page 125).

### 3.47 Batch Directive

A line from a file that is interpreted by the batch server. The line is usually in the form of a comment and is an additional means of passing options to the *qsub* utility.

**Note:** The *qsub* utility is defined in detail in the Shell and Utilities volume of POSIX.1-2008.

### 3.48 Batch Job

A set of computational tasks for a computing system.

Batch jobs are managed by batch servers.

Once created, a batch job may be executing or pending execution. A batch job that is executing has an associated session leader (a process) that initiates and monitors the computational tasks of the batch job.

### 3.49 Batch Job Attribute

A named data type whose value affects the processing of a batch job.

The values of the attributes of a batch job affect the processing of that job by the batch server that manages the batch job.

### 3.50 Batch Job Identifier

A unique name for a batch job. A name that is unique among all other batch job identifiers in a batch system and that identifies the batch server to which the batch job was originally submitted.

### 3.51 Batch Job Name

A label that is an attribute of a batch job. The batch job name is not necessarily unique.

### 3.52 Batch Job Owner

The *username@hostname* of the user submitting the batch job, where *username* is a user name (see also [Section 3.437](#), on page 103) and *hostname* is a network host name.

### 3.53 Batch Job Priority

A value specified by the user that may be used by an implementation to determine the order in which batch jobs are selected to be executed. Job priority has a numeric value in the range -1 024 to 1 023.

**Note:** The batch job priority is not the execution priority (nice value) of the batch job.

### 3.54 Batch Job State

An attribute of a batch job which determines the types of requests that the batch server that manages the batch job can accept for the batch job. Valid states include QUEUED, RUNNING, HELD, WAITING, EXITING, and TRANSITING.

### 3.55 Batch Name Service

A service that assigns batch names that are unique within the batch name space, and that can translate a unique batch name into the location of the named batch entity.

**3.56 Batch Name Space**

The environment within which a batch name is known to be unique.

**3.57 Batch Node**

A host containing part or all of a batch system.

A batch node is a host meeting at least one of the following conditions:

- Capable of executing a batch client
- Contains a routing batch queue
- Contains an execution batch queue

**3.58 Batch Operator**

A user that is authorized to modify some, but not all, of the attributes of jobs and queues, and may change the status of the batch server.

**3.59 Batch Queue**

A manageable object that represents a set of batch jobs and is managed by a single batch server.

**Note:** A set of batch jobs is called a batch queue largely for historical reasons. Jobs are selected from the batch queue for execution based on attributes such as priority, resource requirements, and hold conditions.

See also XCU [Section 3.1.2](#) (on page 2418).

**3.60 Batch Queue Attribute**

A named data type whose value affects the processing of all batch jobs that are members of the batch queue.

A batch queue has attributes that affect the processing of batch jobs that are members of the batch queue.

**3.61 Batch Queue Position**

The place, relative to other jobs in the batch queue, occupied by a particular job in a batch queue. This is defined in part by submission time and priority; see also [Section 3.62](#) (on page 43).

### 3.62 Batch Queue Priority

The maximum job priority allowed for any batch job in a given batch queue.

The batch queue priority is set and may be changed by users with appropriate privileges. The priority is bounded in an implementation-defined manner.

### 3.63 Batch Rerunability

An attribute of a batch job indicating that it may be rerun after an abnormal termination from the beginning without affecting the validity of the results.

### 3.64 Batch Restart

The action of resuming the processing of a batch job from the point of the last checkpoint. Typically, this is done if the batch job has been interrupted because of a system failure.

### 3.65 Batch Server

A computational entity that provides batch services.

### 3.66 Batch Server Name

A string of characters in the portable character set used to specify a particular server in a network.

**Note:** The Portable Character Set is defined in detail in [Section 6.1](#) (on page 125).

### 3.67 Batch Service

Computational and organizational services performed by a batch system on behalf of batch jobs.

Batch services are of two types: requested and deferred.

**Note:** Batch Services are listed in XCU [Table 3-5](#) (on page 2432).

### 3.68 Batch Service Request

A solicitation of services from a batch client to a batch server.

A batch service request may entail the exchange of any number of messages between the batch client and the batch server.

When naming specific types of service requests, the term “request” is qualified by the type of

1406 request, as in *Queue Batch Job Request* and *Delete Batch Job Request*.

### 1407 **3.69 Batch Submission**

1408 The process by which a batch client requests that a batch server create a batch job via a *Queue Job*  
1409 *Request* to perform a specified computational task.

### 1410 **3.70 Batch System**

1411 A collection of one or more batch servers.

### 1412 **3.71 Batch Target User**

1413 The name of a user on the batch destination batch server.

1414 The target user is the user name under whose account the batch job is to execute on the  
1415 destination batch server.

### 1416 **3.72 Batch User**

1417 A user who is authorized to make use of batch services.

### 1418 **3.73 Bind**

1419 The process of assigning a network address to an endpoint.

### 1420 **3.74 Blank Character (<blank>)**

1421 One of the characters that belong to the **blank** character class as defined via the *LC\_CTYPE*  
1422 category in the current locale. In the POSIX locale, a <blank> character is either a <tab> or a  
1423 <space>.

### 1424 **3.75 Blank Line**

1425 A line consisting solely of zero or more <blank> characters terminated by a <newline>; see also  
1426 [Section 3.145](#) (on page 56).



### 3.76 Blocked Process (or Thread)

A process (or thread) that is waiting for some condition (other than the availability of a processor) to be satisfied before it can continue execution.

### 3.77 Blocking

A property of an open file description that causes function calls associated with it to wait for the requested action to be performed before returning.

### 3.78 Block-Mode Terminal

A terminal device operating in a mode incapable of the character-at-a-time input and output operations described by some of the standard utilities.

**Note:** Output Devices and Terminal Types are defined in detail in [Section 10.2](#) (on page 198).

### 3.79 Block Special File

A file that refers to a device. A block special file is normally distinguished from a character special file by providing access to the device in a manner such that the hardware characteristics of the device are not visible.

### 3.80 Braces

The characters ' { ' (left-curly-bracket) and ' } ' (right-curly-bracket). When used in the phrase "enclosed in (curly) braces" the symbol ' { ' immediately precedes the object to be enclosed, and ' } ' immediately follows it. When describing these characters in the portable character set, the names <left-curly-bracket> and <left-brace> are used for ' { ', and <right-curly-bracket> and <right-brace> are used for ' } '.

### 3.81 Brackets

The characters '[' (left-square-bracket) and ']' (right-square-bracket). When used in the phrase “enclosed in (square) brackets” the symbol '[' immediately precedes the object to be enclosed, and ']' immediately follows it. When describing these characters in the portable character set, the names <left-square-bracket> and <right-square-bracket> are used.

### 3.82 Broadcast

The transfer of data from one endpoint to several endpoints, as described in RFC 919 and RFC 922.

### 3.83 Built-In Utility (or Built-In)

A utility implemented within a shell. The utilities referred to as special built-ins have special qualities. Unless qualified, the term “built-in” includes the special built-in utilities. Regular built-ins are not required to be actually built into the shell on the implementation, but they do have special command-search qualities.

**Note:** Special Built-In Utilities are defined in detail in XCU [Section 2.14](#) (on page 2374).

Regular Built-In Utilities are defined in detail in XCU [Section 2.9.1.1](#) (on page 2357).

### 3.84 Byte

An individually addressable unit of data storage that is exactly an octet, used to store a character or a portion of a character; see also [Section 3.87](#) (on page 47). A byte is composed of a contiguous sequence of 8 bits. The least significant bit is called the “low-order” bit; the most significant is called the “high-order” bit.

**Note:** The definition of byte from the ISO C standard is broader than the above and might accommodate hardware architectures with different sized addressable units than octets.

### 3.85 Byte Input/Output Functions

The functions that perform byte-oriented input from streams or byte-oriented output to streams: *fgetc()*, *fgets()*, *fprintf()*, *fputc()*, *fputs()*, *fread()*, *fscanf()*, *fwrite()*, *getc()*, *getchar()*, *getdelim()*, *getline()*, *gets()*, *printf()*, *putc()*, *putchar()*, *puts()*, *scanf()*, *ungetc()*, *vfprintf()*, and *vprintf()*.

**Note:** Functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

### 3.86 Carriage-Return Character (<carriage-return>)

A character that in the output stream indicates that printing should start at the beginning of the

1476 same physical line in which the carriage-return occurred. It is the character designated by '`\r`'  
 1477 in the C language. It is unspecified whether this character is the exact sequence transmitted to an  
 1478 output device by the system to accomplish the movement to the beginning of the line.

### 1479 3.87 Character

1480 A sequence of one or more bytes representing a single graphic symbol or control code.

1481 **Note:** This term corresponds to the ISO C standard term multi-byte character, where a single-byte  
 1482 character is a special case of a multi-byte character. Unlike the usage in the ISO C standard,  
 1483 *character* here has no necessary relationship with storage space, and *byte* is used when storage  
 1484 space is discussed.

1485 See the definition of the portable character set in [Section 6.1](#) (on page 125) for a further  
 1486 explanation of the graphical representations of (abstract) characters, as opposed to character  
 1487 encodings.

### 1488 3.88 Character Array

1489 An array of elements of type **char**.

### 1490 3.89 Character Class

1491 A named set of characters sharing an attribute associated with the name of the class. The classes  
 1492 and the characters that they contain are dependent on the value of the *LC\_CTYPE* category in  
 1493 the current locale.

1494 **Note:** The *LC\_CTYPE* category is defined in detail in [Section 7.3.1](#) (on page 139).

### 1495 3.90 Character Set

1496 A finite set of different characters used for the representation, organization, or control of data.

**3.91 Character Special File**

A file that refers to a device (such as a terminal device file) or that has special properties (such as `/dev/null`).

**Note:** The General Terminal Interface is defined in detail in [Chapter 11](#) (on page 199).

**3.92 Character String**

A contiguous sequence of characters terminated by and including the first null byte.

**3.93 Child Process**

A new process created (by `fork()`, `posix_spawn()`, or `posix_spawnp()`) by a given process. A child process remains the child of the creating process as long as both processes continue to exist.

**Note:** The `fork()`, `posix_spawn()`, and `posix_spawnp()` functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

**3.94 Circumflex Character (<circumflex>)**

The character `' ^ '`.

**3.95 Clock**

A software or hardware object that can be used to measure the apparent or actual passage of time.

The current value of the time measured by a clock can be queried and, possibly, set to a value within the legal range of the clock.

**3.96 Clock Jump**

The difference between two successive distinct values of a clock, as observed from the application via one of the “get time” operations.

**3.97 Clock Tick**

An interval of time; an implementation-defined number of these occur each second. Clock ticks are one of the units that may be used to express a value found in type `clock_t`.

### 3.98 Coded Character Set

A set of unambiguous rules that establishes a character set and the one-to-one relationship between each character of the set and its bit representation.

### 3.99 Codeset

The result of applying rules that map a numeric code value to each element of a character set. An element of a character set may be related to more than one numeric code value but the reverse is not true. However, for state-dependent encodings the relationship between numeric code values and elements of a character set may be further controlled by state information. The character set may contain fewer elements than the total number of possible numeric code values; that is, some code values may be unassigned.

**Note:** Character Encoding is defined in detail in [Section 6.2](#) (on page 128).

### 3.100 Collating Element

The smallest entity used to determine the logical ordering of character or wide-character strings; see also [Section 3.102](#). A collating element consists of either a single character, or two or more characters collating as a single entity. The value of the `LC_COLLATE` category in the current locale determines the current set of collating elements.

### 3.101 Collation

The logical ordering of character or wide-character strings according to defined precedence rules. These rules identify a collation sequence between the collating elements, and such additional rules that can be used to order strings consisting of multiple collating elements.

### 3.102 Collation Sequence

The relative order of collating elements as determined by the setting of the `LC_COLLATE` category in the current locale. The collation sequence is used for sorting and is determined from the collating weights assigned to each collating element. In the absence of weights, the collation sequence is the order in which collating elements are specified between **order\_start** and **order\_end** keywords in the `LC_COLLATE` category.

Multi-level sorting is accomplished by assigning elements one or more collation weights, up to the limit `{COLL_WEIGHTS_MAX}`. On each level, elements may be given the same weight (at the primary level, called an equivalence class; see also [Section 3.151](#), on page 57) or be omitted from the sequence. Strings that collate equally using the first assigned weight (primary ordering) are then compared using the next assigned weight (secondary ordering), and so on.

**Note:** `{COLL_WEIGHTS_MAX}` is defined in detail in [<limits.h>](#).

**3.103 Column Position**

A unit of horizontal measure related to characters in a line.

It is assumed that each character in a character set has an intrinsic column width independent of any output device. Each printable character in the portable character set has a column width of one. The standard utilities, when used as described in POSIX.1-2008, assume that all characters have integral column widths. The column width of a character is not necessarily related to the internal representation of the character (numbers of bits or bytes).

The column position of a character in a line is defined as one plus the sum of the column widths of the preceding characters in the line. Column positions are numbered starting from 1.

**3.104 Command**

A directive to the shell to perform a particular task.

**Note:** Shell Commands are defined in detail in XCU [Section 2.9](#) (on page 2355).

**3.105 Command Language Interpreter**

An interface that interprets sequences of text input as commands. It may operate on an input stream or it may interactively prompt and read commands from a terminal. It is possible for applications to invoke utilities through a number of interfaces, which are collectively considered to act as command interpreters. The most obvious of these are the *sh* utility and the *system()* function, although *popen()* and the various forms of *exec* may also be considered to behave as interpreters.

**Note:** The *sh* utility is defined in detail in the Shell and Utilities volume of POSIX.1-2008.

The *system()*, *popen()*, and *exec* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

**3.106 Composite Graphic Symbol**

A graphic symbol consisting of a combination of two or more other graphic symbols in a single character position, such as a diacritical mark and a base character.

**3.107 Condition Variable**

A synchronization object which allows a thread to suspend execution, repeatedly, until some associated predicate becomes true. A thread whose execution is suspended on a condition variable is said to be blocked on the condition variable.

### 3.108 Connected Socket

A connection-mode socket for which a connection has been established, or a connectionless-mode socket for which a peer address has been set. See also [Section 3.109](#), [Section 3.110](#), [Section 3.111](#), and [Section 3.356](#) (on page 91).

### 3.109 Connection

An association established between two or more endpoints for the transfer of data

### 3.110 Connection Mode

The transfer of data in the context of a connection; see also [Section 3.111](#).

### 3.111 Connectionless Mode

The transfer of data other than in the context of a connection; see also [Section 3.110](#) and [Section 3.124](#) (on page 53).

### 3.112 Control Character

A character, other than a graphic character, that affects the recording, processing, transmission, or interpretation of text.

### 3.113 Control Operator

In the shell command language, a token that performs a control function. It is one of the following symbols:

&    &&    (    )    ;    ;;    newline    |    ||

The end-of-input indicator used internally by the shell is also considered a control operator.

**Note:** Token Recognition is defined in detail in XCU [Section 2.3](#) (on page 2337).

### 3.114 Controlling Process

The session leader that established the connection to the controlling terminal. If the terminal subsequently ceases to be a controlling terminal for this session, the session leader ceases to be the controlling process.

### 3.115 Controlling Terminal

A terminal that is associated with a session. Each session may have at most one controlling terminal associated with it, and a controlling terminal is associated with exactly one session. Certain input sequences from the controlling terminal cause signals to be sent to all processes in the foreground process group associated with the controlling terminal.

**Note:** The General Terminal Interface is defined in detail in [Chapter 11](#) (on page 199).

### 3.116 Conversion Descriptor

A per-process unique value used to identify an open codeset conversion.

### 3.117 Core File

A file of unspecified format that may be generated when a process terminates abnormally.

### 3.118 CPU Time (Execution Time)

The time spent executing a process or thread, including the time spent executing system services on behalf of that process or thread. If the Threads option is supported, then the value of the CPU-time clock for a process is implementation-defined. With this definition the sum of all the execution times of all the threads in a process might not equal the process execution time, even in a single-threaded process, because implementations may differ in how they account for time during context switches or for other reasons.

### 3.119 CPU-Time Clock

A clock that measures the execution time of a particular process or thread.

### 3.120 CPU-Time Timer

A timer attached to a CPU-time clock.

### 3.121 Current Job

In the context of job control, the job that will be used as the default for the *fg* or *bg* utilities. There is at most one current job; see also [Section 3.204](#) (on page 66).



### 3.122 Current Working Directory

See *Working Directory* in [Section 3.447](#) (on page 105).

### 3.123 Cursor Position

The line and column position on the screen denoted by the terminal's cursor.

### 3.124 Datagram

A unit of data transferred from one endpoint to another in connectionless mode service.

### 3.125 Data Segment

Memory associated with a process, that can contain dynamically allocated data.

### 3.126 Deferred Batch Service

A service that is performed as a result of events that are asynchronous with respect to requests.

**Note:** Once a batch job has been created, it is subject to deferred services.

### 3.127 Device

A computer peripheral or an object that appears to the application as such.

### 3.128 Device ID

A non-negative integer used to identify a device.

### 3.129 Directory

A file that contains directory entries. No two directory entries in the same directory have the same name.

### 3.130 Directory Entry (or Link)

An object that associates a filename with a file. Several directory entries can associate names

1650 with the same file.

### 1651 **3.131 Directory Stream**

1652 A sequence of all the directory entries in a particular directory. An open directory stream may be  
1653 implemented using a file descriptor.

### 1654 **3.132 Disarm (a Timer)**

1655 To stop a timer from measuring the passage of time, disabling any future process notifications  
1656 (until the timer is armed again).

### 1657 **3.133 Display**

1658 To output to the user's terminal. If the output is not directed to a terminal, the results are  
1659 undefined.

### 1660 **3.134 Display Line**

1661 A line of text on a physical device or an emulation thereof. Such a line will have a maximum  
1662 number of characters which can be presented.

1663 **Note:** This may also be written as "line on the display".

### 1664 **3.135 Dollar-Sign Character (<dollar-sign>)**

1665 The character ' \$ '.

### 1666 **3.136 Dot**

1667 In the context of naming files, the filename consisting of a single <period> character ( ' . ' ).

1668 **Note:** In the context of shell special built-in utilities, see *dot* in XCU [Section 2.14](#) (on page 2374).

1669 Pathname Resolution is defined in detail in [Section 4.13](#) (on page 111).

### 3.137 Dot-Dot

The filename consisting solely of two <period> characters (" . . ").

**Note:** Pathname Resolution is defined in detail in [Section 4.13](#) (on page 111).

### 3.138 Double-Quote Character

The character ' " ', also known as <quotation-mark>.

**Note:** The “double” adjective in this term refers to the two strokes in the character glyph. POSIX.1-2008 never uses the term “double-quote” to refer to two apostrophes or quotation-marks.

### 3.139 Downshifting

The conversion of an uppercase character that has a single-character lowercase representation into this lowercase representation.

### 3.140 Driver

A module that controls data transferred to and received from devices.

**Note:** Drivers are traditionally written to be a part of the system implementation, although they are frequently written separately from the writing of the implementation. A driver may contain processor-specific code, and therefore be non-portable.

### 3.141 Effective Group ID

An attribute of a process that is used in determining various permissions, including file access permissions; see also [Section 3.189](#) (on page 63).

### 3.142 Effective User ID

An attribute of a process that is used in determining various permissions, including file access permissions; see also [Section 3.436](#) (on page 103).

### 3.143 Eight-Bit Transparency

The ability of a software component to process 8-bit characters without modifying or utilizing any part of the character in a way that is inconsistent with the rules of the current coded character set.

**3.144 Empty Directory**

A directory that contains, at most, directory entries for dot and dot-dot, and has exactly one link to it (other than its own dot entry, if one exists), in dot-dot. No other links to the directory may exist. It is unspecified whether an implementation can ever consider the root directory to be empty.

**3.145 Empty Line**

A line consisting of only a <newline>; see also [Section 3.75](#) (on page 44).

**3.146 Empty String (or Null String)**

A string whose first byte is a null byte.

**3.147 Empty Wide-Character String**

A wide-character string whose first element is a null wide-character code.

**3.148 Encoding Rule**

The rules used to convert between wide-character codes and multi-byte character codes.

**Note:** Stream Orientation and Encoding Rules are defined in detail in XSH [Section 2.5.2](#) (on page 498).

**3.149 Entire Regular Expression**

The concatenated set of one or more basic regular expressions or extended regular expressions that make up the pattern specified for string selection.

**Note:** Regular Expressions are defined in detail in [Chapter 9](#) (on page 181).

### 3.150 Epoch

The time zero hours, zero minutes, zero seconds, on January 1, 1970 Coordinated Universal Time (UTC).

**Note:** See also *Seconds Since the Epoch* defined in [Section 4.16](#) (on page 113).

### 3.151 Equivalence Class

A set of collating elements with the same primary collation weight.

Elements in an equivalence class are typically elements that naturally group together, such as all accented letters based on the same base letter.

The collation order of elements within an equivalence class is determined by the weights assigned on any subsequent levels after the primary weight.

### 3.152 Era

A locale-specific method for counting and displaying years.

**Note:** The *LC\_TIME* category is defined in detail in [Section 7.3.5](#) (on page 159).

### 3.153 Event Management

The mechanism that enables applications to register for and be made aware of external events such as data becoming available for reading.

### 3.154 Executable File

A regular file acceptable as a new process image file by the equivalent of the *exec* family of functions, and thus usable as one form of a utility. The standard utilities described as compilers can produce executable files, but other unspecified methods of producing executable files may also be provided. The internal format of an executable file is unspecified, but a conforming application cannot assume an executable file is a text file.

### 3.155 Execute

To perform command search and execution actions, as defined in the Shell and Utilities volume of POSIX.1-2008; see also [Section 3.201](#) (on page 65).

**Note:** Command Search and Execution is defined in detail in XCU [Section 2.9.1.1](#) (on page 2357).

### 3.156 Execution Time

See *CPU Time* in [Section 3.118](#) (on page 52).

### 3.157 Execution Time Monitoring

A set of execution time monitoring primitives that allow online measuring of thread and process execution times.

### 3.158 Expand

In the shell command language, when not qualified, the act of applying word expansions.

**Note:** Word Expansions are defined in detail in XCU [Section 2.6](#) (on page 2343).

### 3.159 Extended Regular Expression (ERE)

A regular expression (see also [Section 3.321](#), on page 85) that is an alternative to the Basic Regular Expression using a more extensive syntax, occasionally used by some utilities.

**Note:** Extended Regular Expressions are described in detail in [Section 9.4](#) (on page 188).

### 3.160 Extended Security Controls

Implementation-defined security controls allowed by the file access permission and appropriate privileges (see also [Section 3.20](#), on page 36) mechanisms, through which an implementation can support different security policies from those described in POSIX.1-2008.

**Note:** See also *Extended Security Controls* defined in [Section 4.4](#) (on page 108).

File Access Permissions are defined in detail in [Section 4.5](#) (on page 108).

### 3.161 Feature Test Macro

A macro used to determine whether a particular set of features is included from a header.

**Note:** See also XSH [Section 2.2](#) (on page 472).

### 3.162 Field

In the shell command language, a unit of text that is the result of parameter expansion, arithmetic expansion, command substitution, or field splitting. During command processing, the resulting fields are used as the command name and its arguments.

**Note:** Parameter Expansion is defined in detail in XCU [Section 2.6.2](#) (on page 2344).

Arithmetic Expansion is defined in detail in XCU [Section 2.6.4](#) (on page 2348).

Command Substitution is defined in detail in XCU [Section 2.6.3](#) (on page 2347).

Field Splitting is defined in detail in XCU [Section 2.6.5](#) (on page 2349).

For further information on command processing, see XCU [Section 2.9.1](#) (on page 2355).

### 3.163 FIFO Special File (or FIFO)

A type of file with the property that data written to such a file is read on a first-in-first-out basis.

**Note:** Other characteristics of FIFOs are described in the System Interfaces volume of POSIX.1-2008, *lseek()*, *open()*, *read()*, and *write()*.

### 3.164 File

An object that can be written to, or read from, or both. A file has certain attributes, including access permissions and type. File types include regular file, character special file, block special file, FIFO special file, symbolic link, socket, and directory. Other types of files may be supported by the implementation.

### 3.165 File Description

See *Open File Description* in [Section 3.258](#) (on page 74).

**3.166 File Descriptor**

A per-process unique, non-negative integer used to identify an open file for the purpose of file access. The value of a newly-created file descriptor is from zero to {OPEN\_MAX}-1. A file descriptor can have a value greater than or equal to {OPEN\_MAX} if the value of {OPEN\_MAX} has decreased (see *sysconf()*) since the file descriptor was opened. File descriptors may also be used to implement message catalog descriptors and directory streams; see also [Section 3.258](#) (on page 74).

**Note:** {OPEN\_MAX} is defined in detail in [<limits.h>](#).

**3.167 File Group Class**

The property of a file indicating access permissions for a process related to the group identification of a process. A process is in the file group class of a file if the process is not in the file owner class and if the effective group ID or one of the supplementary group IDs of the process matches the group ID associated with the file. Other members of the class may be implementation-defined.

**3.168 File Mode**

An object containing the file mode bits and some information about the file type of a file.

**Note:** File mode bits and file types are defined in detail in [<sys/stat.h>](#).

**3.169 File Mode Bits**

A file's file permission bits, set-user-ID-on-execution bit (S\_ISUID), set-group-ID-on-execution bit (S\_ISGID), and, on directories, the restricted deletion flag bit (S\_ISVTX).

**Note:** File Mode Bits are defined in detail in [<sys/stat.h>](#).

**3.170 Filename**

A sequence of bytes consisting of 1 to {NAME\_MAX} bytes used to name a file. The bytes composing the name shall not contain the <NUL> or <slash> characters. In the context of a pathname, each filename shall be followed by a <slash> or a <NUL> character; elsewhere, a filename followed by a <NUL> character forms a string (but not necessarily a character string). The filenames **dot** and **dot-dot** have special meaning. A filename is sometimes referred to as a "pathname component". See also [Section 3.271](#) (on page 76).

**Note:** Pathname Resolution is defined in detail in [Section 4.13](#) (on page 111).



**3.171 Filename String**

A string consisting of a filename followed by a <NUL> character.

**3.172 File Offset**

The byte position in the file where the next I/O operation begins. Each open file description associated with a regular file, block special file, or directory has a file offset. A character special file that does not refer to a terminal device may have a file offset. There is no file offset specified for a pipe or FIFO.

**3.173 File Other Class**

The property of a file indicating access permissions for a process related to the user and group identification of a process. A process is in the file other class of a file if the process is not in the file owner class or file group class.

**3.174 File Owner Class**

The property of a file indicating access permissions for a process related to the user identification of a process. A process is in the file owner class of a file if the effective user ID of the process matches the user ID of the file.

**3.175 File Permission Bits**

Information about a file that is used, along with other information, to determine whether a process has read, write, or execute/search permission to a file. The bits are divided into three parts: owner, group, and other. Each part is used with the corresponding file class of processes. These bits are contained in the file mode.

**Note:** File modes are defined in detail in [<sys/stat.h>](#).

File Access Permissions are defined in detail in [Section 4.5](#) (on page 108).

**3.176 File Serial Number**

A per-file system unique identifier for a file.

**3.177 File System**

A collection of files and certain of their attributes. It provides a name space for file serial numbers referring to those files.

**3.178 File Type**

See *File* in [Section 3.164](#) (on page 59).

**3.179 Filter**

A command whose operation consists of reading data from standard input or a list of input files and writing data to standard output. Typically, its function is to perform some transformation on the data stream.

**3.180 First Open (of a File)**

When a process opens a file that is not currently an open file within any process.

**3.181 Flow Control**

The mechanism employed by a communications provider that constrains a sending entity to wait until the receiving entities can safely receive additional data without loss.

**3.182 Foreground Job**

See *Foreground Process Group* in [Section 3.184](#).

**3.183 Foreground Process**

A process that is a member of a foreground process group.

**3.184 Foreground Process Group (or Foreground Job)**

A process group whose member processes have certain privileges, denied to processes in background process groups, when accessing their controlling terminal. Each session that has established a connection with a controlling terminal has at most one process group of the session as the foreground process group of that controlling terminal.

**Note:** The General Terminal Interface is defined in detail in [Chapter 11](#).

**3.185 Foreground Process Group ID**

The process group ID of the foreground process group.

### 3.186 Form-Feed Character (<form-feed>)

A character that in the output stream indicates that printing should start on the next page of an output device. It is the character designated by ‘\f’ in the C language. If the form-feed is not the first character of an output line, the result is unspecified. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the movement to the next page.

### 3.187 Graphic Character

A member of the **graph** character class of the current locale.

**Note:** The **graph** character class is defined in detail in [Section 7.3.1](#) (on page 139).

### 3.188 Group Database

A system database that contains at least the following information for each group ID:

- Group name
- Numerical group ID
- List of users allowed in the group

The list of users allowed in the group is used by the *newgrp* utility.

**Note:** The *newgrp* utility is defined in detail in the Shell and Utilities volume of POSIX.1-2008.

### 3.189 Group ID

A non-negative integer, which can be contained in an object of type **gid\_t**, that is used to identify a group of system users. Each system user is a member of at least one group. When the identity of a group is associated with a process, a group ID value is referred to as a real group ID, an effective group ID, one of the supplementary group IDs, or a saved set-group-ID. The value (**gid\_t**)-1 shall not be a valid group ID, but does have a defined use in some interfaces defined in this standard.

### 3.190 Group Name

A string that is used to identify a group; see also [Section 3.188](#). To be portable across conforming systems, the value is composed of characters from the portable filename character set. The <hyphen-minus> should not be used as the first character of a portable group name.

**3.191 Hard Limit**

A system resource limitation that may be reset to a lesser or greater limit by a privileged process. A non-privileged process is restricted to only lowering its hard limit.

**3.192 Hard Link**

The relationship between two directory entries that represent the same file; see also [Section 3.130](#) (on page 53). The result of an execution of the *ln* utility (without the *-s* option) or the *link()* function. This term is contrasted against symbolic link; see also [Section 3.381](#) (on page 95).

**3.193 Home Directory**

The directory specified by the *HOME* environment variable.

**3.194 Host Byte Order**

The arrangement of bytes in any integer type when using a specific machine architecture.

**Note:** Two common methods of byte ordering are big-endian and little-endian. Big-endian is a format for storage of binary data in which the most significant byte is placed first, with the rest in descending order. Little-endian is a format for storage or transmission of binary data in which the least significant byte is placed first, with the rest in ascending order. See also [Section 4.10](#) (on page 110).

**3.195 Incomplete Line**

A sequence of one or more non-*<newline>* characters at the end of the file.

**3.196 Inf**

A value representing +infinity or a value representing -infinity that can be stored in a floating type. Not all systems support the Inf values.

**3.197 Instrumented Application**

An application that contains at least one call to the trace point function *posix\_trace\_event()*. Each process of an instrumented application has a mapping of trace event names to trace event type identifiers. This mapping is used by the trace stream that is created for that process.

**3.198 Interactive Shell**

A processing mode of the shell that is suitable for direct user interaction.

**3.199 Internationalization**

The provision within a computer program of the capability of making itself adaptable to the requirements of different native languages, local customs, and coded character sets.

**3.200 Interprocess Communication**

A functionality enhancement to add a high-performance, deterministic interprocess communication facility for local communication.

**3.201 Invoke**

To perform command search and execution actions, except that searching for shell functions and special built-in utilities is suppressed; see also [Section 3.155](#) (on page 58).

**Note:** Command Search and Execution is defined in detail in XCU [Section 2.9.1.1](#) (on page 2357).

**3.202 Job**

A set of processes, comprising a shell pipeline, and any processes descended from it, that are all in the same process group.

**Note:** See also XCU [Section 2.9.2](#) (on page 2358).

**3.203 Job Control**

A facility that allows users selectively to stop (suspend) the execution of processes and continue (resume) their execution at a later point. The user typically employs this facility via the interactive interface jointly supplied by the terminal I/O driver and a command interpreter.

**3.204 Job Control Job ID**

A handle that is used to refer to a job. The job control job ID can be any of the forms shown in the following table:

**Table 3-1** Job Control Job ID Formats

Job Control Job ID	Meaning
%%	Current job.
%+	Current job.
%-	Previous job.
% <i>n</i>	Job number <i>n</i> .
% <i>string</i>	Job whose command begins with <i>string</i> .
%? <i>string</i>	Job whose command contains <i>string</i> .

**3.205 Last Close (of a File)**

When a process closes a file, resulting in the file not being an open file within any process.

**3.206 Line**

A sequence of zero or more non-<newline> characters plus a terminating <newline> character.

**3.207 Linger**

The period of time before terminating a connection, to allow outstanding data to be transferred.

**3.208 Link**

See *Directory Entry* in [Section 3.130](#) (on page 53).

**3.209 Link Count**

The number of directory entries that refer to a particular file.

**3.210 Live Process**

An address space with one or more threads executing within that address space, and the required system resources for those threads. +

1957 **Note:** Many of the system resources defined by POSIX.1-2008 are shared among all of the threads +  
 1958 within a process. These include the process ID, the parent process ID, process group ID, session +  
 1959 membership, real, effective, and saved set-user-ID, real, effective, and saved set-group-ID, +  
 1960 supplementary group IDs, current working directory, root directory, file mode creation mask, +  
 1961 and file descriptors. +

### 1962 **3.211 Local Customs**

1963 The conventions of a geographical area or territory for such things as date, time, and currency  
 1964 formats.

### 1965 **3.212 Local Interprocess Communication (Local IPC)**

1966 The transfer of data between processes in the same system.

### 1967 **3.213 Locale**

1968 The definition of the subset of a user's environment that depends on language and cultural  
 1969 conventions.

1970 **Note:** Locales are defined in detail in [Chapter 7](#) (on page 135).

### 1971 **3.214 Localization**

1972 The process of establishing information within a computer system specific to the operation of  
 1973 particular native languages, local customs, and coded character sets.

### 1974 **3.215 Login**

1975 The unspecified activity by which a user gains access to the system. Each login is associated  
 1976 with exactly one login name.

### 1977 **3.216 Login Name**

1978 A user name that is associated with a login.

### 1979 **3.217 Map**

1980 To create an association between a page-aligned range of the address space of a process and  
 1981 some memory object, such that a reference to an address in that range of the address space

1982 results in a reference to the associated memory object. The mapped memory object is not  
 1983 necessarily memory-resident.

### 1984 **3.218 Marked Message**

1985 A STREAMS message on which a certain flag is set. Marking a message gives the application  
 1986 protocol-specific information. An application can use *ioctl()* to determine whether a given  
 1987 message is marked.

1988 **Note:** The *ioctl()* function is defined in detail in the System Interfaces volume of POSIX.1-2008.

### 1989 **3.219 Matched**

1990 A state applying to a sequence of zero or more characters when the characters in the sequence  
 1991 correspond to a sequence of characters defined by a basic regular expression or extended regular  
 1992 expression pattern.

1993 **Note:** Regular Expressions are defined in detail in [Chapter 9](#) (on page 181).

### 1994 **3.220 Memory Mapped Files**

1995 A facility to allow applications to access files as part of the address space.

### 1996 **3.221 Memory Object**

1997 One of:

- 1998 • A file (see [Section 3.164](#), on page 59)
- 1999 • A shared memory object (see [Section 3.346](#), on page 89)
- 2000 • A typed memory object (see [Section 3.429](#), on page 102)

2001 When used in conjunction with *mmap()*, a memory object appears in the address space of the  
 2002 calling process.

2003 **Note:** The *mmap()* function is defined in detail in the System Interfaces volume of POSIX.1-2008.

### 2004 **3.222 Memory-Resident**

2005 The process of managing the implementation in such a way as to provide an upper bound on  
 2006 memory access times.



### 2007 3.223 Message

2008 In the context of programmatic message passing, information that can be transferred between  
2009 processes or threads by being added to and removed from a message queue. A message consists  
2010 of a fixed-size message buffer.

### 2011 3.224 Message Catalog

2012 In the context of providing natural language messages to the user, a file or storage area  
2013 containing program messages, command prompts, and responses to prompts for a particular  
2014 native language, territory, and codeset.

### 2015 3.225 Message Catalog Descriptor

2016 In the context of providing natural language messages to the user, a per-process unique value  
2017 used to identify an open message catalog. A message catalog descriptor may be implemented  
2018 using a file descriptor.

### 2019 3.226 Message Queue

2020 In the context of programmatic message passing, an object to which messages can be added and  
2021 removed. Messages may be removed in the order in which they were added or in priority order.

### 2022 3.227 Mode

2023 A collection of attributes that specifies a file's type and its access permissions.

2024 **Note:** File Access Permissions are defined in detail in [Section 4.5](#) (on page 108).

### 2025 3.228 Monotonic Clock

2026 A clock measuring real time, whose value cannot be set via `clock_settime()` and which cannot  
2027 have negative clock jumps.

### 2028 3.229 Mount Point

2029 Either the system root directory or a directory for which the *st\_dev* field of structure **stat** differs  
2030 from that of its parent directory.

2031 **Note:** The **stat** structure is defined in detail in [<sys/stat.h>](#).

### 2032 3.230 Multi-Character Collating Element

2033 A sequence of two or more characters that collate as an entity. For example, in some coded  
2034 character sets, an accented character is represented by a non-spacing accent, followed by the  
2035 letter. Other examples are the Spanish elements *ch* and *ll*.

### 2036 3.231 Multi-Threaded Library

2037 A library containing object files that were produced by compiling with *c99* using the flags +  
2038 output by *getconf* POSIX\_V7\_THREADS\_CFLAGS, or by compiling using a non-standard utility +  
2039 with equivalent flags, and which makes use of interfaces that are only made available by *c99* +  
2040 when the **-l pthread** option is used or makes use of SIGEV\_THREAD notifications. +

### 2041 3.232 Multi-Threaded Process

2042 A process that contains more than one thread. +

### 2043 3.233 Multi-Threaded Program

2044 A program whose executable file was produced by compiling with *c99* using the flags output by +  
2045 *getconf* POSIX\_V7\_THREADS\_CFLAGS, and linking with *c99* using the flags output by *getconf* +  
2046 POSIX\_V7\_THREADS\_LDFLAGS and the **-l pthread** option, or by compiling and linking using +  
2047 a non-standard utility with equivalent flags. Execution of a multi-threaded program initially +  
2048 creates a single-threaded process; the process can create additional threads using *pthread\_create()* +  
2049 or SIGEV\_THREAD notifications. +

### 2050 3.234 Mutex

2051 A synchronization object used to allow multiple threads to serialize their access to shared data.  
2052 The name derives from the capability it provides; namely, mutual-exclusion. The thread that has  
2053 locked a mutex becomes its owner and remains the owner until that same thread unlocks the  
2054 mutex.

### 3.235 Name

In the shell command language, a word consisting solely of underscores, digits, and alphabetic characters from the portable character set. The first character of a name is not a digit.

**Note:** The Portable Character Set is defined in detail in [Section 6.1](#) (on page 125).

### 3.236 Named STREAM

A STREAMS-based file descriptor that is attached to a name in the file system name space. All subsequent operations on the named STREAM act on the STREAM that was associated with the file descriptor until the name is disassociated from the STREAM.

### 3.237 NaN (Not a Number)

A set of values that may be stored in a floating type but that are neither Inf nor valid floating-point numbers. Not all systems support NaN values.

### 3.238 Native Language

A computer user's spoken or written language, such as American English, British English, Danish, Dutch, French, German, Italian, Japanese, Norwegian, or Swedish.

### 3.239 Negative Response

An input string that matches one of the responses acceptable to the *LC\_MESSAGES* category keyword **noexpr**, matching an extended regular expression in the current locale.

**Note:** The *LC\_MESSAGES* category is defined in detail in [Section 7.3.6](#) (on page 165).

### 3.240 Network

A collection of interconnected hosts.

**Note:** The term "network" in POSIX.1-2008 is used to refer to the network of hosts. The term "batch system" is used to refer to the network of batch servers.

### 3.241 Network Address

A network-visible identifier used to designate specific endpoints in a network. Specific endpoints on host systems have addresses, and host systems may also have addresses.

### 3.242 Network Byte Order

The way of representing any integer type such that, when transmitted over a network via a network endpoint, the **int** type is transmitted as an appropriate number of octets with the most significant octet first, followed by any other octets in descending order of significance.

**Note:** This order is more commonly known as big-endian ordering. See also [Section 4.10](#) (on page 110).

### 3.243 Newline Character (<newline>)

A character that in the output stream indicates that printing should start at the beginning of the next line. It is the character designated by '`\n`' in the C language. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the movement to the next line.

### 3.244 Nice Value

A number used as advice to the system to alter process scheduling. Numerically smaller values give a process additional preference when scheduling a process to run. Numerically larger values reduce the preference and make a process less likely to run. Typically, a process with a smaller nice value runs to completion more quickly than an equivalent process with a higher nice value. The symbol {NZERO} specifies the default nice value of the system.

### 3.245 Non-Blocking

A property of an open file description that causes function calls involving it to return without delay when it is detected that the requested action associated with the function call cannot be completed without unknown delay.

**Note:** The exact semantics are dependent on the type of file associated with the open file description. For data reads from devices such as ttys and FIFOs, this property causes the read to return immediately when no data was available. Similarly, for writes, it causes the call to return immediately when the thread would otherwise be delayed in the write operation; for example, because no space was available. For networking, it causes functions not to await protocol events (for example, acknowledgements) to occur. See also XSH [Section 2.10.7](#) (on page 525).

### 3.246 Non-Spacing Characters

A character, such as a character representing a diacritical mark in the ISO/IEC 6937:2001 standard coded graphic character set, which is used in combination with other characters to form composite graphic symbols.

2111 **3.247 NUL**

2112 A character with all bits set to zero.

2113 **3.248 Null Byte**

2114 A byte with all bits set to zero.

2115 **3.249 Null Pointer**

2116 A pointer obtained by converting an integer constant expression with the value 0, or such an  
 2117 expression cast to type **void \***, to a pointer type; for example, **(char \*)0**. The C language  
 2118 guarantees that a null pointer compares unequal to a pointer to any object or function, so it is  
 2119 used by many functions that return pointers to indicate an error.

2120 **3.250 Null String**2121 See *Empty String* in [Section 3.146](#) (on page 56).2122 **3.251 Null Wide-Character Code**

2123 A wide-character code with all bits set to zero.

2124 **3.252 Number-Sign Character (<number-sign>)**

2125 The character ' # ', also known as hash sign.

2126 **3.253 Object File**

2127 A regular file containing the output of a compiler, formatted as input to a linkage editor for  
 2128 linking with other object files into an executable form. The methods of linking are unspecified  
 2129 and may involve the dynamic linking of objects at runtime. The internal format of an object file  
 2130 is unspecified, but a conforming application cannot assume an object file is a text file.

2131 **3.254 Octet**

2132 Unit of data representation that consists of eight contiguous bits.

**2133 3.255 Offset Maximum**

2134 An attribute of an open file description representing the largest value that can be used as a file  
2135 offset.

**2136 3.256 Opaque Address**

2137 An address such that the entity making use of it requires no details about its contents or format.

**2138 3.257 Open File**

2139 A file that is currently associated with a file descriptor.

**2140 3.258 Open File Description**

2141 A record of how a process or group of processes is accessing a file. Each file descriptor refers to  
2142 exactly one open file description, but an open file description can be referred to by more than  
2143 one file descriptor. The file offset, file status, and file access modes are attributes of an open file  
2144 description.

**2145 3.259 Operand**

2146 An argument to a command that is generally used as an object supplying information to a utility  
2147 necessary to complete its processing. Operands generally follow the options in a command line.

2148 **Note:** Utility Argument Syntax is defined in detail in [Section 12.1](#) (on page 213).

**2149 3.260 Operator**

2150 In the shell command language, either a control operator or a redirection operator.

### 3.261 Option

An argument to a command that is generally used to specify changes in the utility's default behavior.

**Note:** Utility Argument Syntax is defined in detail in [Section 12.1](#) (on page 213).

### 3.262 Option-Argument

A parameter that follows certain options. In some cases an option-argument is included within the same argument string as the option—in most cases it is the next argument.

**Note:** Utility Argument Syntax is defined in detail in [Section 12.1](#) (on page 213).

### 3.263 Orientation

A stream has one of three orientations: unoriented, byte-oriented, or wide-oriented.

**Note:** For further information, see XSH [Section 2.5.2](#) (on page 498).

### 3.264 Orphaned Process Group

A process group in which the parent of every member is either itself a member of the group or is not a member of the group's session.

### 3.265 Page

The granularity of process memory mapping or locking.

Physical memory and memory objects can be mapped into the address space of a process on page boundaries and in integral multiples of pages. Process address space can be locked into memory (made memory-resident) on page boundaries and in integral multiples of pages.

### 3.266 Page Size

The size, in bytes, of the system unit of memory allocation, protection, and mapping. On systems that have segment rather than page-based memory architectures, the term "page" means a segment.

**3.267 Parameter**

In the shell command language, an entity that stores values. There are three types of parameters: variables (named parameters), positional parameters, and special parameters. Parameter expansion is accomplished by introducing a parameter with the '\$' character.

**Note:** See also XCU [Section 2.5](#) (on page 2339).

In the C language, an object declared as part of a function declaration or definition that acquires a value on entry to the function, or an identifier following the macro name in a function-like macro definition.

**3.268 Parent Directory**

When discussing a given directory, the directory that both contains a directory entry for the given directory and is represented by the pathname dot-dot in the given directory.

When discussing other types of files, a directory containing a directory entry for the file under discussion.

This concept does not apply to dot and dot-dot.

**3.269 Parent Process**

The process which created (or inherited) the process under discussion.

**3.270 Parent Process ID**

An attribute of a new process identifying the parent of the process. The parent process ID of a process is the process ID of its creator, for the lifetime of the creator. After the creator's lifetime has ended, the parent process ID is the process ID of an implementation-defined system process.

**3.271 Pathname**

A string that is used to identify a file. In the context of POSIX.1-2008, a pathname may be limited to {PATH\_MAX} bytes, including the terminating null byte. It has optional beginning <slash> characters, followed by zero or more filenames separated by <slash> characters. A pathname can optionally contain one or more trailing <slash> characters. Multiple successive <slash> characters are considered to be the same as one <slash>, except for the case of exactly two leading <slash> characters.

**Note:** If a pathname consists of only bytes corresponding to characters from the portable filename character set (see [Section 3.282](#), on page 79), <slash> characters, and a single terminating <NUL> character, the pathname will be usable as a character string in all supported locales; otherwise, the pathname might only be a string (rather than a character string). Additionally, since the single-byte encoding of the <slash> character is required to be the same across all locales and to not occur within a multi-byte character, references to a <slash> character within a pathname are well-defined even when the pathname is not a character string. However, this property does not necessarily hold for the remaining characters within the portable filename



2209 character set.

2210 Pathname Resolution is defined in detail in [Section 4.13](#) (on page 111).

### 2211 3.272 Pathname Component

2212 See *Filename* in [Section 3.170](#) (on page 60).

### 2213 3.273 Path Prefix

2214 The part of a pathname up to, but not including, the last component and any trailing <slash>  
2215 characters, unless the pathname consists entirely of <slash> characters, in which case the path  
2216 prefix is ' / ' for a pathname containing either a single <slash> or three or more <slash>  
2217 characters, and ' / / ' for the pathname //. The path prefix of a pathname containing no <slash>  
2218 characters is empty, but is treated as referring to the current working directory.

2219 **Note:** The term is used both in the sense of identifying part of a pathname that forms the prefix and of  
2220 joining a non-empty path prefix to a filename to form a pathname. In the latter case, the path  
2221 prefix need not have a trailing <slash> (in which case the joining is done with a <slash>  
2222 character).

### 2223 3.274 Pattern

2224 A sequence of characters used either with regular expression notation or for pathname  
2225 expansion, as a means of selecting various character strings or pathnames, respectively.

2226 **Note:** Regular Expressions are defined in detail in [Chapter 9](#) (on page 181).

2227 See also XCU [Section 2.6.6](#) (on page 2350).

2228 The syntaxes of the two types of patterns are similar, but not identical; POSIX.1-2008 always  
2229 indicates the type of pattern being referred to in the immediate context of the use of the term.

### 2230 3.275 Period Character (<period>)

2231 The character ' . '. The term “period” is contrasted with dot (see also [Section 3.136](#), on page 54),  
2232 which is used to describe a specific directory entry.

### 3.276 Permissions

Attributes of an object that determine the privilege necessary to access or manipulate the object.

**Note:** File Access Permissions are defined in detail in [Section 4.5](#) (on page 108).

### 3.277 Persistence

A mode for semaphores, shared memory, and message queues requiring that the object and its state (including data, if any) are preserved after the object is no longer referenced by any process.

Persistence of an object does not imply that the state of the object is maintained across a system crash or a system reboot.

### 3.278 Pipe

An object identical to a FIFO which has no links in the file hierarchy.

**Note:** The *pipe()* function is defined in detail in the System Interfaces volume of POSIX.1-2008.

### 3.279 Polling

A scheduling scheme whereby the local process periodically checks until the pre-specified events (for example, read, write) have occurred.

### 3.280 Portable Character Set

The collection of characters that are required to be present in all locales supported by conforming systems.

**Note:** The Portable Character Set is defined in detail in [Section 6.1](#) (on page 125).

This term is contrasted against the smaller portable filename character set; see also [Section 3.282](#) (on page 79).

**3.281 Portable Filename**

A filename consisting only of characters from the portable filename character set.

**Note:** Applications should avoid using filenames that have the <hyphen> character as the first character since this may cause problems when filenames are passed as command line arguments.

**3.282 Portable Filename Character Set**

The set of characters from which portable filenames are constructed.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z  
a b c d e f g h i j k l m n o p q r s t u v w x y z  
0 1 2 3 4 5 6 7 8 9 . \_ -

The last three characters are the <period>, <underscore>, and <hyphen-minus> characters, respectively. See also [Section 3.271](#) (on page 76).

**3.283 Positional Parameter**

In the shell command language, a parameter denoted by a single digit or one or more digits in curly braces.

**Note:** For further information, see XCU [Section 2.5.1](#) (on page 2339).

**3.284 Preallocation**

The reservation of resources in a system for a particular use.

Preallocation does not imply that the resources are immediately allocated to that use, but merely indicates that they are guaranteed to be available in bounded time when needed.

**3.285 Preempted Process (or Thread)**

A running thread whose execution is suspended due to another thread becoming runnable at a higher priority.

**3.286 Previous Job**

In the context of job control, the job that will be used as the default for the *fg* or *bg* utilities if the current job exits. There is at most one previous job; see also [Section 3.204](#) (on page 66).

### 2280 3.287 Printable Character

2281 One of the characters included in the **print** character classification of the *LC\_CTYPE* category in  
2282 the current locale.

2283 **Note:** The *LC\_CTYPE* category is defined in detail in [Section 7.3.1](#) (on page 139).

### 2284 3.288 Printable File

2285 A text file consisting only of the characters included in the **print** and **space** character  
2286 classifications of the *LC\_CTYPE* category and the <backspace>, all in the current locale.

2287 **Note:** The *LC\_CTYPE* category is defined in detail in [Section 7.3.1](#) (on page 139).

### 2288 3.289 Priority

2289 A non-negative integer associated with processes or threads whose value is constrained to a  
2290 range defined by the applicable scheduling policy. Numerically higher values represent higher  
2291 priorities.

### 2292 3.290 Priority Band

2293 The queuing order applied to normal priority STREAMS messages. High priority STREAMS  
2294 messages are not grouped by priority bands. The only differentiation made by the STREAMS  
2295 mechanism is between zero and non-zero bands, but specific protocol modules may differentiate  
2296 between priority bands.

### 2297 3.291 Priority Inversion

2298 A condition in which a thread that is not voluntarily suspended (waiting for an event or time  
2299 delay) is not running while a lower priority thread is running. Such blocking of the higher  
2300 priority thread is often caused by contention for a shared resource.

### 2301 3.292 Priority Scheduling

2302 A performance and determinism improvement facility to allow applications to determine the  
2303 order in which threads that are ready to run are granted access to processor resources.

### 2304 3.293 Priority-Based Scheduling

2305 Scheduling in which the selection of a running thread is determined by the priorities of the  
2306 runnable processes or threads.

### 3.294 Privilege

See *Appropriate Privileges* in [Section 3.20](#) (on page 36).

### 3.295 Process

A live process (see [Section 3.210](#), on page 66) or a zombie process (see [Section 3.452](#), on page 106). The lifetime of a process is described in [Section 3.301](#) (on page 82).

### 3.296 Process Group

A collection of processes that permits the signaling of related processes. Each process in the system is a member of a process group that is identified by a process group ID. A newly created process joins the process group of its creator.

### 3.297 Process Group ID

The unique positive integer identifier representing a process group during its lifetime.

**Note:** See also *Process Group ID Reuse* defined in [Section 4.14](#) (on page 113).

### 3.298 Process Group Leader

A process whose process ID is the same as its process group ID.

### 3.299 Process Group Lifetime

The period of time that begins when a process group is created and ends when the last remaining process in the group leaves the group, due either to the end of the lifetime of the last process or to the last remaining process calling the *setsid()* or *setpgid()* functions.

**Note:** The *setsid()* and *setpgid()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

### 3.300 Process ID

The unique positive integer identifier representing a process during its lifetime.

**Note:** See also *Process ID Reuse* defined in [Section 4.14](#) (on page 113).

### 3.301 Process Lifetime

The period of time that begins when a process is created and ends when its process ID is returned to the system.

See also [Section 3.210](#) (on page 66), [Section 3.303](#), and [Section 3.452](#) (on page 106).

**Note:** Process creation is defined in detail in the descriptions of the *fork()*, *posix\_spawn()*, and *posix\_spawnnp()*, functions in the System Interfaces volume of POSIX.1-2008.

### 3.302 Process Memory Locking

A performance improvement facility to bind application programs into the high-performance random access memory of a computer system. This avoids potential latencies introduced by the operating system in storing parts of a program that were not recently referenced on secondary memory devices.

### 3.303 Process Termination

There are two kinds of process termination:

1. Normal termination occurs by a return from *main()*, when requested with the *exit()*, *\_exit()*, or *\_Exit()* functions; or when the last thread in the process terminates by returning from its start function, by calling the *pthread\_exit()* function, or through cancellation.
2. Abnormal termination occurs when requested by the *abort()* function or when some signals are received.

**Note:** The consequences of process termination can be found in the description of the *\_Exit()* function in the System Interfaces volume of POSIX.1-2008. The *\_exit()*, *\_Exit()*, *abort()*, and *exit()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

### 3.304 Process-To-Process Communication

The transfer of data between processes.

### 3.305 Process Virtual Time

The measurement of time in units elapsed by the system clock while a process is executing.

### 3.306 Program

A prepared sequence of instructions to the system to accomplish a defined task. The term “program” in POSIX.1-2008 encompasses applications written in the Shell Command Language, complex utility input languages (for example, *awk*, *lex*, *sed*, and so on), and high-level languages.

### 3.307 Protocol

A set of semantic and syntactic rules for exchanging information.

### 3.308 Pseudo-Terminal

A facility that provides an interface that is identical to the terminal subsystem, except where noted otherwise in POSIX.1-2008. A pseudo-terminal is composed of two devices: the “master device” and a “slave device”. The slave device provides processes with an interface that is identical to the terminal interface, although there need not be hardware behind that interface. Anything written on the master device is presented to the slave as an input and anything written on the slave device is presented as an input on the master side.

### 3.309 Radix Character

The character that separates the integer part of a number from the fractional part.

### 3.310 Read-Only File System

A file system that has implementation-defined characteristics restricting modifications.

**Note:** File Times Update is described in detail in [Section 4.9](#) (on page 109).

### 3.311 Read-Write Lock

Multiple readers, single writer (read-write) locks allow many threads to have simultaneous read-only access to data while allowing only one thread to have write access at any given time. They are typically used to protect data that is read-only more frequently than it is changed.

Read-write locks can be used to synchronize threads in the current process and other processes if they are allocated in memory that is writable and shared among the cooperating processes and have been initialized for this behavior.

### 2381 **3.312 Real Group ID**

2382 The attribute of a process that, at the time of process creation, identifies the group of the user  
2383 who created the process; see also [Section 3.189](#) (on page 63).

### 2384 **3.313 Real Time**

2385 Time measured as total units elapsed by the system clock without regard to which thread is  
2386 executing.

### 2387 **3.314 Realtime Signal Extension**

2388 A determinism improvement facility to enable asynchronous signal notifications to an  
2389 application to be queued without impacting compatibility with the existing signal functions.

### 2390 **3.315 Real User ID**

2391 The attribute of a process that, at the time of process creation, identifies the user who created the  
2392 process; see also [Section 3.436](#) (on page 103).

### 2393 **3.316 Record**

2394 A collection of related data units or words which is treated as a unit.

### 2395 **3.317 Redirection**

2396 In the shell command language, a method of associating files with the input or output of  
2397 commands.

2398 **Note:** For further information, see XCU [Section 2.7](#) (on page 2350).



### 3.318 Redirection Operator

In the shell command language, a token that performs a redirection function. It is one of the following symbols:

<      >      >|      <<      >>      <&      >&      <<-      <>

### 3.319 Referenced Shared Memory Object

A shared memory object that is open or has one or more mappings defined on it.

### 3.320 Refresh

To ensure that the information on the user's terminal screen is up-to-date.

### 3.321 Regular Expression

A pattern that selects specific strings from a set of character strings.

**Note:** Regular Expressions are described in detail in [Chapter 9](#) (on page 181).

### 3.322 Region

In the context of the address space of a process, a sequence of addresses.

In the context of a file, a sequence of offsets.

### 3.323 Regular File

A file that is a randomly accessible sequence of bytes, with no further structure imposed by the system.

### 2416 3.324 Relative Pathname

2417 A pathname not beginning with a <slash> character.

2418 **Note:** Pathname Resolution is defined in detail in [Section 4.13](#) (on page 111).

### 2419 3.325 Relocatable File

2420 A file holding code or data suitable for linking with other object files to create an executable or a  
2421 shared object file.

### 2422 3.326 Relocation

2423 The process of connecting symbolic references with symbolic definitions. For example, when a  
2424 program calls a function, the associated call instruction transfers control to the proper  
2425 destination address at execution.

### 2426 3.327 Requested Batch Service

2427 A service that is either rejected or performed prior to a response from the service to the  
2428 requester.

### 2429 3.328 (Time) Resolution

2430 The minimum time interval that a clock can measure or whose passage a timer can detect.

### 2431 3.329 Robust Mutex

2432 A mutex with the *robust* attribute set.

2433 **Note:** The *robust* attribute is defined in detail by the `pthread_mutexattr_getrobust()` function.

### 2434 3.330 Root Directory

2435 A directory, associated with a process, that is used in pathname resolution for pathnames that  
2436 begin with a <slash> character.

### 2437 3.331 Runnable Process (or Thread)

2438 A thread that is capable of being a running thread, but for which no processor is available.

### 3.332 Running Process (or Thread)

A thread currently executing on a processor. On multi-processor systems there may be more than one such thread in a system at a time.

### 3.333 Saved Resource Limits

An attribute of a process that provides some flexibility in the handling of unrepresentable resource limits, as described in the *exec* family of functions and *setrlimit()*.

**Note:** The *exec* and *setrlimit()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

### 3.334 Saved Set-Group-ID

An attribute of a process that allows some flexibility in the assignment of the effective group ID attribute, as described in the *exec* family of functions and *setgid()*.

**Note:** The *exec* and *setgid()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

### 3.335 Saved Set-User-ID

An attribute of a process that allows some flexibility in the assignment of the effective user ID attribute, as described in the *exec* family of functions and *setuid()*.

**Note:** The *exec* and *setuid()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

### 3.336 Scheduling

The application of a policy to select a runnable process or thread to become a running process or thread, or to alter one or more of the thread lists.

### 3.337 Scheduling Allocation Domain

The set of processors on which an individual thread can be scheduled at any given time.

### 3.338 Scheduling Contention Scope

A property of a thread that defines the set of threads against which that thread competes for resources.

2465 For example, in a scheduling decision, threads sharing scheduling contention scope compete for  
2466 processor resources. In POSIX.1-2008, a thread has scheduling contention scope of either  
2467 PTHREAD\_SCOPE\_SYSTEM or PTHREAD\_SCOPE\_PROCESS.

### 2468 3.339 Scheduling Policy

2469 A set of rules that is used to determine the order of execution of processes or threads to achieve  
2470 some goal.

2471 **Note:** Scheduling Policy is defined in detail in [Section 4.15](#) (on page 113).

### 2472 3.340 Screen

2473 A rectangular region of columns and lines on a terminal display. A screen may be a portion of a  
2474 physical display device or may occupy the entire physical area of the display device.

### 2475 3.341 Scroll

2476 To move the representation of data vertically or horizontally relative to the terminal screen.  
2477 There are two types of scrolling:

- 2478 1. The cursor moves with the data.
- 2479 2. The cursor remains stationary while the data moves.

### 2480 3.342 Semaphore

2481 A minimum synchronization primitive to serve as a basis for more complex synchronization  
2482 mechanisms to be defined by the application program.

2483 **Note:** Semaphores are defined in detail in [Section 4.17](#) (on page 114).

### 3.343 Session

A collection of process groups established for job control purposes. Each process group is a member of a session. A process is considered to be a member of the session of which its process group is a member. A newly created process joins the session of its creator. A process can alter its session membership; see *setsid()*. There can be multiple process groups in the same session.

**Note:** The *setsid()* function is defined in detail in the System Interfaces volume of POSIX.1-2008.

### 3.344 Session Leader

A process that has created a session.

**Note:** For further information, see the *setsid()* function defined in the System Interfaces volume of POSIX.1-2008.

### 3.345 Session Lifetime

The period between when a session is created and the end of the lifetime of all the process groups that remain as members of the session.

### 3.346 Shared Memory Object

An object that represents memory that can be mapped concurrently into the address space of more than one process.

### 3.347 Shell

A program that interprets sequences of text input as commands. It may operate on an input stream or it may interactively prompt and read commands from a terminal.

### 3.348 Shell, the

The Shell Command Language Interpreter; a specific instance of a shell.

**Note:** For further information, see the *sh* utility defined in the Shell and Utilities volume of POSIX.1-2008.

**3.349 Shell Script**

A file containing shell commands. If the file is made executable, it can be executed by specifying its name as a simple command. Execution of a shell script causes a shell to execute the commands within the script. Alternatively, a shell can be requested to execute the commands in a shell script by specifying the name of the shell script as the operand to the *sh* utility.

**Note:** Simple Commands are defined in detail in XCU [Section 2.9.1](#) (on page 2355).

The *sh* utility is defined in detail in the Shell and Utilities volume of POSIX.1-2008.

**3.350 Signal**

A mechanism by which a process or thread may be notified of, or affected by, an event occurring in the system. Examples of such events include hardware exceptions and specific actions by processes. The term signal is also used to refer to the event itself.

**3.351 Signal Stack**

Memory established for a thread, in which signal handlers catching signals sent to that thread are executed.

**3.352 Single-Quote Character**

The character designated by '`\``' in the C language, also known as <apostrophe>.

**3.353 Single-Threaded Process**

A process that contains a single thread.

**3.354 Single-Threaded Program**

A program whose executable file was produced by compiling with *c99* without using the flags output by *getconf* POSIX\_V7\_THREADS\_CFLAGS and linking with *c99* using neither the flags output by *getconf* POSIX\_V7\_THREADS\_LDFLAGS nor the **-l pthread** option, or by compiling and linking using a non-standard utility with equivalent flags. Execution of a single-threaded program creates a single-threaded process; if the process attempts to create additional threads using *pthread\_create()* or SIGEV\_THREAD notifications, the behavior is undefined. If the process uses *dlopen()* to load a multi-threaded library, the behavior is undefined.

### 2533 3.355 Slash Character (<slash>)

2534 The character ' / ', also known as solidus.

### 2535 3.356 Socket

2536 A file of a particular type that is used as a communications endpoint for process-to-process  
2537 communication as described in the System Interfaces volume of POSIX.1-2008.

### 2538 3.357 Socket Address

2539 An address associated with a socket or remote endpoint, including an address family identifier  
2540 and addressing information specific to that address family. The address may include multiple  
2541 parts, such as a network address associated with a host system and an identifier for a specific  
2542 endpoint.

### 2543 3.358 Soft Limit

2544 A resource limitation established for each process that the process may set to any value less than  
2545 or equal to the hard limit.

### 2546 3.359 Source Code

2547 When dealing with the Shell Command Language, input to the command language interpreter.  
2548 The term “shell script” is synonymous with this meaning.

2549 When dealing with an ISO/IEC-conforming programming language, source code is input to a  
2550 compiler conforming to that ISO/IEC standard.

2551 Source code also refers to the input statements prepared for the following standard utilities: *awk*,  
2552 *bc*, *ed*, *ex*, *lex*, *localedef*, *make*, *sed*, and *yacc*.

2553 Source code can also refer to a collection of sources meeting any or all of these meanings.

2554 **Note:** The *awk*, *bc*, *ed*, *ex*, *lex*, *localedef*, *make*, *sed*, and *yacc* utilities are defined in detail in the Shell and  
2555 Utilities volume of POSIX.1-2008.

**3.360 Space Character (<space>)**

The character defined in the portable character set as <space>. The <space> character is a member of the **space** character class of the current locale, but represents the single character, and not all of the possible members of the class; see also [Section 3.442](#) (on page 104).

**3.361 Spawn**

A process creation primitive useful for systems that have difficulty with *fork()* and as an efficient replacement for *fork()/exec*.

**3.362 Special Built-In**

See *Built-In Utility* in [Section 3.83](#) (on page 46).

**3.363 Special Parameter**

In the shell command language, a parameter named by a single character from the following list:

\* @ # ? ! - \$ 0

**Note:** For further information, see XCU [Section 2.5.2](#) (on page 2340).

**3.364 Spin Lock**

A synchronization object used to allow multiple threads to serialize their access to shared data.

**3.365 Sporadic Server**

A scheduling policy for threads and processes that reserves a certain amount of execution capacity for processing aperiodic events at a given priority level.

**3.366 Standard Error**

An output stream usually intended to be used for diagnostic messages.

**3.367 Standard Input**

An input stream usually intended to be used for primary data input.



### 3.368 Standard Output

An output stream usually intended to be used for primary data output.

### 3.369 Standard Utilities

The utilities described in the Shell and Utilities volume of POSIX.1-2008.

### 3.370 Stream

Appearing in lowercase, a stream is a file access object that allows access to an ordered sequence of characters, as described by the ISO C standard. Such objects can be created by the *fdopen()*, *fmemopen()*, *fopen()*, *open\_memstream()*, or *popen()* functions, and are associated with a file descriptor. A stream provides the additional services of user-selectable buffering and formatted input and output; see also [Section 3.371](#).

**Note:** For further information, see XSH [Section 2.5](#) (on page 495).

The *fdopen()*, *fmemopen()*, *fopen()*, *open\_memstream()*, and *popen()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

### 3.371 STREAM

Appearing in uppercase, STREAM refers to a full-duplex connection between a process and an open device or pseudo-device. It optionally includes one or more intermediate processing modules that are interposed between the process end of the STREAM and the device driver (or pseudo-device driver) end of the STREAM; see also [Section 3.370](#).

**Note:** For further information, see XSH [Section 2.6](#) (on page 500).

### 3.372 STREAM End

The STREAM end is the driver end of the STREAM and is also known as the downstream end of the STREAM.

### 3.373 STREAM Head

The STREAM head is the beginning of the STREAM and is at the boundary between the system and the application process. This is also known as the upstream end of the STREAM.

### 3.374 STREAMS Multiplexor

A driver with multiple STREAMS connected to it. Multiplexing with STREAMS connected above is referred to as N-to-1, or “upper multiplexing”. Multiplexing with STREAMS connected below is referred to as 1-to-N or “lower multiplexing”.

### 3.375 String

A contiguous sequence of bytes terminated by and including the first null byte.

### 3.376 Subshell

A shell execution environment, distinguished from the main or current shell execution environment.

**Note:** For further information, see XCU [Section 2.12](#) (on page 2371).

### 3.377 Successfully Transferred

For a write operation to a regular file, when the system ensures that all data written is readable on any subsequent open of the file (even one that follows a system or power failure) in the absence of a failure of the physical storage medium.

For a read operation, when an image of the data on the physical storage medium is available to the requesting process.

### 3.378 Supplementary Group ID

An attribute of a process used in determining file access permissions. A process has up to {NGROUPS\_MAX} supplementary group IDs in addition to the effective group ID. The supplementary group IDs of a process are set to the supplementary group IDs of the parent process when the process is created.

### 3.379 Suspended Job

A job that has received a SIGSTOP, SIGTSTP, SIGTTIN, or SIGTTOU signal that caused the process group to stop. A suspended job is a background job, but a background job is not necessarily a suspended job.

### 3.380 Symbolic Constant

An object-like macro defined with a constant value.

Unless stated otherwise, the following shall apply to every symbolic constant:

- It expands to a compile-time constant expression with an integer type.
- It may be defined as another type of constant—e.g., an enumeration constant—as well as being a macro.
- It need not be usable in `#if` preprocessing directives.

### 3.381 Symbolic Link

A type of file with the property that when the file is encountered during pathname resolution, a string stored by the file is used to modify the pathname resolution. The stored string has a length of {SYMLINK\_MAX} bytes or fewer.

**Note:** Pathname Resolution is defined in detail in [Section 4.13](#) (on page 111).

### 3.382 Synchronized Input and Output

A determinism and robustness improvement mechanism to enhance the data input and output mechanisms, so that an application can ensure that the data being manipulated is physically present on secondary mass storage devices.

### 3.383 Synchronized I/O Completion

The state of an I/O operation that has either been successfully transferred or diagnosed as unsuccessful.

### 3.384 Synchronized I/O Data Integrity Completion

For read, when the operation has been completed or diagnosed if unsuccessful. The read is complete only when an image of the data has been successfully transferred to the requesting process. If there were any pending write requests affecting the data to be read at the time that the synchronized read operation was requested, these write requests are successfully transferred prior to reading the data.

For write, when the operation has been completed or diagnosed if unsuccessful. The write is complete only when the data specified in the write request is successfully transferred and all file system information required to retrieve the data is successfully transferred.

File attributes that are not necessary for data retrieval (access time, modification time, status change time) need not be successfully transferred prior to returning to the calling process.

### 3.385 Synchronized I/O File Integrity Completion

Identical to a synchronized I/O data integrity completion with the addition that all file attributes relative to the I/O operation (including access time, modification time, status change time) are successfully transferred prior to returning to the calling process.

### 3.386 Synchronized I/O Operation

An I/O operation performed on a file that provides the application assurance of the integrity of its data and files.

### 3.387 Synchronous I/O Operation

An I/O operation that causes the thread requesting the I/O to be blocked from further use of the processor until that I/O operation completes.

**Note:** A synchronous I/O operation does not imply synchronized I/O data integrity completion or synchronized I/O file integrity completion.

### 3.388 Synchronously-Generated Signal

A signal that is attributable to a specific thread.

For example, a thread executing an illegal instruction or touching invalid memory causes a synchronously-generated signal. Being synchronous is a property of how the signal was generated and not a property of the signal number.

### 3.389 System

An implementation of POSIX.1-2008.

### 3.390 System Boot

An unspecified sequence of events that may result in the loss of transitory data; that is, data that is not saved in permanent storage. For example, message queues, shared memory, semaphores, and processes.

### 3.391 System Clock

A clock with at least one second resolution that contains seconds since the Epoch.

### 3.392 System Console

A device that receives messages sent by the *syslog()* function, and the *fntmsg()* function when the MM\_CONSOLE flag is set.

**Note:** The *syslog()* and *fntmsg()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

### 3.393 System Crash

An interval initiated by an unspecified circumstance that causes all processes (possibly other than special system processes) to be terminated in an undefined manner, after which any changes to the state and contents of files created or written to by an application prior to the interval are undefined, except as required elsewhere in POSIX.1-2008.

### 3.394 System Databases

An implementation provides two system databases: the “group database” (see also [Section 3.188](#), on page 63) and the “user database” (see also [Section 3.435](#), on page 103).

### 3.395 System Documentation

All documentation provided with an implementation except for the conformance document. Electronically distributed documents for an implementation are considered part of the system documentation.

### 3.396 System Process

An object other than a process executing an application, that is provided by the system and has a process ID.

### 3.397 System Reboot

See *System Boot* defined in [Section 3.390](#) (on page 96).

### 3.398 System Trace Event

A trace event that is generated by the implementation, in response either to a system-initiated action or to an application-requested action, except for a call to *posix\_trace\_event()*. When supported by the implementation, a system-initiated action generates a process-independent system trace event and an application-requested action generates a process-dependent system trace event. For a system trace event not defined by POSIX.1-2008, the associated trace event

2711 type identifier is derived from the implementation-defined name for this trace event, and the  
2712 associated data is of implementation-defined content and length.

### 2713 **3.399 System-Wide**

2714 Pertaining to events occurring in all processes existing in an implementation at a given point in  
2715 time.

### 2716 **3.400 Tab Character (<tab>)**

2717 A character that in the output stream indicates that printing or displaying should start at the  
2718 next horizontal tabulation position on the current line. It is the character designated by ‘\t’ in  
2719 the C language. If the current position is at or past the last defined horizontal tabulation  
2720 position, the behavior is unspecified. It is unspecified whether this character is the exact  
2721 sequence transmitted to an output device by the system to accomplish the tabulation.

### 2722 **3.401 Terminal (or Terminal Device)**

2723 A character special file that obeys the specifications of the general terminal interface.

2724 **Note:** The General Terminal Interface is defined in detail in [Chapter 11](#) (on page 199).

### 2725 **3.402 Text Column**

2726 A roughly rectangular block of characters capable of being laid out side-by-side next to other  
2727 text columns on an output page or terminal screen. The widths of text columns are measured in  
2728 column positions.

### 2729 **3.403 Text File**

2730 A file that contains characters organized into zero or more lines. The lines do not contain NUL  
2731 characters and none can exceed {LINE\_MAX} bytes in length, including the <newline>  
2732 character. Although POSIX.1-2008 does not distinguish between text files and binary files (see  
2733 the ISO C standard), many utilities only produce predictable or meaningful output when  
2734 operating on text files. The standard utilities that have such restrictions always specify “text  
2735 files” in their STDIN or INPUT FILES sections.

### 3.404 Thread

A single flow of control within a process. Each thread has its own thread ID, scheduling priority and policy, *errno* value, floating point environment, thread-specific key/value bindings, and the required system resources to support a flow of control. Anything whose address may be determined by a thread, including but not limited to static variables, storage obtained via *malloc()*, directly addressable storage obtained through implementation-defined functions, and automatic variables, are accessible to all threads in the same process.

**Note:** The *malloc()* function is defined in detail in the System Interfaces volume of POSIX.1-2008.

### 3.405 Thread ID

Each thread in a process is uniquely identified during its lifetime by a value of type **pthread\_t** called a thread ID.

### 3.406 Thread List

An ordered set of runnable threads that all have the same ordinal value for their priority.

The ordering of threads on the list is determined by a scheduling policy or policies. The set of thread lists includes all runnable threads in the system.

### 3.407 Thread-Safe

A thread-safe function can be safely invoked concurrently with other calls to the same function, or with calls to any other thread-safe functions, by multiple threads. Each function defined in the System Interfaces volume of POSIX.1-2008 is thread-safe unless explicitly stated otherwise. Examples are any “pure” function, a function which holds a mutex locked while it is accessing static storage, or objects shared among threads.

### 3.408 Thread-Specific Data Key

A process global handle of type **pthread\_key\_t** which is used for naming thread-specific data.

Although the same key value may be used by different threads, the values bound to the key by *pthread\_setspecific()* and accessed by *pthread\_getspecific()* are maintained on a per-thread basis and persist for the life of the calling thread.

**Note:** The *pthread\_getspecific()* and *pthread\_setspecific()* functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

### 3.409 Tilde Character (<tilde>)

The character ‘~’.

### 2766 3.410 Timeouts

2767 A method of limiting the length of time an interface will block; see also [Section 3.76](#) (on page 45).

### 2768 3.411 Timer

2769 A mechanism that can notify a thread when the time as measured by a particular clock has  
2770 reached or passed a specified value, or when a specified amount of time has passed.

### 2771 3.412 Timer Overrun

2772 A condition that occurs each time a timer, for which there is already an expiration signal queued  
2773 to the process, expires.

### 2774 3.413 Token

2775 In the shell command language, a sequence of characters that the shell considers as a single unit  
2776 when reading input. A token is either an operator or a word.

2777 **Note:** The rules for reading input are defined in detail in XCU [Section 2.3](#) (on page 2337).

### 2778 3.414 Trace Analyzer Process

2779 A process that extracts trace events from a trace stream to retrieve information about the  
2780 behavior of an application.

### 2781 3.415 Trace Controller Process

2782 A process that creates a trace stream for tracing a process.

### 2783 3.416 Trace Event

2784 A data object that represents an action executed by the system, and that is recorded in a trace  
2785 stream.

### 2786 3.417 Trace Event Type

2787 A data object type that defines a class of trace event.



**2788 3.418 Trace Event Type Mapping**

2789 A one-to-one mapping between trace event types and trace event names.

**2790 3.419 Trace Filter**

2791 A filter that allows the trace controller process to specify those trace event types that are to be  
2792 ignored; that is, not generated.

**2793 3.420 Trace Generation Version**

2794 A data object that is an implementation-defined character string, generated by the trace system  
2795 and describing the origin and version of the trace system.

**2796 3.421 Trace Log**

2797 The flushed image of a trace stream, if the trace stream is created with a trace log.

**2798 3.422 Trace Point**

2799 An action that may cause a trace event to be generated.

**2800 3.423 Trace Stream**

2801 An opaque object that contains trace events plus internal data needed to interpret those trace  
2802 events.

**2803 3.424 Trace Stream Identifier**

2804 A handle to manage tracing operations in a trace stream.

**2805 3.425 Trace System**

2806 A system that allows both system and user trace events to be generated into a trace stream.  
2807 These trace events can be retrieved later.

**2808 3.426 Traced Process**

2809 A process for which at least one trace stream has been created. A traced process is also called a  
2810 target process.

**2811 3.427 Tracing Status of a Trace Stream**

2812 A status that describes the state of an active trace stream. The tracing status of a trace stream can  
2813 be retrieved from the trace stream attributes. An active trace stream can be in one of two states:  
2814 running or suspended.

**2815 3.428 Typed Memory Name Space**

2816 A system-wide name space that contains the names of the typed memory objects present in the  
2817 system. It is configurable for a given implementation.

**2818 3.429 Typed Memory Object**

2819 A combination of a typed memory pool and a typed memory port. The entire contents of the  
2820 pool are accessible from the port. The typed memory object is identified through a name that  
2821 belongs to the typed memory name space.

**2822 3.430 Typed Memory Pool**

2823 An extent of memory with the same operational characteristics. Typed memory pools may be  
2824 contained within each other.

**2825 3.431 Typed Memory Port**

2826 A hardware access path to one or more typed memory pools.

**2827 3.432 Unbind**

2828 Remove the association between a network address and an endpoint.

**2829 3.433 Unit Data**

2830 See *Datagram* in [Section 3.124](#) (on page 53).

### 3.434 Upshifting

The conversion of a lowercase character that has a single-character uppercase representation into this uppercase representation.

### 3.435 User Database

A system database that contains at least the following information for each user ID:

- User name
- Numerical user ID
- Initial numerical group ID
- Initial working directory
- Initial user program

The initial numerical group ID is used by the *newgrp* utility. Any other circumstances under which the initial values are operative are implementation-defined.

If the initial user program field is null, an implementation-defined program is used.

If the initial working directory field is null, the interpretation of that field is implementation-defined.

**Note:** The *newgrp* utility is defined in detail in the Shell and Utilities volume of POSIX.1-2008.

### 3.436 User ID

A non-negative integer that is used to identify a system user. When the identity of a user is associated with a process, a user ID value is referred to as a real user ID, an effective user ID, or a saved set-user-ID.

### 3.437 User Name

A string that is used to identify a user; see also [Section 3.435](#). To be portable across systems conforming to POSIX.1-2008, the value is composed of characters from the portable filename character set. The <hyphen> character should not be used as the first character of a portable user name.

### 3.438 User Trace Event

A trace event that is generated explicitly by the application as a result of a call to *posix\_trace\_event()*.

### 3.439 Utility

A program, excluding special built-in utilities provided as part of the Shell Command Language, that can be called by name from a shell to perform a specific task, or related set of tasks.

**Note:** For further information on special built-in utilities, see XCU [Section 2.14](#) (on page 2374).

### 3.440 Variable

In the shell command language, a named parameter.

**Note:** For further information, see XCU [Section 2.5](#) (on page 2339).

### 3.441 Vertical-Tab Character (<vertical-tab>)

A character that in the output stream indicates that printing should start at the next vertical tabulation position. It is the character designated by '`\v`' in the C language. If the current position is at or past the last defined vertical tabulation position, the behavior is unspecified. It is unspecified whether this character is the exact sequence transmitted to an output device by the system to accomplish the tabulation.

### 3.442 White Space

A sequence of one or more characters that belong to the **space** character class as defined via the `LC_CTYPE` category in the current locale.

In the POSIX locale, white space consists of one or more <blank> (<space> and <tab> characters), <newline>, <carriage-return>, <form-feed>, and <vertical-tab> characters.

### 3.443 Wide-Character Code (C Language)

An integer value corresponding to a single graphic symbol or control code.

**Note:** C Language Wide-Character Codes are defined in detail in [Section 6.3](#) (on page 129).

### 3.444 Wide-Character Input/Output Functions

The functions that perform wide-oriented input from streams or wide-oriented output to streams: *fgetwc()*, *fgetws()*, *fputwc()*, *fputws()*, *fwprintf()*, *fwscanf()*, *getwc()*, *getwchar()*, *putwc()*, *putwchar()*, *ungetwc()*, *vfwprintf()*, *vfwscanf()*, *vwprintf()*, *vwscanf()*, *wprintf()*, and *wscanf()*.

**Note:** These functions are defined in detail in the System Interfaces volume of POSIX.1-2008.

### 3.445 Wide-Character String

A contiguous sequence of wide-character codes terminated by and including the first null wide-character code.

### 3.446 Word

In the shell command language, a token other than an operator. In some cases a word is also a portion of a word token: in the various forms of parameter expansion, such as *\${name-word}*, and variable assignment, such as *name=word*, the word is the portion of the token depicted by *word*. The concept of a word is no longer applicable following word expansions—only fields remain.

**Note:** For further information, see XCU [Section 2.6.2](#) (on page 2344) and [Section 2.6](#) (on page 2343).

### 3.447 Working Directory (or Current Working Directory)

A directory, associated with a process, that is used in pathname resolution for pathnames that do not begin with a <slash> character.

### 3.448 Worldwide Portability Interface

Functions for handling characters in a codeset-independent manner.

### 3.449 Write

To output characters to a file, such as standard output or standard error. Unless otherwise stated, standard output is the default output destination for all uses of the term “write”; see the distinction between display and write in [Section 3.133](#) (on page 54).

### 3.450 XSI

The X/Open System Interfaces (XSI) option is the core application programming interface for C and *sh* programming for systems conforming to the Single UNIX Specification. This is a

2907 superset of the mandatory requirements for conformance to POSIX.1-2008.

### 2908 **3.451 XSI-Conformant**

2909 A system which allows an application to be built using a set of services that are consistent across  
2910 all systems that conform to POSIX.1-2008 and that support the XSI option.

2911 **Note:** See also [Chapter 2](#) (on page 15).

### 2912 **3.452 Zombie Process**

2913 The remains of a live process (see [Section 3.210](#), on page 66) after it terminates (see [Section 3.303](#),  
2914 on page 82) and before its status information (see XSH [Section 2.13](#), on page 548) is consumed by  
2915 its parent process.

### 2916 **3.453 $\pm 0$**

2917 The algebraic sign provides additional information about any variable that has the value zero  
2918 when the representation allows the sign to be determined.

2919

Chapter 4

2920

# General Concepts

2921

For the purposes of POSIX.1-2008, the general concepts given in [Chapter 4](#) apply.

2922

**Note:** No shading to denote extensions or options occurs in this chapter. Where the terms and definitions given in this chapter are used elsewhere in text related to extensions and options, they are shaded as appropriate.

2924

2925

## 4.1 Concurrent Execution

2926

Functions that suspend the execution of the calling thread shall not cause the execution of other threads to be indefinitely suspended.

2927

2928

## 4.2 Default Initialization

2929

Default initialization causes an object to be initialized according to these rules:

2930

- If it has pointer type, it is initialized to a null pointer.

2931

- If it has arithmetic type, it is initialized to (positive or unsigned) zero.

2932

- If it is an aggregate, every member is initialized (recursively) according to these rules.

2933

- If it is a union, the first named member is initialized (recursively) according to these rules.

2934

For an object of aggregate type with an explicit initializer, the initialization shall occur in initializer list order, each initializer provided for a particular subobject overriding any previously listed initializer for the same subobject; all subobjects that are not initialized explicitly shall be initialized implicitly according to the rules for default initialization.

2935

2936

2937

2938

Objects with static storage duration but no explicit initializer shall be initialized implicitly according to the rules for default initialization.

2939

2940

An explicit initializer of `{ 0 }` works to perform explicit default initialization for any object of scalar or aggregate type, and for any storage duration.

2941

2942

**Notes:**

2943

1. The ISO C standard does not require a compiler to set any field alignment padding bits in a structure or array definition to a particular value. Because of this, a structure initialized using `{ 0 }` might not *memcmp()* as equal to the same structure initialized using *memset()* to zero. For consistent results, portable applications comparing structures should test each field individually.

2944

2945

2946

2947

2948

2. If an implementation treats the all-zero bit pattern of a pointer object as a null pointer, and the all-zero bit pattern of a floating-point object as equivalent to positive 0, then *memset()* to zero and *calloc()* have the same effects as default initialization for all named members of a structure. Implementations that define `__STDC_IEC_559__` guarantee that the all-zero bit pattern of a floating-point object represents 0.0.

2949

2950

2951

MX

2952

### 4.3 Directory Protection

If a directory is writable and the mode bit `S_ISVTX` is set on the directory, a process may remove or rename files within that directory only if one or more of the following is true:

- The effective user ID of the process is the same as that of the owner ID of the file.
- The effective user ID of the process is the same as that of the owner ID of the directory.
- The process has appropriate privileges.
- Optionally, the file is writable by the process. Whether or not files that are writable by the process can be removed or renamed is implementation-defined.

If the `S_ISVTX` bit is set on a non-directory file, the behavior is unspecified.

### 4.4 Extended Security Controls

An implementation may provide implementation-defined extended security controls (see [Section 3.160](#), on page 58). These permit an implementation to provide security mechanisms to implement different security policies than those described in POSIX.1-2008. These mechanisms shall not alter or override the defined semantics of any of the interfaces in POSIX.1-2008.

### 4.5 File Access Permissions

The standard file access control mechanism uses the file permission bits, as described below.

Implementations may provide *additional* or *alternate* file access control mechanisms, or both. An additional access control mechanism shall only further restrict the access permissions defined by the file permission bits. An alternate file access control mechanism shall:

- Specify file permission bits for the file owner class, file group class, and file other class of that file, corresponding to the access permissions.
- Be enabled only by explicit user action, on a per-file basis by the file owner or a user with appropriate privileges.
- Be disabled for a file after the file permission bits are changed for that file with `chmod()`. The disabling of the alternate mechanism need not disable any additional mechanisms supported by an implementation.

Whenever a process requests file access permission for read, write, or execute/search, if no additional mechanism denies access, access shall be determined as follows:

- If a process has appropriate privileges:
  - If read, write, or directory search permission is requested, access shall be granted.
  - If execute permission is requested, access shall be granted if execute permission is granted to at least one user by the file permission bits or by an alternate access control mechanism; otherwise, access shall be denied.
- Otherwise:
  - The file permission bits of a file contain read, write, and execute/search permissions for the file owner class, file group class, and file other class.



2989 — Access shall be granted if an alternate access control mechanism is not enabled and  
 2990 the requested access permission bit is set for the class (file owner class, file group  
 2991 class, or file other class) to which the process belongs, or if an alternate access control  
 2992 mechanism is enabled and it allows the requested access; otherwise, access shall be  
 2993 denied.

## 2994 4.6 File Hierarchy

2995 Files in the system are organized in a hierarchical structure in which all of the non-terminal  
 2996 nodes are directories and all of the terminal nodes are any other type of file. Since multiple  
 2997 directory entries may refer to the same file, the hierarchy is properly described as a “directed  
 2998 graph”.

## 2999 4.7 Filenames

3000 Uppercase and lowercase letters shall retain their unique identities between conforming  
 3001 implementations.

## 3002 4.8 Filename Portability

3003 For a filename to be portable across implementations conforming to POSIX.1-2008, it shall  
 3004 consist only of the portable filename character set as defined in [Section 3.282](#) (on page 79).

3005 **Note:** Applications should avoid using filenames that have the <hyphen-minus> character as the first  
 3006 character since this may cause problems when filenames are passed as command line  
 3007 arguments.

## 3008 4.9 File Times Update

3009 Many operations have requirements to update file timestamps. These requirements do not apply +  
 3010 to streams that have no underlying file description (for example, memory streams created by +  
 3011 `open_memstream()` have no underlying file description). +

3012 Each file has three distinct associated timestamps: the time of last data access, the time of last  
 3013 data modification, and the time the file status last changed. These values are returned in the file  
 3014 characteristics structure **struct stat**, as described in [<sys/stat.h>](#) (on page 391).

3015 Each function or utility in POSIX.1-2008 that reads or writes data (even if the data does not  
 3016 change) or performs an operation to change file status (even if the file status does not change)  
 3017 indicates which of the appropriate timestamps shall be marked for update. If an implementation  
 3018 of such a function or utility marks for update one of these timestamps in a place or time not  
 3019 specified by POSIX.1-2008, this shall be documented, except that any changes caused by  
 3020 pathname resolution need not be documented. For the other functions or utilities in  
 3021 POSIX.1-2008 (those that are not explicitly required to read or write file data or change file  
 3022 status, but that in some implementations happen to do so), the effect is unspecified.

3023 An implementation may update timestamps that are marked for update immediately, or it may

update such timestamps periodically. At the point in time when an update occurs, any marked timestamps shall be set to the current time and the update marks shall be cleared. All timestamps that are marked for update shall be updated when the file ceases to be open by any process or before a *fstat()*, *fstatat()*, *fsync()*, *futimens()*, *lstat()*, *stat()*, *utime()*, *utimensat()*, or *utimes()* is successfully performed on the file. Other times at which updates are done are unspecified. Marks for update, and updates themselves, shall not be done for files on read-only file systems; see [Section 3.310](#) (on page 83).

The resolution of timestamps of files in a file system is implementation-defined, but shall be no coarser than one-second resolution. The three timestamps shall always have values that are supported by the file system. Whenever any of a file's timestamps are to be set to a value *V* according to the rules of the preceding paragraphs of this section, the implementation shall immediately set the timestamp to the greatest value supported by the file system that is not greater than *V*.

## 4.10 Host and Network Byte Orders

When data is transmitted over the network, it is sent as a sequence of octets (8-bit unsigned values). If an entity (such as an address or a port number) can be larger than 8 bits, it needs to be stored in several octets. The convention is that all such values are stored with 8 bits in each octet, and with the first (lowest-addressed) octet holding the most-significant bits. This is called "network byte order".

Network byte order may not be convenient for processing actual values. For this, it is more sensible for values to be stored as ordinary integers. This is known as "host byte order". In host byte order:

- The most significant bit might not be stored in the first byte in address order.
- Bits might not be allocated to bytes in any obvious order at all.

8-bit values stored in `uint8_t` objects do not require conversion to or from host byte order, as they have the same representation. 16 and 32-bit values can be converted using the *htonl()*, *htons()*, *ntohl()*, and *ntohs()* functions. When reading data that is to be converted to host byte order, it should either be received directly into a `uint16_t` or `uint32_t` object or should be copied from an array of bytes using *memcpy()* or similar. Passing the data through other types could cause the byte order to be changed. Similar considerations apply when sending data.

## 4.11 Measurement of Execution Time

The mechanism used to measure execution time shall be implementation-defined. The implementation shall also define to whom the CPU time that is consumed by interrupt handlers and system services on behalf of the operating system will be charged. See [Section 3.118](#) (on page 52).

## 4.12 Memory Synchronization

Applications shall ensure that access to any memory location by more than one thread of control (threads or processes) is restricted such that no thread of control can read or modify a memory location while another thread of control may be modifying it. Such access is restricted using functions that synchronize thread execution and also synchronize memory with respect to other threads. The following functions synchronize memory with respect to other threads:

<code>fork()</code>	<code>pthread_mutex_trylock()</code>	<code>pthread_rwlock_unlock()</code>
<code>pthread_barrier_wait()</code>	<code>pthread_mutex_unlock()</code>	<code>pthread_rwlock_wrlock()</code>
<code>pthread_cond_broadcast()</code>	<code>pthread_spin_lock()</code>	<code>sem_post()</code>
<code>pthread_cond_signal()</code>	<code>pthread_spin_trylock()</code>	<code>sem_timedwait()</code>
<code>pthread_cond_timedwait()</code>	<code>pthread_spin_unlock()</code>	<code>sem_trywait()</code>
<code>pthread_cond_wait()</code>	<code>pthread_rwlock_rdlock()</code>	<code>sem_wait()</code>
<code>pthread_create()</code>	<code>pthread_rwlock_timedrdlock()</code>	<code>semctl()</code>
<code>pthread_join()</code>	<code>pthread_rwlock_timedwrlock()</code>	<code>semop()</code>
<code>pthread_mutex_lock()</code>	<code>pthread_rwlock_tryrdlock()</code>	<code>wait()</code>
<code>pthread_mutex_timedlock()</code>	<code>pthread_rwlock_trywrlock()</code>	<code>waitpid()</code>

The `pthread_once()` function shall synchronize memory for the first call in each thread for a given **pthread\_once\_t** object. If the *init\_routine* called by `pthread_once()` is a cancellation point and is canceled, a call to `pthread_once()` or the same **pthread\_once\_t** object made from a cancellation cleanup handler shall also synchronize memory.

The `pthread_mutex_lock()` function need not synchronize memory if the mutex type is `PTHREAD_MUTEX_RECURSIVE` and the calling thread already owns the mutex. The `pthread_mutex_unlock()` function need not synchronize memory if the mutex type is `PTHREAD_MUTEX_RECURSIVE` and the mutex has a lock count greater than one.

Unless explicitly stated otherwise, if one of the above functions returns an error, it is unspecified whether the invocation causes memory to be synchronized.

Applications may allow more than one thread of control to read a memory location simultaneously.

## 4.13 Pathname Resolution

Pathname resolution is performed for a process to resolve a pathname to a particular directory entry for a file in the file hierarchy. There may be multiple pathnames that resolve to the same directory entry, and multiple directory entries for the same file. When a process resolves a pathname of an existing directory entry, the entire pathname shall be resolved as described below. When a process resolves a pathname of a directory entry that is to be created immediately after the pathname is resolved, pathname resolution terminates when all components of the path prefix of the last component have been resolved. It is then the responsibility of the process to create the final component.

Each filename in the pathname is located in the directory specified by its predecessor (for example, in the pathname fragment **a/b**, file **b** is located in directory **a**). Pathname resolution shall fail if this cannot be accomplished. If the pathname begins with a <slash>, the predecessor of the first filename in the pathname shall be taken to be the root directory of the process (such pathnames are referred to as “absolute pathnames”). If the pathname does not begin with a <slash>, the predecessor of the first filename of the pathname shall be taken to be either the current working directory of the process or for certain interfaces the directory identified by a file descriptor passed to the interface (such pathnames are referred to as “relative pathnames”).

The interpretation of a pathname component is dependent on the value of {NAME\_MAX} and \_POSIX\_NO\_TRUNC associated with the path prefix of that component. If any pathname component is longer than {NAME\_MAX}, the implementation shall consider this an error.

A pathname that contains at least one non-`<slash>` character and that ends with one or more trailing `<slash>` characters shall not be resolved successfully unless the last pathname component before the trailing `<slash>` characters names an existing directory or a directory entry that is to be created for a directory immediately after the pathname is resolved. Interfaces using pathname resolution may specify additional constraints<sup>6</sup> when a pathname that does not name an existing directory contains at least one non-`<slash>` character and contains one or more trailing `<slash>` characters.

If a symbolic link is encountered during pathname resolution, the behavior shall depend on whether the pathname component is at the end of the pathname and on the function being performed. If all of the following are true, then pathname resolution is complete:

1. This is the last pathname component of the pathname.
2. The pathname has no trailing `<slash>`.
3. The function is required to act on the symbolic link itself, or certain arguments direct that the function act on the symbolic link itself.

In all other cases, the system shall prefix the remaining pathname, if any, with the contents of the symbolic link, except that if the contents of the symbolic link is the empty string, then either pathname resolution shall fail with functions reporting an [ENOENT] error and utilities writing an equivalent diagnostic message, or the pathname of the directory containing the symbolic link shall be used in place of the contents of the symbolic link. If the contents of the symbolic link consist solely of `<slash>` characters, then all leading `<slash>` characters of the remaining pathname shall be omitted from the resulting combined pathname, leaving only the leading `<slash>` characters from the symbolic link contents. In the cases where prefixing occurs, if the combined length exceeds {PATH\_MAX}, and the implementation considers this to be an error, pathname resolution shall fail with functions reporting an [ENAMETOOLONG] error and utilities writing an equivalent diagnostic message. Otherwise, the resolved pathname shall be the resolution of the pathname just created. If the resulting pathname does not begin with a `<slash>`, the predecessor of the first filename of the pathname is taken to be the directory containing the symbolic link.

If the system detects a loop in the pathname resolution process, pathname resolution shall fail with functions reporting an [ELOOP] error and utilities writing an equivalent diagnostic message. The same may happen if during the resolution process more symbolic links were followed than the implementation allows. This implementation-defined limit shall not be smaller than {SYMLOOP\_MAX}.

The special filename dot shall refer to the directory specified by its predecessor. The special filename dot-dot shall refer to the parent directory of its predecessor directory. As a special case, in the root directory, dot-dot may refer to the root directory itself.

A pathname consisting of a single `<slash>` shall resolve to the root directory of the process. A null pathname shall not be successfully resolved. If a pathname begins with two successive `<slash>` characters, the first component following the leading `<slash>` characters may be interpreted in an implementation-defined manner, although more than two leading `<slash>` characters shall be treated as a single `<slash>` character.

Pathname resolution for a given pathname shall yield the same results when used by any

6. The only interfaces that further constrain pathnames in POSIX.1-2008 are the `rename()` and `renameat()` functions (see XSH `rename()`) and the `mv` utility (see XCU `mv`).

3151 interface in POSIX.1-2008 as long as there are no changes to any files evaluated during pathname  
3152 resolution for the given pathname between resolutions.

#### 3153 4.14 Process ID Reuse

3154 A process group ID shall not be reused by the system until the process group lifetime ends.

3155 A process ID shall not be reused by the system until the process lifetime ends. In addition, if  
3156 there exists a process group whose process group ID is equal to that process ID, the process ID  
3157 shall not be reused by the system until the process group lifetime ends. A process that is not a  
3158 system process shall not have a process ID of 1.

#### 3159 4.15 Scheduling Policy

3160 A scheduling policy affects process or thread ordering:

- 3161 • When a process or thread is a running thread and it becomes a blocked thread
- 3162 • When a process or thread is a running thread and it becomes a preempted thread
- 3163 • When a process or thread is a blocked thread and it becomes a runnable thread
- 3164 • When a running thread calls a function that can change the priority or scheduling policy of  
3165 a process or thread
- 3166 • In other scheduling policy-defined circumstances

3167 Conforming implementations shall define the manner in which each of the scheduling policies  
3168 may modify the priorities or otherwise affect the ordering of processes or threads at each of the  
3169 occurrences listed above. Additionally, conforming implementations shall define in what other  
3170 circumstances and in what manner each scheduling policy may modify the priorities or affect  
3171 the ordering of processes or threads.

#### 3172 4.16 Seconds Since the Epoch

3173 A value that approximates the number of seconds that have elapsed since the Epoch. A  
3174 Coordinated Universal Time name (specified in terms of seconds (*tm\_sec*), minutes (*tm\_min*),  
3175 hours (*tm\_hour*), days since January 1 of the year (*tm\_yday*), and calendar year minus 1900  
3176 (*tm\_year*)) is related to a time represented as seconds since the Epoch, according to the  
3177 expression below.

3178 If the year is <1970 or the value is negative, the relationship is undefined. If the year is ≥1970 and  
3179 the value is non-negative, the value is related to a Coordinated Universal Time name according  
3180 to the C-language expression, where *tm\_sec*, *tm\_min*, *tm\_hour*, *tm\_yday*, and *tm\_year* are all  
3181 integer types:

3182 
$$tm\_sec + tm\_min*60 + tm\_hour*3600 + tm\_yday*86400 +$$
  
3183 
$$(tm\_year-70)*31536000 + ((tm\_year-69)/4)*86400 -$$
  
3184 
$$((tm\_year-1)/100)*86400 + ((tm\_year+299)/400)*86400$$

3185 The relationship between the actual time of day and the current value for seconds since the  
3186 Epoch is unspecified.

How any changes to the value of seconds since the Epoch are made to align to a desired relationship with the current actual time is implementation-defined. As represented in seconds since the Epoch, each and every day shall be accounted for by exactly 86 400 seconds.

**Note:** The last three terms of the expression add in a day for each year that follows a leap year starting with the first leap year since the Epoch. The first term adds a day every 4 years starting in 1973, the second subtracts a day back out every 100 years starting in 2001, and the third adds a day back in every 400 years starting in 2001. The divisions in the formula are integer divisions; that is, the remainder is discarded leaving only the integer quotient.

## 4.17 Semaphore

A minimum synchronization primitive to serve as a basis for more complex synchronization mechanisms to be defined by the application program.

For the semaphores associated with the Semaphores option, a semaphore is represented as a shareable resource that has a non-negative integer value. When the value is zero, there is a (possibly empty) set of threads awaiting the availability of the semaphore.

For the semaphores associated with the X/Open System Interfaces (XSI) option, a semaphore is a positive integer (0 through 32767). The *semget()* function can be called to create a set or array of semaphores. A semaphore set can contain one or more semaphores up to an implementation-defined value.

### Semaphore Lock Operation

An operation that is applied to a semaphore. If, prior to the operation, the value of the semaphore is zero, the semaphore lock operation shall cause the calling thread to be blocked and added to the set of threads awaiting the semaphore; otherwise, the value shall be decremented.

### Semaphore Unlock Operation

An operation that is applied to a semaphore. If, prior to the operation, there are any threads in the set of threads awaiting the semaphore, then some thread from that set shall be removed from the set and becomes unblocked; otherwise, the semaphore value shall be incremented.

## 4.18 Thread-Safety

Refer to XSH [Section 2.9](#) (on page 512).



## 4.19 Tracing

The trace system allows a traced process to have a selection of events created for it. Traces consist of streams of trace event types.

A trace event type is identified on the one hand by a trace event type name, also referenced as a trace event name, and on the other hand by a trace event type identifier. A trace event name is a human-readable string. A trace event type identifier is an opaque identifier used by the trace system. There shall be a one-to-one relationship between trace event type identifiers and trace event names for a given trace stream and also for a given traced process. The trace event type identifier shall be generated automatically from a trace event name by the trace system either when a trace controller process invokes *posix\_trace\_trid\_eventid\_open()* or when an instrumented application process invokes *posix\_trace\_eventid\_open()*. Trace event type identifiers are used to filter trace event types, to allow interpretation of user data, and to identify the kind of trace point that generated a trace event.

Each trace event shall be of a particular trace event type, and associated with a trace event type identifier. The execution of a trace point shall generate a trace event if a trace stream has been created and started for the process that executed the trace point and if the corresponding trace event type identifier is not ignored by filtering.

A generated trace event shall be recorded in a trace stream, and optionally also in a trace log if a trace log is associated with the trace stream, except that:

- For a trace stream, if no resources are available for the event, the event is lost.
- For a trace log, if no resources are available for the event, or a flush operation does not succeed, the event is lost.

A trace event recorded in an active trace stream may be retrieved by an application having appropriate privileges.

A trace event recorded in a trace log may be retrieved by an application having appropriate privileges after opening the trace log as a pre-recorded trace stream, with the function *posix\_trace\_open()*.

When a trace event is reported it is possible to retrieve the following:

- A trace event type identifier
- A timestamp
- The process ID of the traced process, if the trace event is process-dependent
- Any optional trace event data including its length
- If the Threads option is supported, the thread ID, if the trace event is process-dependent
- The program address at which the trace point was invoked

Trace events may be mapped from trace event types to trace event names. One such mapping shall be associated with each trace stream. An active trace stream is associated with a traced process, and also with its children if the Trace Inherit option is supported and also the inheritance policy is set to *\_POSIX\_TRACE\_INHERIT*. Therefore each traced process has a mapping of the trace event names to trace event type identifiers that have been defined for that process.

Traces can be recorded into either trace streams or trace logs.

The implementation and format of a trace stream are unspecified. A trace stream need not be and generally is not persistent. A trace stream may be either active or pre-recorded:

- An active trace stream is a trace stream that has been created and has not yet been shut down. It can be of one of the two following classes:

1. An active trace stream without a trace log that was created with the *posix\_trace\_create()* function
2. If the Trace Log option is supported, an active trace stream with a trace log that was created with the *posix\_trace\_create\_withlog()* function

- A pre-recorded trace stream is a trace stream that was opened from a trace log object using the *posix\_trace\_open()* function.

An active trace stream can loop. This behavior means that when the resources allocated by the trace system for the trace stream are exhausted, the trace system reuses the resources associated with the oldest recorded trace events to record new trace events.

If the Trace Log option is supported, an active trace stream with a trace log can be flushed. This operation causes the trace system to write trace events from the trace stream to the associated trace log, following the defined policies or using an explicit function call. After this operation, the trace system may reuse the resources associated with the flushed trace events.

An active trace stream with or without a trace log can be cleared. This operation shall cause all the resources associated with this trace stream to be reinitialized. The trace stream shall behave as if it was returning from its creation, except that the mapping of trace event type identifiers to trace event names shall not be cleared. If a trace log was associated with this trace stream, the trace log shall also be reinitialized.

A trace log shall be recorded when the *posix\_trace\_shutdown()* operation is invoked or during tracing, depending on the tracing strategy which is defined by a log policy. After the trace stream has been shut down, the trace information can be retrieved from the associated trace log using the same interface used to retrieve information from an active trace stream.

For a traced process, if the Trace Inherit option is supported and the trace stream's inheritance attribute is `_POSIX_TRACE_INHERIT`, the initial targeted traced process shall be traced together with all of its future children. The *posix\_pid* member of each trace event in a trace stream shall be the process ID of the traced process.

Each trace point may be an implementation-defined action such as a context switch, or an application-programmed action such as a call to a specific operating system service (for example, *fork()*) or a call to *posix\_trace\_event()*.

Trace points may be filtered. The operation of the filter is to filter out (ignore) selected trace events. By default, no trace events are filtered.

The results of the tracing operations can be analyzed and monitored by a trace controller process or a trace analyzer process.

Only the trace controller process has control of the trace stream it has created. The control of the operation of a trace stream is done using its corresponding trace stream identifier. The trace controller process is able to:

- Initialize the attributes of a trace stream
- Create the trace stream
- Start and stop tracing
- Know the mapping of the traced process



- If the Trace Event Filter option is supported, filter the type of trace events to be recorded
- Shut the trace stream down

A traced process may also be a trace controller process. Only the trace controller process can control its trace stream(s). A trace stream created by a trace controller process shall be shut down if its controller process terminates or executes another file.

A trace controller process may also be a trace analyzer process. Trace analysis can be done concurrently with the traced process or can be done off-line, in the same or in a different platform.

## 4.20 Treatment of Error Conditions for Mathematical Functions

For all the functions in the `<math.h>` header, an application wishing to check for error situations should set `errno` to 0 and call `feclearexcept(FE_ALL_EXCEPT)` before calling the function. On return, if `errno` is non-zero or `fetestexcept(FE_INVALID | FE_DIVBYZERO | FE_OVERFLOW | FE_UNDERFLOW)` is non-zero, an error has occurred.

On implementations that support the IEC 60559 Floating-Point option, whether or when functions in the `<math.h>` header raise an undeserved underflow floating-point exception is unspecified. Otherwise, as implied by XSH `feraiseexcept()`, the `<math.h>` functions do not raise spurious floating-point exceptions (detectable by the user), other than the inexact floating-point exception.

The following error conditions are defined for all functions in the `<math.h>` header.

### 4.20.1 Domain Error

A “domain error” shall occur if an input argument is outside the domain over which the mathematical function is defined. The description of each function lists any required domain errors; an implementation may define additional domain errors, provided that such errors are consistent with the mathematical definition of the function.

On a domain error, the function shall return an implementation-defined value; if the integer expression `(math_errhandling & MATH_ERRNO)` is non-zero, `errno` shall be set to [EDOM]; if the integer expression `(math_errhandling & MATH_ERREXCEPT)` is non-zero, the “invalid” floating-point exception shall be raised.

### 4.20.2 Pole Error

A “pole error” occurs if the mathematical result of the function is an exact infinity (for example, `log(0.0)`).

On a pole error, the function shall return the value of the macro `HUGE_VAL`, `HUGE_VALF`, or `HUGE_VALL` according to the return type, with the same sign as the correct value of the function; if the integer expression `(math_errhandling & MATH_ERRNO)` is non-zero, `errno` shall be set to [ERANGE]; if the integer expression `(math_errhandling & MATH_ERREXCEPT)` is non-zero, the “divide-by-zero” floating-point exception shall be raised.

### 4.20.3 Range Error

A “range error” shall occur if the finite mathematical result of the function cannot be represented in an object of the specified type, due to extreme magnitude.

#### 4.20.3.1 Result Overflows

A floating result overflows if the magnitude of the mathematical result is finite but so large that the mathematical result cannot be represented without extraordinary roundoff error in an object of the specified type. If a floating result overflows and default rounding is in effect, then the function shall return the value of the macro `HUGE_VAL`, `HUGE_VALF`, or `HUGE_VALL` according to the return type, with the same sign as the correct value of the function; if the integer expression `(math_errhandling & MATH_ERRNO)` is non-zero, `errno` shall be set to `[ERANGE]`; if the integer expression `(math_errhandling & MATH_ERREXCEPT)` is non-zero, the “overflow” floating-point exception shall be raised.

#### 4.20.3.2 Result Underflows

The result underflows if the magnitude of the mathematical result is so small that the mathematical result cannot be represented, without extraordinary roundoff error, in an object of the specified type. If the result underflows, the function shall return an implementation-defined value whose magnitude is no greater than the smallest normalized positive number in the specified type; if the integer expression `(math_errhandling & MATH_ERRNO)` is non-zero, whether `errno` is set to `[ERANGE]` is implementation-defined; if the integer expression `(math_errhandling & MATH_ERREXCEPT)` is non-zero, whether the “underflow” floating-point exception is raised is implementation-defined.

## 4.21 Treatment of NaN Arguments for the Mathematical Functions

For functions called with a NaN argument, no errors shall occur and a NaN shall be returned, except where stated otherwise.

If a function with one or more NaN arguments returns a NaN result, the result should be the same as one of the NaN arguments (after possible type conversion), except perhaps for the sign.

On implementations that support the IEC 60559:1989 standard floating point, functions with signaling NaN argument(s) shall be treated as if the function were called with an argument that is a required domain error and shall return a quiet NaN result, except where stated otherwise.

**Note:** The function might never see the signaling NaN, since it might trigger when the arguments are evaluated during the function call.

On implementations that support the IEC 60559:1989 standard floating point, for those functions that do not have a documented domain error, the following shall apply:

These functions shall fail if:

Domain Error Any argument is a signaling NaN.

Either, the integer expression `(math_errhandling & MATH_ERRNO)` is non-zero and `errno` shall be set to `[EDOM]`, or the integer expression `(math_errhandling & MATH_ERREXCEPT)` is non-zero and the invalid floating-point exception shall be raised.

## 4.22 Utility

A utility program shall be either an executable file, such as might be produced by a compiler or linker system from computer source code, or a file of shell source code, directly interpreted by the shell. The program may have been produced by the user, provided by the system implementor, or acquired from an independent distributor.

The system may implement certain utilities as shell functions (see XCU [Section 2.9.5](#), on page 2364) or built-in utilities, but only an application that is aware of the command search order (as described in XCU [Section 2.9.1.1](#), on page 2357) or of performance characteristics can discern differences between the behavior of such a function or built-in utility and that of an executable file.

## 4.23 Variable Assignment

In the shell command language, a word consisting of the following parts:

*varname=value*

When used in a context where assignment is defined to occur and at no other time, the *value* (representing a word or field) shall be assigned as the value of the variable denoted by *varname*.

**Note:** For further information, see XCU [Section 2.9.1](#) (on page 2355).

The *varname* and *value* parts shall meet the requirements for a name and a word, respectively, except that they are delimited by the embedded unquoted <equals-sign>, in addition to other delimiters.

**Note:** Additional delimiters are described in XCU [Section 2.3](#) (on page 2337).

When a variable assignment is done, the variable shall be created if it did not already exist. If *value* is not specified, the variable shall be given a null value.

**Note:** An alternative form of variable assignment:

*symbol=value*

(where *symbol* is a valid word delimited by an <equals-sign>, but not a valid name) produces unspecified results. The form *symbol=value* is used by the KornShell *name[expression]=value* syntax.



## File Format Notation

The STDIN, STDOUT, STDERR, INPUT FILES, and OUTPUT FILES sections of the utility descriptions use a syntax to describe the data organization within the files, when that organization is not otherwise obvious. The syntax is similar to that used by the System Interfaces volume of POSIX.1-2008 *printf()* function, as described in this chapter. When used in STDIN or INPUT FILES sections of the utility descriptions, this syntax describes the format that could have been used to write the text to be read, not a format that could be used by the System Interfaces volume of POSIX.1-2008 *scanf()* function to read the input file.

The description of an individual record is as follows:

"<format>", [<arg1>, <arg2>, . . . , <argn>]

The *format* is a character string that contains three types of objects defined below:

1. *Characters* that are not "escape sequences" or "conversion specifications", as described below, shall be copied to the output.
2. *Escape Sequences* represent non-graphic characters and the escape character (<backslash>).
3. *Conversion Specifications* specify the output format of each argument; see below.

The following characters have the following special meaning in the format string:

' ' (An empty character position.) Represents one or more <blank> characters.

Δ Represents exactly one <space> character.

Table 5-1 lists escape sequences and associated actions on display devices capable of the action.

**Table 5-1** Escape Sequences and Associated Actions

Escape Sequence	Represents Character	Terminal Action
\\	<backslash>	Print the <backslash> character.
\a	<alert>	Attempt to alert the user through audible or visible notification.
\b	<backspace>	Move the printing position to one column before the current position, unless the current position is the start of a line.
\f	<form-feed>	Move the printing position to the initial printing position of the next logical page.
\n	<newline>	Move the printing position to the start of the next line.
\r	<carriage-return>	Move the printing position to the start of the current line.
\t	<tab>	Move the printing position to the next tab position on the current line. If there are no more tab positions remaining on the line, the behavior is undefined.
\v	<vertical-tab>	Move the printing position to the start of the next <vertical-tab> position. If there are no more <vertical-tab> positions left on the page, the behavior is undefined.

Each conversion specification is introduced by the <percent-sign> character ('%'). After the character '%', the following shall appear in sequence:

*flags* Zero or more *flags*, in any order, that modify the meaning of the conversion specification.

*field width* An optional string of decimal digits to specify a minimum field width. For an output field, if the converted value has fewer bytes than the field width, it shall be padded on the left (or right, if the left-adjustment flag ('-'), described below, has been given) to the field width.

*precision* Gives the minimum number of digits to appear for the d, o, i, u, x, or X conversion specifiers (the field is padded with leading zeros), the number of digits to appear after the radix character for the e and f conversion specifiers, the maximum number of significant digits for the g conversion specifier; or the maximum number of bytes to be written from a string in the s conversion specifier. The precision shall take the form of a <period> ('.') followed by a decimal digit string; a null digit string is treated as zero.

*conversion specifier characters*

A conversion specifier character (see below) that indicates the type of conversion to be applied.

The *flag* characters and their meanings are:

– The result of the conversion shall be left-justified within the field.

– The result of a signed conversion shall always begin with a sign ('+' or '-').

<space> If the first character of a signed conversion is not a sign, a <space> shall be prefixed to the result. This means that if the <space> and '+' flags both appear, the <space> flag shall be ignored.

# The value shall be converted to an alternative form. For c, d, i, u, and s conversion specifiers, the behavior is undefined. For the o conversion specifier, it shall increase the precision to force the first digit of the result to be a zero. For x or X conversion specifiers, a non-zero result has 0x or 0X prefixed to it, respectively. For a, A, e, E, f, F, g, and G conversion specifiers, the result shall always contain a radix character, even if no digits follow the radix character. For g and G conversion specifiers, trailing zeros shall not be removed from the result as they usually are.

0 For a, A, d, e, E, f, F, g, G, i, o, u, x, and X conversion specifiers, leading zeros (following any indication of sign or base) shall be used to pad to the field width rather than performing space padding, except when converting an infinity or NaN. If the '0' and '-' flags both appear, the '0' flag shall be ignored. For d, i, o, u, x, and X conversion specifiers, if a precision is specified, the '0' flag shall be ignored. For other conversion specifiers, the behavior is undefined.

Each conversion specifier character shall result in fetching zero or more arguments. The results are undefined if there are insufficient arguments for the format. If the format is exhausted while arguments remain, the excess arguments shall be ignored.

The conversion specifiers and their meanings are:

a,A The floating-point number argument representing a floating-point number shall be converted in the style "[–]0xh.hhhhp±d", where there is one hexadecimal digit (which shall be non-zero if the argument is a normalized floating-point number and is otherwise unspecified) before the decimal-point character and the number of hexadecimal digits after it is equal to the precision; if the precision is missing

3484		and FLT_RADIX is a power of 2, then the precision shall be sufficient for an exact
3485		representation of the value; if the precision is missing and FLT_RADIX is not a
3486		power of 2, then the precision shall be sufficient to distinguish different floating-
3487		point values in the internal representation used by the utility, except that trailing
3488		zeros may be omitted; if the precision is zero and the # flag is not specified, no
3489		decimal-point character shall appear. The letters "abcdef" shall be used for a
3490		conversion and the letters "ABCDEF" for A conversion. The A conversion specifier
3491		produces a number with X and P instead of x and p. The exponent shall always
3492		contain at least one digit, and only as many more digits as necessary to represent
3493		the decimal exponent of 2. If the value is zero, the exponent shall be zero. A
3494		floating-point number argument representing an infinity or NaN shall be
3495		converted in the style of an f or F conversion specifier.
3496	d,i,o,u,x,X	The integer argument shall be written as signed decimal (d or i), unsigned octal
3497		(o), unsigned decimal (u), or unsigned hexadecimal notation (x and X). The d and
3498		i specifiers shall convert to signed decimal in the style "[−]dddd". The x
3499		conversion specifier shall use the numbers and letters "0123456789abcdef" and
3500		the X conversion specifier shall use the numbers and letters
3501		"0123456789ABCDEF". The <i>precision</i> component of the argument shall specify
3502		the minimum number of digits to appear. If the value being converted can be
3503		represented in fewer digits than the specified minimum, it shall be expanded with
3504		leading zeros. The default precision shall be 1. The result of converting a zero
3505		value with a precision of 0 shall be no characters. If both the field width and
3506		precision are omitted, the implementation may precede, follow, or precede and
3507		follow numeric arguments of types d, i, and u with <blank> characters; arguments
3508		of type o (octal) may be preceded with leading zeros.
3509	f,F	The floating-point number argument shall be written in decimal notation in the
3510		style [−]ddd.ddd, where the number of digits after the radix character (shown here
3511		as a decimal point) shall be equal to the <i>precision</i> specification. The LC_NUMERIC
3512		locale category shall determine the radix character to use in this format. If the
3513		<i>precision</i> is omitted from the argument, six digits shall be written after the radix
3514		character; if the <i>precision</i> is explicitly 0, no radix character shall appear.
3515		A floating-point number argument representing an infinity shall be converted in
3516		one of the styles "[−]inf" or "[−]infinity"; which style is implementation-
3517		defined. A floating-point number argument representing a NaN shall be converted
3518		in one of the styles "[−]nan( <i>n-char-sequence</i> )" or "[−]nan"; which style,
3519		and the meaning of any <i>n-char-sequence</i> , is implementation-defined. The F
3520		conversion specifier produces "INF", "INFINITY", or "NAN" instead of "inf",
3521		"infinity", or "nan", respectively.
3522	e,E	The floating-point number argument shall be written in the style [−]d.ddde±dd (the
3523		symbol '±' indicates either a <plus-sign> or <hyphen-minus>), where there is one
3524		digit before the radix character (shown here as a decimal point) and the number of
3525		digits after it is equal to the precision. The LC_NUMERIC locale category shall
3526		determine the radix character to use in this format. When the precision is missing,
3527		six digits shall be written after the radix character; if the precision is 0, no radix
3528		character shall appear. The E conversion specifier shall produce a number with E
3529		instead of e introducing the exponent. The exponent shall always contain at least
3530		two digits. However, if the value to be written requires an exponent greater than
3531		two digits, additional exponent digits shall be written as necessary.
3532		A floating-point number argument representing an infinity or NaN shall be
3533		converted in the style of an f or F conversion specifier.



3534	g,G	The floating-point number argument shall be written in style <code>f</code> or <code>e</code> (or in style <code>F</code> or <code>E</code> in the case of a <code>G</code> conversion specifier), with the precision specifying the number of significant digits. The style used depends on the value converted: style <code>e</code> (or <code>E</code> ) shall be used only if the exponent resulting from the conversion is less than $-4$ or greater than or equal to the precision. Trailing zeros are removed from the result. A radix character shall appear only if it is followed by a digit.
3535		
3536		
3537		
3538		
3539		
3540		A floating-point number argument representing an infinity or NaN shall be converted in the style of an <code>f</code> or <code>F</code> conversion specifier.
3541		
3542	c	The single-byte character argument shall be written.
3543	s	The argument shall be taken to be a string and bytes from the string shall be written until the end of the string or the number of bytes indicated by the <i>precision</i> specification of the argument is reached. If the precision is omitted from the argument, it shall be taken to be infinite, so all bytes up to the end of the string shall be written.
3544		
3545		
3546		
3547		
3548	%	Write a ' % ' character; no argument is converted.
3549		In no case does a nonexistent or insufficient field width cause truncation of a field; if the result of a conversion is wider than the field width, the field is simply expanded to contain the conversion result. The term “field width” should not be confused with the term “precision” used in the description of <code>%s</code> .
3550		
3551		
3552		

### Examples

To represent the output of a program that prints a date and time in the form Sunday, July 3, 10:02, where *weekday* and *month* are strings:

```
"%s,%s,%d,%d:%.2d\n" <weekday>, <month>, <day>, <hour>, <min>
```

To show '  $\pi$  ' written to 5 decimal places:

```
"pi=%f\n", <value of  $\pi$ >
```

To show an input file format consisting of five <colon>-separated fields:

```
"%s:%s:%s:%s:%s\n", <arg1>, <arg2>, <arg3>, <arg4>, <arg5>
```



## 6.1 Portable Character Set

Conforming implementations shall support one or more coded character sets. Each supported locale shall include the *portable character set*, which is the set of symbolic names for characters in Table 6-1. This is used to describe characters within the text of POSIX.1-2008. The first eight entries in Table 6-1 and all characters in Table 6-2 (on page 130) are defined in the ISO/IEC 6429:1992 standard. The rest of the characters in Table 6-1 are defined in the ISO/IEC 10646-1:2000 standard.

**Table 6-1** Portable Character Set

Symbolic Name(s)	Glyph	UCS	Description
<NUL>		<U0000>	NULL (NUL)
<alert>, <BEL>		<U0007>	BELL
<backspace>, <BS>		<U0008>	BACKSPACE
<tab>, <HT>		<U0009>	CHARACTER TABULATION
<newline>, <LF>		<U000A>	LINE FEED (LF)
<vertical-tab>, <VT>		<U000B>	LINE TABULATION
<form-feed>, <FF>		<U000C>	FORM FEED (FF)
<carriage-return>, <CR>		<U000D>	CARRIAGE RETURN (CR)
<space>		<U0020>	SPACE
<exclamation-mark>	!	<U0021>	EXCLAMATION MARK
<quotation-mark>	"	<U0022>	QUOTATION MARK
<number-sign>	#	<U0023>	NUMBER SIGN
<dollar-sign>	\$	<U0024>	DOLLAR SIGN
<percent-sign>	%	<U0025>	PERCENT SIGN
<ampersand>	&	<U0026>	AMPERSAND
<apostrophe>	'	<U0027>	APOSTROPHE
<left-parenthesis>	(	<U0028>	LEFT PARENTHESIS
<right-parenthesis>	)	<U0029>	RIGHT PARENTHESIS
<asterisk>	*	<U002A>	ASTERISK
<plus-sign>	+	<U002B>	PLUS SIGN
<comma>	,	<U002C>	COMMA
<hyphen-minus>, <hyphen>	–	<U002D>	HYPHEN-MINUS
<full-stop>, <period>	.	<U002E>	FULL STOP
<slash>, <solidus>	/	<U002F>	SOLIDUS
<zero>	0	<U0030>	DIGIT ZERO
<one>	1	<U0031>	DIGIT ONE
<two>	2	<U0032>	DIGIT TWO
<three>	3	<U0033>	DIGIT THREE
<four>	4	<U0034>	DIGIT FOUR

	Symbolic Name(s)	Glyph	UCS	Description
3601				
3602	<five>	5	<U0035>	DIGIT FIVE
3603	<six>	6	<U0036>	DIGIT SIX
3604	<seven>	7	<U0037>	DIGIT SEVEN
3605	<eight>	8	<U0038>	DIGIT EIGHT
3606	<nine>	9	<U0039>	DIGIT NINE
3607	<colon>	:	<U003A>	COLON
3608	<semicolon>	;	<U003B>	SEMICOLON
3609	<less-than-sign>	<	<U003C>	LESS-THAN SIGN
3610	<equals-sign>	=	<U003D>	EQUALS SIGN
3611	<greater-than-sign>	>	<U003E>	GREATER-THAN SIGN
3612	<question-mark>	?	<U003F>	QUESTION MARK
3613	<commercial-at>	@	<U0040>	COMMERCIAL AT
3614	<A>	A	<U0041>	LATIN CAPITAL LETTER A
3615	<B>	B	<U0042>	LATIN CAPITAL LETTER B
3616	<C>	C	<U0043>	LATIN CAPITAL LETTER C
3617	<D>	D	<U0044>	LATIN CAPITAL LETTER D
3618	<E>	E	<U0045>	LATIN CAPITAL LETTER E
3619	<F>	F	<U0046>	LATIN CAPITAL LETTER F
3620	<G>	G	<U0047>	LATIN CAPITAL LETTER G
3621	<H>	H	<U0048>	LATIN CAPITAL LETTER H
3622	<I>	I	<U0049>	LATIN CAPITAL LETTER I
3623	<J>	J	<U004A>	LATIN CAPITAL LETTER J
3624	<K>	K	<U004B>	LATIN CAPITAL LETTER K
3625	<L>	L	<U004C>	LATIN CAPITAL LETTER L
3626	<M>	M	<U004D>	LATIN CAPITAL LETTER M
3627	<N>	N	<U004E>	LATIN CAPITAL LETTER N
3628	<O>	O	<U004F>	LATIN CAPITAL LETTER O
3629	<P>	P	<U0050>	LATIN CAPITAL LETTER P
3630	<Q>	Q	<U0051>	LATIN CAPITAL LETTER Q
3631	<R>	R	<U0052>	LATIN CAPITAL LETTER R
3632	<S>	S	<U0053>	LATIN CAPITAL LETTER S
3633	<T>	T	<U0054>	LATIN CAPITAL LETTER T
3634	<U>	U	<U0055>	LATIN CAPITAL LETTER U
3635	<V>	V	<U0056>	LATIN CAPITAL LETTER V
3636	<W>	W	<U0057>	LATIN CAPITAL LETTER W
3637	<X>	X	<U0058>	LATIN CAPITAL LETTER X
3638	<Y>	Y	<U0059>	LATIN CAPITAL LETTER Y
3639	<Z>	Z	<U005A>	LATIN CAPITAL LETTER Z
3640	<left-square-bracket>	[	<U005B>	LEFT SQUARE BRACKET
3641	<backslash>, <reverse-solidus>	\	<U005C>	REVERSE SOLIDUS
3642	<right-square-bracket>	]	<U005D>	RIGHT SQUARE BRACKET
3643	<circumflex-accent>, <circumflex>	^	<U005E>	CIRCUMFLEX ACCENT
3644	<low-line>, <underscore>	_	<U005F>	LOW LINE
3645	<grave-accent>	`	<U0060>	GRAVE ACCENT
3646	<a>	a	<U0061>	LATIN SMALL LETTER A
3647	<b>	b	<U0062>	LATIN SMALL LETTER B
3648	<c>	c	<U0063>	LATIN SMALL LETTER C
3649	<d>	d	<U0064>	LATIN SMALL LETTER D
3650	<e>	e	<U0065>	LATIN SMALL LETTER E
3651	<f>	f	<U0066>	LATIN SMALL LETTER F
3652	<g>	g	<U0067>	LATIN SMALL LETTER G

	Symbolic Name(s)	Glyph	UCS	Description
3653				
3654	<h>	h	<U0068>	LATIN SMALL LETTER H
3655	<i>	i	<U0069>	LATIN SMALL LETTER I
3656	<j>	j	<U006A>	LATIN SMALL LETTER J
3657	<k>	k	<U006B>	LATIN SMALL LETTER K
3658	<l>	l	<U006C>	LATIN SMALL LETTER L
3659	<m>	m	<U006D>	LATIN SMALL LETTER M
3660	<n>	n	<U006E>	LATIN SMALL LETTER N
3661	<o>	o	<U006F>	LATIN SMALL LETTER O
3662	<p>	p	<U0070>	LATIN SMALL LETTER P
3663	<q>	q	<U0071>	LATIN SMALL LETTER Q
3664	<r>	r	<U0072>	LATIN SMALL LETTER R
3665	<s>	s	<U0073>	LATIN SMALL LETTER S
3666	<t>	t	<U0074>	LATIN SMALL LETTER T
3667	<u>	u	<U0075>	LATIN SMALL LETTER U
3668	<v>	v	<U0076>	LATIN SMALL LETTER V
3669	<w>	w	<U0077>	LATIN SMALL LETTER W
3670	<x>	x	<U0078>	LATIN SMALL LETTER X
3671	<y>	y	<U0079>	LATIN SMALL LETTER Y
3672	<z>	z	<U007A>	LATIN SMALL LETTER Z
3673	<left-brace>, <left-curly-bracket>	{	<U007B>	LEFT CURLY BRACKET
3674	<vertical-line>		<U007C>	VERTICAL LINE
3675	<right-brace>, <right-curly-bracket>	}	<U007D>	RIGHT CURLY BRACKET
3676	<tilde>	~	<U007E>	TILDE

POSIX.1-2008 uses character names other than the above, but only in an informative way; for example, in examples to illustrate the use of characters beyond the portable character set with the facilities of POSIX.1-2008.

**Table 6-1** (on page 125) defines the characters in the portable character set and the corresponding symbolic character names used to identify each character in a character set description file. Characters defined in **Table 6-2** (on page 130) may also be used in character set description files.

POSIX.1-2008 places only the following requirements on the encoded values of the characters in the portable character set:

- If the encoded values associated with each member of the portable character set are not invariant across all locales supported by the implementation, if an application uses any pair of locales where the character encodings differ, or accesses data from an application using a locale which has different encodings from the locales used by the application, the results are unspecified.
- The encoded values associated with the digits 0 to 9 shall be such that the value of each character after 0 shall be one greater than the value of the previous character.
- A null character, NUL, which has all bits set to zero, shall be in the set of characters.
- The encoded values associated with <period>, <slash>, <newline>, and <carriage-return> shall be invariant across all locales supported by the implementation.
- The encoded values associated with <slash> and <period> shall be invariant across all locales supported by the implementation.
- The encoded values associated with the members of the portable character set are each represented in a single byte. Moreover, if the value is stored in an object of C-language type **char**, it is guaranteed to be positive (except the NUL, which is always zero).

Conforming implementations shall support certain character and character set attributes, as

defined in [Section 7.2](#) (on page 136).

## 6.2 Character Encoding

The POSIX locale shall contain 256 single-byte characters including the characters in [Table 6-1](#) (on page 125) and [Table 6-2](#) (on page 130), which have the properties listed in [Section 7.3.1](#) (on page 139). It is unspecified whether characters not listed in those two tables are classified as **punct** or **cntrl**, or neither. Other locales shall contain the characters in [Table 6-1](#) (on page 125) and may contain any or all of the control characters identified in [Table 6-2](#) (on page 130); the presence, meaning, and representation of any additional characters are locale-specific.

In locales other than the POSIX locale, a character may have a state-dependent encoding. There are two types of these encodings:

- A single-shift encoding (where each character not in the initial shift state is preceded by a shift code) can be defined if each shift-code and character sequence is considered a multi-byte character. This is done using the concatenated-constant format in a character set description file, as described in [Section 6.4](#) (on page 129). If the implementation supports a character encoding of this type, all of the standard utilities in the Shell and Utilities volume of POSIX.1-2008 shall support it. Use of a single-shift encoding with any of the functions in the System Interfaces volume of POSIX.1-2008 that do not specifically mention the effects of state-dependent encoding is implementation-defined.
- A locking-shift encoding (where the state of the character is determined by a shift code that may affect more than the single character following it) cannot be defined with the current character set description file format. Use of a locking-shift encoding with any of the standard utilities in the Shell and Utilities volume of POSIX.1-2008 or with any of the functions in the System Interfaces volume of POSIX.1-2008 that do not specifically mention the effects of state-dependent encoding is implementation-defined.

While in the initial shift state, all characters in the portable character set shall retain their usual interpretation and shall not alter the shift state. The interpretation for subsequent bytes in the sequence shall be a function of the current shift state. A byte with all bits zero shall be interpreted as the null character independent of shift state. Such a byte shall not occur as part of any other character. Likewise, the byte values used to encode <period>, <slash>, <newline>, and <carriage-return> shall not occur as part of any other character in any locale.

The maximum allowable number of bytes in a character in the current locale shall be indicated by {MB\_CUR\_MAX}, defined in the [<stdlib.h>](#) header and by the [<mb\\_cur\\_max>](#) value in a character set description file; see [Section 6.4](#) (on page 129). The implementation's maximum number of bytes in a character shall be defined by the C-language macro {MB\_LEN\_MAX}.

## 6.3 C Language Wide-Character Codes

In the shell, the standard utilities are written so that the encodings of characters are described by the locale's `LC_CTYPE` definition (see [Section 7.3.1](#), on page 139) and there is no differentiation between characters consisting of single octets (8-bit bytes) or multiple bytes. However, in the C language, a differentiation is made. To ease the handling of variable length characters, the C language has introduced the concept of wide-character codes.

All wide-character codes in a given process consist of an equal number of bits. This is in contrast to characters, which can consist of a variable number of bytes. The byte or byte sequence that represents a character can also be represented as a wide-character code. Wide-character codes thus provide a uniform size for manipulating text data. A wide-character code having all bits zero is the null wide-character code (see [Section 3.251](#), on page 73), and terminates wide-character strings (see [Section 3.443](#), on page 104). The wide-character value for each member of the portable character set shall equal its value when used as the lone character in an integer character constant. Wide-character codes for other characters are locale and implementation-defined. State shift bytes shall not have a wide-character code representation. POSIX.1-2008 provides no means of defining a wide-character codeset.

## 6.4 Character Set Description File

Implementations shall provide a character set description file for at least one coded character set supported by the implementation. These files are referred to elsewhere in POSIX.1-2008 as *charmap* files. It is implementation-defined whether or not users or applications can provide additional character set description files.

POSIX.1-2008 does not require that multiple character sets or codesets be supported. Although multiple *charmap* files are supported, it is the responsibility of the implementation to provide the file or files; if only one is provided, only that one is accessible using the *localedef* utility's `-f` option.

Each character set description file, except those that use the ISO/IEC 10646-1:2000 standard position values as the encoding values, shall define characteristics for the coded character set and the encoding for the characters specified in [Table 6-1](#) (on page 125), and may define encoding for additional characters supported by the implementation. Other information about the coded character set may also be in the file. Coded character set character values shall be defined using symbolic character names followed by character encoding values.

Each symbolic name specified in [Table 6-1](#) (on page 125) shall be included in the file. Each character in [Table 6-1](#) (on page 125) (each row in the table) shall be mapped to a unique coding value. For each character in [Table 6-2](#) (on page 130) that exists in the character set described by the file, the character's symbolic name(s) from [Table 6-2](#) (on page 130) and the character's single-byte encoding value shall be included in the file.

3771

Table 6-2 Non-Portable Control Characters

3772

3773

3774

3775

3776

3777

3778

3779

3780

3781

3782

3783

3784

3785

3786

3787

3788

3789

3790

3791

3792

3793

3794

3795

3796

3797

Symbolic Name(s)	UCS	Description
<SOH>	<U0001>	START OF HEADING
<STX>	<U0002>	START OF TEXT
<ETX>	<U0003>	END OF TEXT
<EOT>	<U0004>	END OF TRANSMISSION
<ENQ>	<U0005>	ENQUIRY
<ACK>	<U0006>	ACKNOWLEDGE
<SO>	<U000E>	SHIFT OUT
<SI>	<U000F>	SHIFT IN
<DLE>	<U0010>	DATA LINK ESCAPE
<DC1>	<U0011>	DEVICE CONTROL ONE
<DC2>	<U0012>	DEVICE CONTROL TWO
<DC3>	<U0013>	DEVICE CONTROL THREE
<DC4>	<U0014>	DEVICE CONTROL FOUR
<NAK>	<U0015>	NEGATIVE ACKNOWLEDGE
<SYN>	<U0016>	SYNCHRONOUS IDLE
<ETB>	<U0017>	END OF TRANSMISSION BLOCK
<CAN>	<U0018>	CANCEL
<EM>	<U0019>	END OF MEDIUM
<SUB>	<U001A>	SUBSTITUTE
<ESC>	<U001B>	ESCAPE
<IS4>, <FS>	<U001C>	INFORMATION SEPARATOR FOUR
<IS3>, <GS>	<U001D>	INFORMATION SEPARATOR THREE
<IS2>, <RS>	<U001E>	INFORMATION SEPARATOR TWO
<IS1>, <US>	<U001F>	INFORMATION SEPARATOR ONE
<DEL>	<U007F>	DELETE

3798

3799

3800

The following declarations can precede the character definitions. Each shall consist of the symbol shown in the following list, starting in column 1, including the surrounding brackets, followed by one or more <blank> characters, followed by the value to be assigned to the symbol.

3801

3802

3803

3804

**<code\_set\_name>** The name of the coded character set for which the character set description file is defined. The characters of the name shall be taken from the set of characters with visible glyphs defined in Table 6-1 (on page 125).

3805

3806

**<mb\_cur\_max>** The maximum number of bytes in a multi-byte character. This shall default to 1.

3807

3808

3809

XSI

**<mb\_cur\_min>** An unsigned positive integer value that defines the minimum number of bytes in a character for the encoded character set. On XSI-conformant systems, <mb\_cur\_min> shall always be 1.

3810

3811

3812

3813

**<escape\_char>** The character used to indicate that the characters following shall be interpreted in a special way, as defined later in this section. This shall default to <backslash> ('\\'), which is the character used in all the following text and examples, unless otherwise noted.

3814

3815

3816

**<comment\_char>** The character that, when placed in column 1 of a charmap line, is used to indicate that the line shall be ignored. The default character shall be the <number-sign> ('#').

3817

The character set mapping definitions shall be all the lines immediately following an identifier



line containing the string "CHARMAP" starting in column 1, and preceding a trailer line containing the string "END CHARMAP" starting in column 1. Empty lines and lines containing a **<comment\_char>** in the first column shall be ignored. Each non-comment line of the character set mapping definition (that is, between the "CHARMAP" and "END CHARMAP" lines of the file) shall be in either of two forms:

```
"%s %s %s\n", <symbolic-name>, <encoding>, <comments>
```

or:

```
"%s...%s %s %s\n", <symbolic-name>, <symbolic-name>,  
    <encoding>, <comments>
```

In the first format, the line in the character set mapping definition shall define a single symbolic name and a corresponding encoding. A symbolic name is one or more characters from the set shown with visible glyphs in [Table 6-1](#) (on page 125), enclosed between angle brackets. A character following an escape character is interpreted as itself; for example, the sequence "<\\>" represents the symbolic name ">" enclosed between angle brackets.

In the second format, the line in the character set mapping definition shall define a range of one or more symbolic names. In this form, the symbolic names shall consist of zero or more non-numeric characters from the set shown with visible glyphs in [Table 6-1](#) (on page 125), followed by an integer formed by one or more decimal digits. Both integers shall contain the same number of digits. The characters preceding the integer shall be identical in the two symbolic names, and the integer formed by the digits in the second symbolic name shall be equal to or greater than the integer formed by the digits in the first name. This shall be interpreted as a series of symbolic names formed from the common part and each of the integers between the first and the second integer, inclusive. As an example, <j0101>...<j0104> is interpreted as the symbolic names <j0101>, <j0102>, <j0103>, and <j0104>, in that order.

A character set mapping definition line shall exist for all symbolic names specified in [Table 6-1](#) (on page 125), and shall define the coded character value that corresponds to the character indicated in the table, or the coded character value that corresponds to the control character symbolic name. If the control characters commonly associated with the symbolic names in [Table 6-2](#) (on page 130) are supported by the implementation, the symbolic name and the corresponding encoding value shall be included in the file. Additional unique symbolic names may be included. A coded character value can be represented by more than one symbolic name.

The encoding part is expressed as one (for single-byte character values) or more concatenated decimal, octal, or hexadecimal constants in the following formats:

```
"%cd%u", <escape_char>, <decimal byte value>
```

```
"%cx%x", <escape_char>, <hexadecimal byte value>
```

```
"%co", <escape_char>, <octal byte value>
```

Decimal constants shall be represented by two or three decimal digits, preceded by the escape character and the lowercase letter 'd'; for example, "\d05", "\d97", or "\d143". Hexadecimal constants shall be represented by two hexadecimal digits, preceded by the escape character and the lowercase letter 'x'; for example, "\x05", "\x61", or "\x8f". Octal constants shall be represented by two or three octal digits, preceded by the escape character; for example, "\05", "\141", or "\217". In a portable charmap file, each constant represents an 8-bit byte. When constants are concatenated for multi-byte character values, they shall be of the same type, and interpreted in sequence from first to last with the first byte of the multi-byte character specified by the first byte in the sequence. The manner in which these constants are represented in the character stored in the system is implementation-defined. (This notation was chosen for reasons of portability. There is no requirement that the internal representation in the computer memory be in this same order.) Omitting bytes from a multi-byte character

definition produces undefined results.

In lines defining ranges of symbolic names, the encoded value shall be the value for the first symbolic name in the range (the symbolic name preceding the ellipsis). Subsequent symbolic names defined by the range shall have encoding values in increasing order. Bytes shall be treated as unsigned octets, and carry shall be propagated between the bytes as necessary to represent the range. However, because this causes a null byte in the second or subsequent bytes of a character, such a declaration should not be specified. For example, the line:

```
<j0101>...<j0104> \d129\d254
```

is interpreted as:

```
<j0101>          \d129\d254
<j0102>          \d129\d255
<j0103>          \d130\d00
<j0104>          \d130\d01
```

The expanded declaration of the symbol <j0103> in the above example is an invalid specification, because it contains a null byte in the second byte of a character.

The comment is optional.

POSIX.1-2008 provides no means of defining a wide-character codeset.

The following declarations can follow the character set mapping definitions (after the "END CHARMAP" statement). Each shall consist of the keyword shown in the following list, starting in column 1, followed by the value(s) to be associated to the keyword, as defined below.

**WIDTH** A non-negative integer value defining the column width (see [Section 3.103](#), on page 50) for the printable characters in the coded character set specified in [Table 6-1](#) (on page 125) and [Table 6-2](#) (on page 130). Coded character set character values shall be defined using symbolic character names followed by column width values. Defining a character with more than one **WIDTH** produces undefined results. The **END WIDTH** keyword shall be used to terminate the **WIDTH** definitions. Specifying the width of a non-printable character in a **WIDTH** declaration produces undefined results.

#### **WIDTH\_DEFAULT**

A non-negative integer value defining the default column width for any printable character not listed by one of the **WIDTH** keywords. If no **WIDTH\_DEFAULT** keyword is included in the charmap, the default character width shall be 1.

#### **Example**

After the "END CHARMAP" statement, a syntax for a width definition would be:

```
WIDTH
<A> 1
<B> 1
<C>...<Z> 1
...
<fool>...<foon> 2
...
END WIDTH
```

In this example, the numerical code point values represented by the symbols <A> and <B> are assigned a width of 1. The code point values <C> to <Z> inclusive (<C>, <D>, <E>, and so on) are also assigned a width of 1. Using <A>...<Z> would have required fewer lines, but the



alternative was shown to demonstrate flexibility. The keyword **WIDTH\_DEFAULT** could have been added as appropriate.

#### 6.4.1 State-Dependent Character Encodings

This section addresses the use of state-dependent character encodings (that is, those in which the encoding of a character is dependent on one or more shift codes that may precede it).

A single-shift encoding (where each character not in the initial shift state is preceded by a shift code) can be defined in the charmap format if each shift-code/character sequence is considered a multi-byte character, defined using the concatenated-constant format described in [Section 6.4](#) (on page 129). If the implementation supports a character encoding of this type, all of the standard utilities shall support it. A locking-shift encoding (where the state of the character is determined by a shift code that may affect more than the single character following it) could be defined with an extension to the charmap format described in [Section 6.4](#) (on page 129).

If the implementation supports a character encoding of this type, any of the standard utilities that describe character (*versus* byte) or text-file manipulation shall have the following characteristics:

1. The utility shall process the statefully encoded data as a concatenation of state-independent characters. The presence of redundant locking shifts shall not affect the comparison of two statefully encoded strings.
2. A utility that divides, truncates, or extracts substrings from statefully encoded data shall produce output that contains locking shifts at the beginning or end of the resulting data, if appropriate, to retain correct state information.



## 7.1 General

A locale is the definition of the subset of a user's environment that depends on language and cultural conventions. It is made up from one or more categories. Each category is identified by its name and controls specific aspects of the behavior of components of the system. Category names correspond to the following environment variable names:

*LC\_CTYPE* Character classification and case conversion.

*LC\_COLLATE* Collation order.

*LC\_MONETARY* Monetary formatting.

*LC\_NUMERIC* Numeric, non-monetary formatting.

*LC\_TIME* Date and time formats.

*LC\_MESSAGES* Formats of informative and diagnostic messages and interactive responses.

The standard utilities in the Shell and Utilities volume of POSIX.1-2008 shall base their behavior on the current locale, as defined in the ENVIRONMENT VARIABLES section for each utility. The behavior of some of the C-language functions defined in the System Interfaces volume of POSIX.1-2008 shall also be modified based on a locale selection. The locale to be used by these functions can be selected in the following ways:

1. For functions such as *isalnum\_l()* that take a locale object as an argument, a locale object can be obtained from *newlocale()* or *duplocale()* and passed to the function.
2. For functions that do not take a locale object as an argument, the current locale for the thread can be set by calling *uselocale()* or the global locale for the process can be set by calling *setlocale()*. Such functions shall use the current locale of the calling thread if one has been set for that thread; otherwise, they shall use the global locale.

Locales other than those supplied by the implementation can be created via the *localedef* utility, provided that the *\_POSIX2\_LOCALEDEF* symbol is defined on the system. Even if *localedef* is not provided, all implementations conforming to the System Interfaces volume of POSIX.1-2008 shall provide one or more locales that behave as described in this chapter. The input to the utility is described in [Section 7.3](#) (on page 136). The value that is used to specify a locale when using environment variables shall be the string specified as the *name* operand to the *localedef* utility when the locale was created. The strings "C" and "POSIX" are reserved as identifiers for the POSIX locale (see [Section 7.2](#), on page 136). When the value of a locale environment variable begins with a <slash> (' / '), it shall be interpreted as the pathname of the locale definition; the type of file (regular, directory, and so on) used to store the locale definition is implementation-defined. If the value does not begin with a <slash>, the mechanism used to locate the locale is implementation-defined.

If different character sets are used by the locale categories, the results achieved by an application utilizing these categories are undefined. Likewise, if different codesets are used for the data

being processed by interfaces whose behavior is dependent on the current locale, or the codeset is different from the codeset assumed when the locale was created, the result is also undefined.

Applications can select the desired locale by calling the *newlocale()* or *setlocale()* function with the appropriate value. If the function is invoked with an empty string, such as:

```
newlocale(LC_ALL_MASK, "", (locale_t)0);
```

or:

```
setlocale(LC_ALL, "");
```

the value of the corresponding environment variable is used. If the environment variable is unset or is set to the empty string, the implementation shall set the appropriate environment as defined in [Chapter 8](#) (on page 173).

## 7.2 POSIX Locale

Conforming systems shall provide a POSIX locale, also known as the C locale. In POSIX.1 the requirements for the POSIX locale are more extensive than the requirements for the C locale as specified in the ISO C standard. However, in a conforming POSIX implementation, the POSIX locale and the C locale are identical. The behavior of standard utilities and functions in the POSIX locale shall be as if the locale was defined via the *localedef* utility with input data from the POSIX locale tables in [Section 7.3](#).

For C-language programs, the POSIX locale shall be the default locale when the *setlocale()* function is not called.

The POSIX locale can be specified by assigning to the appropriate environment variables the values "C" or "POSIX".

All implementations shall define a locale as the default locale, to be invoked when no environment variables are set, or set to the empty string. This default locale can be the POSIX locale or any other implementation-defined locale. Some implementations may provide facilities for local installation administrators to set the default locale, customizing it for each location. POSIX.1-2008 does not require such a facility.

## 7.3 Locale Definition

The capability to specify additional locales to those provided by an implementation is optional, denoted by the `_POSIX2_LOCALEDEF` symbol. If the option is not supported, only implementation-supplied locales are available. Such locales shall be documented using the format specified in this section.

Locales can be described with the file format presented in this section. The file format is that accepted by the *localedef* utility. For the purposes of this section, the file is referred to as the "locale definition file", but no locales shall be affected by this file unless it is processed by *localedef* or some similar mechanism. Any requirements in this section imposed upon the utility shall apply to *localedef* or to any other similar utility used to install locale information using the locale definition file format described here.

The locale definition file shall contain one or more locale category source definitions, and shall not contain more than one definition for the same locale category. If the file contains source definitions for more than one category, implementation-defined categories, if present, shall

appear after the categories defined by [Section 7.1](#) (on page 135). A category source definition contains either the definition of a category or a **copy** directive. For a description of the **copy** directive, see *localedef*. In the event that some of the information for a locale category, as specified in this volume of POSIX.1-2008, is missing from the locale source definition, the behavior of that category, if it is referenced, is unspecified.

A category source definition shall consist of a category header, a category body, and a category trailer. A category header shall consist of the character string naming of the category, beginning with the characters *LC\_*. The category trailer shall consist of the string "END", followed by one or more <blank> characters and the string used in the corresponding category header.

The category body shall consist of one or more lines of text. Each line shall contain an identifier, optionally followed by one or more operands. Identifiers shall be either keywords, identifying a particular locale element, or collating elements. In addition to the keywords defined in this volume of POSIX.1-2008, the source can contain implementation-defined keywords. Each keyword within a locale shall have a unique name (that is, two categories cannot have a commonly-named keyword); no keyword shall start with the characters *LC\_*. Identifiers shall be separated from the operands by one or more <blank> characters.

Operands shall be characters, collating elements, or strings of characters. Strings shall be enclosed in double-quotes. Literal double-quotes within strings shall be preceded by the <escape character>, described below. When a keyword is followed by more than one operand, the operands shall be separated by <semicolon> characters; <blank> characters shall be allowed both before and after a <semicolon>.

The first category header in the file can be preceded by a line modifying the comment character. It shall have the following format, starting in column 1:

```
"comment_char %c\n", <comment character>
```

The comment character shall default to the <number-sign> ('#'). Blank lines and lines containing the <comment character> in the first position shall be ignored.

The first category header in the file can be preceded by a line modifying the escape character to be used in the file. It shall have the following format, starting in column 1:

```
"escape_char %c\n", <escape character>
```

The escape character shall default to <backslash>, which is the character used in all examples shown in this volume of POSIX.1-2008.

A line can be continued by placing an escape character as the last character on the line; this continuation character shall be discarded from the input. Although the implementation need not accept any one portion of a continued line with a length exceeding {LINE\_MAX} bytes, it shall place no limits on the accumulated length of the continued line. Comment lines shall not be continued on a subsequent line using an escaped <newline>.

Individual characters, characters in strings, and collating elements shall be represented using symbolic names, as defined below. In addition, characters can be represented using the characters themselves or as octal, hexadecimal, or decimal constants. When non-symbolic notation is used, the resultant locale definitions are in many cases not portable between systems. The left angle bracket ('<') is a reserved symbol, denoting the start of a symbolic name; when used to represent itself it shall be preceded by the escape character. The following rules apply to character representation:

1. A character can be represented via a symbolic name, enclosed within angle brackets '<' and '>'. The symbolic name, including the angle brackets, shall exactly match a symbolic name defined in the charmap file specified via the *localedef* -f option, and it shall be replaced by a character value determined from the value associated with the symbolic

name in the charmap file. The use of a symbolic name not found in the charmap file shall constitute an error, unless the category is *LC\_CTYPE* or *LC\_COLLATE*, in which case it shall constitute a warning condition (see *localedef* for a description of actions resulting from errors and warnings). The specification of a symbolic name in a **collating-element** or **collating-symbol** section that duplicates a symbolic name in the charmap file (if present) shall be an error. Use of the escape character or a right angle bracket within a symbolic name is invalid unless the character is preceded by the escape character.

For example:

```
<c>;<c-cedilla>  "<M><a><y>"
```

2. A character in the portable character set can be represented by the character itself, in which case the value of the character is implementation-defined. (Implementations may allow other characters to be represented as themselves, but such locale definitions are not portable.) Within a string, the double-quote character, the escape character, and the right angle bracket character shall be escaped (preceded by the escape character) to be interpreted as the character itself. Outside strings, the characters:

```
,      ;      <      >      escape_char
```

shall be escaped to be interpreted as the character itself.

For example:

```
c      "May"
```

3. A character can be represented as an octal constant. An octal constant shall be specified as the escape character followed by two or three octal digits. Each constant shall represent a byte value. Multi-byte values can be represented by concatenated constants specified in byte order with the last constant specifying the least significant byte of the character.

For example:

```
\143;\347;\143\150  "\115\141\171"
```

4. A character can be represented as a hexadecimal constant. A hexadecimal constant shall be specified as the escape character followed by an 'x' followed by two hexadecimal digits. Each constant shall represent a byte value. Multi-byte values can be represented by concatenated constants specified in byte order with the last constant specifying the least significant byte of the character.

For example:

```
\x63;\xe7;\x63\x68  "\x4d\x61\x79"
```

5. A character can be represented as a decimal constant. A decimal constant shall be specified as the escape character followed by a 'd' followed by two or three decimal digits. Each constant represents a byte value. Multi-byte values can be represented by concatenated constants specified in byte order with the last constant specifying the least significant byte of the character.

For example:

```
\d99;\d231;\d99\d104  "\d77\d97\d121"
```

Implementations may accept single-digit octal, decimal, or hexadecimal constants following the escape character. Only characters existing in the character set for which the locale definition is created shall be specified, whether using symbolic names, the characters themselves, or octal, decimal, or hexadecimal constants. If a charmap file is present, only characters defined in the charmap can be specified using octal, decimal, or hexadecimal constants. Symbolic names not

present in the charmap file can be specified and shall be ignored, as specified under item 1 above.

### 7.3.1 LC\_CTYPE

The *LC\_CTYPE* category shall define character classification, case conversion, and other character attributes. In addition, a series of characters can be represented by three adjacent <period> characters representing an ellipsis symbol ("..."). The ellipsis specification shall be interpreted as meaning that all values between the values preceding and following it represent valid characters. The ellipsis specification shall be valid only within a single encoded character set; that is, within a group of characters of the same size. An ellipsis shall be interpreted as including in the list all characters with an encoded value higher than the encoded value of the character preceding the ellipsis and lower than the encoded value of the character following the ellipsis.

For example:

```
\x30; ... ;\x39;
```

includes in the character class all characters with encoded values between the endpoints.

The following keywords shall be recognized. In the descriptions, the term “automatically included” means that it shall not be an error either to include or omit any of the referenced characters; the implementation provides them if missing (even if the entire keyword is missing) and accepts them silently if present. When the implementation automatically includes a missing character, it shall have an encoded value dependent on the charmap file in effect (see the description of the *localedef -f* option); otherwise, it shall have a value derived from an implementation-defined character mapping.

The character classes **digit**, **xdigit**, **lower**, **upper**, and **space** have a set of automatically included characters. These only need to be specified if the character values (that is, encoding) differ from the implementation default values. It is not possible to define a locale without these automatically included characters unless some implementation extension is used to prevent their inclusion. Such a definition would not be a proper superset of the C or POSIX locale and, thus, it might not be possible for conforming applications to work properly.

**copy** Specify the name of an existing locale which shall be used as the definition of this category. If this keyword is specified, no other keyword shall be specified.

**upper** Define characters to be classified as uppercase letters.

In the POSIX locale, only:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

shall be included:

In a locale definition file, no character specified for the keywords **cntrl**, **digit**, **punct**, or **space** shall be specified. The uppercase letters <A> to <Z>, as defined in [Section 6.4](#) (on page 129) (the portable character set), are automatically included in this class.

**lower** Define characters to be classified as lowercase letters.

In the POSIX locale, only:

a b c d e f g h i j k l m n o p q r s t u v w x y z

shall be included.



4143		In a locale definition file, no character specified for the keywords <b>cntrl</b> , <b>digit</b> ,
4144		<b>punct</b> , or <b>space</b> shall be specified. The lowercase letters <a> to <z> of the
4145		portable character set are automatically included in this class.
4146	<b>alpha</b>	Define characters to be classified as letters.
4147		In the POSIX locale, only characters in the classes <b>upper</b> and <b>lower</b> shall be
4148		included.
4149		In a locale definition file, no character specified for the keywords <b>cntrl</b> , <b>digit</b> ,
4150		<b>punct</b> , or <b>space</b> shall be specified. Characters classified as either <b>upper</b> or
4151		<b>lower</b> are automatically included in this class.
4152	<b>digit</b>	Define the characters to be classified as numeric digits.
4153		In the POSIX locale, only:
4154		0 1 2 3 4 5 6 7 8 9
4155		shall be included.
4156		In a locale definition file, only the digits <zero>, <one>, <two>, <three>,
4157		<four>, <five>, <six>, <seven>, <eight>, and <nine> shall be specified, and in
4158		contiguous ascending sequence by numerical value. The digits <zero> to
4159		<nine> of the portable character set are automatically included in this class.
4160	<b>alnum</b>	Define characters to be classified as letters and numeric digits. Only the
4161		characters specified for the <b>alpha</b> and <b>digit</b> keywords shall be specified.
4162		Characters specified for the keywords <b>alpha</b> and <b>digit</b> are automatically
4163		included in this class.
4164	<b>space</b>	Define characters to be classified as white-space characters.
4165		In the POSIX locale, exactly <space>, <form-feed>, <newline>, <carriage-
4166		return>, <tab>, and <vertical-tab> shall be included.
4167		In a locale definition file, no character specified for the keywords <b>upper</b> ,
4168		<b>lower</b> , <b>alpha</b> , <b>digit</b> , <b>graph</b> , or <b>xdigit</b> shall be specified. The <space>, <form-
4169		feed>, <newline>, <carriage-return>, <tab>, and <vertical-tab> of the portable
4170		character set, and any characters included in the class <b>blank</b> are automatically
4171		included in this class.
4172	<b>cntrl</b>	Define characters to be classified as control characters.
4173		In the POSIX locale, no characters in classes <b>alpha</b> or <b>punct</b> shall be included.
4174		In a locale definition file, no character specified for the keywords <b>upper</b> ,
4175		<b>lower</b> , <b>alpha</b> , <b>digit</b> , <b>punct</b> , <b>graph</b> , <b>print</b> , or <b>xdigit</b> shall be specified.
4176	<b>punct</b>	Define characters to be classified as punctuation characters.
4177		In the POSIX locale, neither the <space> nor any characters in classes <b>alpha</b> ,
4178		<b>digit</b> , or <b>cntrl</b> shall be included.
4179		In a locale definition file, no character specified for the keywords <b>upper</b> ,
4180		<b>lower</b> , <b>alpha</b> , <b>digit</b> , <b>cntrl</b> , <b>xdigit</b> , or as the <space> shall be specified.
4181	<b>graph</b>	Define characters to be classified as printable characters, not including the
4182		<space>.
4183		In the POSIX locale, all characters in classes <b>alpha</b> , <b>digit</b> , and <b>punct</b> shall be
4184		included; no characters in class <b>cntrl</b> shall be included.



4185		In a locale definition file, characters specified for the keywords <b>upper</b> , <b>lower</b> ,
4186		<b>alpha</b> , <b>digit</b> , <b>xdigit</b> , and <b>punct</b> are automatically included in this class. No
4187		character specified for the keyword <b>cntrl</b> shall be specified.
4188	<b>print</b>	Define characters to be classified as printable characters, including the
4189		<space>.
4190		In the POSIX locale, all characters in class <b>graph</b> shall be included; no
4191		characters in class <b>cntrl</b> shall be included.
4192		In a locale definition file, characters specified for the keywords <b>upper</b> , <b>lower</b> ,
4193		<b>alpha</b> , <b>digit</b> , <b>xdigit</b> , <b>punct</b> , <b>graph</b> , and the <space> are automatically included
4194		in this class. No character specified for the keyword <b>cntrl</b> shall be specified.
4195	<b>xdigit</b>	Define the characters to be classified as hexadecimal digits.
4196		In the POSIX locale, only:
4197		0 1 2 3 4 5 6 7 8 9 A B C D E F a b c d e f
4198		shall be included.
4199		In a locale definition file, only the characters defined for the class <b>digit</b> shall be
4200		specified, in contiguous ascending sequence by numerical value, followed by
4201		one or more sets of six characters representing the hexadecimal digits 10 to 15
4202		inclusive, with each set in ascending order (for example, <A>, <B>, <C>, <D>,
4203		<E>, <F>, <a>, <b>, <c>, <d>, <e>, <f>). The digits <zero> to <nine>, the
4204		uppercase letters <A> to <F>, and the lowercase letters <a> to <f> of the
4205		portable character set are automatically included in this class.
4206	<b>blank</b>	Define characters to be classified as <blank> characters.
4207		In the POSIX locale, only the <space> and <tab> shall be included.
4208		In a locale definition file, the <space> and <tab> are automatically included in
4209		this class.
4210	<b>charclass</b>	Define one or more locale-specific character class names as strings separated
4211		by <semicolon> characters. Each named character class can then be defined
4212		subsequently in the <i>LC_CTYPE</i> definition. A character class name shall consist
4213		of at least one and at most {CHARCLASS_NAME_MAX} bytes of
4214		alphanumeric characters from the portable filename character set. The first
4215		character of a character class name shall not be a digit. The name shall not
4216		match any of the <i>LC_CTYPE</i> keywords defined in this volume of
4217		POSIX.1-2008. Future versions of this standard will not specify any <i>LC_CTYPE</i>
4218		keywords containing uppercase letters.
4219	<i>charclass-name</i>	Define characters to be classified as belonging to the named locale-specific
4220		character class. In the POSIX locale, locale-specific named character classes
4221		need not exist.
4222		If a class name is defined by a <b>charclass</b> keyword, but no characters are
4223		subsequently assigned to it, this is not an error; it represents a class without
4224		any characters belonging to it.
4225		The <i>charclass-name</i> can be used as the <i>property</i> argument to the <i>wctype()</i>
4226		function, in regular expression and shell pattern-matching bracket
4227		expressions, and by the <i>tr</i> command.

4228           **toupper**           Define the mapping of lowercase letters to uppercase letters.

4229                           In the POSIX locale, the 26 lowercase characters: |

4230                           a b c d e f g h i j k l m n o p q r s t u v w x y z

4231                           shall be mapped to the corresponding 26 uppercase characters:

4232                           A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

4233                           In a locale definition file, the operand shall consist of character pairs,  
 4234                           separated by <semicolon> characters. The characters in each character pair  
 4235                           shall be separated by a <comma> and the pair enclosed by parentheses. The  
 4236                           first character in each pair is the lowercase letter, the second the corresponding  
 4237                           uppercase letter. Only characters specified for the keywords **lower** and **upper**  
 4238                           shall be specified. The lowercase letters <a> to <z>, and their corresponding  
 4239                           uppercase letters <A> to <Z>, of the portable character set are automatically  
 4240                           included in this mapping, but only when the **toupper** keyword is omitted  
 4241                           from the locale definition.

4242           **tolower**           Define the mapping of uppercase letters to lowercase letters.

4243                           In the POSIX locale, the 26 uppercase characters: |

4244                           A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

4245                           shall be mapped to the corresponding 26 lowercase characters:

4246                           a b c d e f g h i j k l m n o p q r s t u v w x y z

4247                           In a locale definition file, the operand shall consist of character pairs,  
 4248                           separated by <semicolon> characters. The characters in each character pair  
 4249                           shall be separated by a <comma> and the pair enclosed by parentheses. The  
 4250                           first character in each pair is the uppercase letter, the second the  
 4251                           corresponding lowercase letter. Only characters specified for the keywords  
 4252                           **lower** and **upper** shall be specified. If the **tolower** keyword is omitted from  
 4253                           the locale definition, the mapping is the reverse mapping of the one specified  
 4254                           for **toupper**.

4255           The following table shows the character class combinations allowed:

4256                           **Table 7-1** Valid Character Class Combinations

In Class	Can Also Belong To										
	upper	lower	alpha	digit	space	cntrl	punct	graph	print	xdigit	blank
4259 <b>upper</b>		—	A	x	x	x	x	A	A	—	x
4260 <b>lower</b>	—		A	x	x	x	x	A	A	—	x
4261 <b>alpha</b>	—	—		x	x	x	x	A	A	—	x
4262 <b>digit</b>	x	x	x		x	x	x	A	A	A	x
4263 <b>space</b>	x	x	x	x		—	*	*	*	x	—
4264 <b>cntrl</b>	x	x	x	x	—		x	x	x	x	—
4265 <b>punct</b>	x	x	x	x	—	x		A	A	x	—
4266 <b>graph</b>	—	—	—	—	—	x	—		A	—	—
4267 <b>print</b>	—	—	—	—	—	x	—	—		—	—
4268 <b>xdigit</b>	—	—	—	—	x	x	x	A	A		x
4269 <b>blank</b>	x	x	x	x	A	—	*	*	*	x	

**Notes:**

1. Explanation of codes:
  - A Automatically included; see text.
  - Permitted.
  - x Mutually-exclusive.
  - \* See note 2.
2. The <space>, which is part of the **space** and **blank** classes, cannot belong to **punct** or **graph**, but shall automatically belong to the **print** class. Other **space** or **blank** characters can be classified as any of **punct**, **graph**, or **print**.

## 7.3.1.1 LC\_CTYPE Category in the POSIX Locale

The minimum character classifications for the POSIX locale follow; the code listing depicts the *localedef* input, and the table represents the same information, sorted by character. Implementations may add additional characters to the **cntrl** and **punct** classifications but shall not make any other additions.

```
LC_CTYPE
# The following is the minimum POSIX locale LC_CTYPE.
# "alpha" is by definition "upper" and "lower"
# "alnum" is by definition "alpha" and "digit"
# "print" is by definition "alnum", "punct", and the <space>
# "graph" is by definition "alnum" and "punct"
#
upper    <A>;<B>;<C>;<D>;<E>;<F>;<G>;<H>;<I>;<J>;<K>;<L>;<M>;\
         <N>;<O>;<P>;<Q>;<R>;<S>;<T>;<U>;<V>;<W>;<X>;<Y>;<Z>
#
lower    <a>;<b>;<c>;<d>;<e>;<f>;<g>;<h>;<i>;<j>;<k>;<l>;<m>;\
         <n>;<o>;<p>;<q>;<r>;<s>;<t>;<u>;<v>;<w>;<x>;<y>;<z>
#
digit    <zero>;<one>;<two>;<three>;<four>;<five>;<six>;\
         <seven>;<eight>;<nine>
#
space    <tab>;<newline>;<vertical-tab>;<form-feed>;\
         <carriage-return>;<space>
#
cntrl    <alert>;<backspace>;<tab>;<newline>;<vertical-tab>;\
         <form-feed>;<carriage-return>;\
         <NUL>;<SOH>;<STX>;<ETX>;<EOT>;<ENQ>;<ACK>;<SO>;\
         <SI>;<DLE>;<DC1>;<DC2>;<DC3>;<DC4>;<NAK>;<SYN>;\
         <ETB>;<CAN>;<EM>;<SUB>;<ESC>;<IS4>;<IS3>;<IS2>;\
         <IS1>;<DEL>
#
punct    <exclamation-mark>;<quotation-mark>;<number-sign>;\
         <dollar-sign>;<percent-sign>;<ampersand>;<apostrophe>;\
         <left-parenthesis>;<right-parenthesis>;<asterisk>;\
         <plus-sign>;<comma>;<hyphen-minus>;<period>;<slash>;\
         <colon>;<semicolon>;<less-than-sign>;<equals-sign>;\
         <greater-than-sign>;<question-mark>;<commercial-at>;\
         <left-square-bracket>;<backslash>;<right-square-bracket>;\
```

```

4317         <circumflex>;<underscore>;<grave-accent>;<left-curly-bracket>;\
4318         <vertical-line>;<right-curly-bracket>;<tilde>
4319     #
4320     xdigit    <zero>;<one>;<two>;<three>;<four>;<five>;<six>;<seven>;\
4321             <eight>;<nine>;<A>;<B>;<C>;<D>;<E>;<F>;<a>;<b>;<c>;<d>;<e>;<f>
4322     #
4323     blank    <space>;<tab>
4324     #
4325     toupper   (<a>,<A>);(<b>,<B>);(<c>,<C>);(<d>,<D>);(<e>,<E>);\
4326             (<f>,<F>);(<g>,<G>);(<h>,<H>);(<i>,<I>);(<j>,<J>);\
4327             (<k>,<K>);(<l>,<L>);(<m>,<M>);(<n>,<N>);(<o>,<O>);\
4328             (<p>,<P>);(<q>,<Q>);(<r>,<R>);(<s>,<S>);(<t>,<T>);\
4329             (<u>,<U>);(<v>,<V>);(<w>,<W>);(<x>,<X>);(<y>,<Y>);(<z>,<Z>)
4330     #
4331     tolower   (<A>,<a>);(<B>,<b>);(<C>,<c>);(<D>,<d>);(<E>,<e>);\
4332             (<F>,<f>);(<G>,<g>);(<H>,<h>);(<I>,<i>);(<J>,<j>);\
4333             (<K>,<k>);(<L>,<l>);(<M>,<m>);(<N>,<n>);(<O>,<o>);\
4334             (<P>,<p>);(<Q>,<q>);(<R>,<r>);(<S>,<s>);(<T>,<t>);\
4335             (<U>,<u>);(<V>,<v>);(<W>,<w>);(<X>,<x>);(<Y>,<y>);(<Z>,<z>)
4336     END LC_CTYPE

```

	Symbolic Name	Other Case	Character Classes
4337			
4338	<NUL>		cntrl
4339	<SOH>		cntrl
4340	<STX>		cntrl
4341	<ETX>		cntrl
4342	<EOT>		cntrl
4343	<ENQ>		cntrl
4344	<ACK>		cntrl
4345	<alert>		cntrl
4346	<backspace>		cntrl
4347	<tab>		cntrl, space, blank
4348	<newline>		cntrl, space
4349	<vertical-tab>		cntrl, space
4350	<form-feed>		cntrl, space
4351	<carriage-return>		cntrl, space
4352	<SO>		cntrl
4353	<SI>		cntrl
4354	<DLE>		cntrl
4355	<DC1>		cntrl
4356	<DC2>		cntrl
4357	<DC3>		cntrl
4358	<DC4>		cntrl
4359	<NAK>		cntrl
4360	<SYN>		cntrl
4361	<ETB>		cntrl
4362	<CAN>		cntrl
4363	<EM>		cntrl
4364	<SUB>		cntrl
4365	<ESC>		cntrl
4366	<IS4>		cntrl
4367	<IS3>		cntrl

	Symbolic Name	Other Case	Character Classes
4368	<IS2>		cntrl
4369	<IS1>		cntrl
4370	<space>		space, print, blank
4371	<exclamation-mark>		punct, print, graph
4372	<quotation-mark>		punct, print, graph
4373	<number-sign>		punct, print, graph
4374	<dollar-sign>		punct, print, graph
4375	<percent-sign>		punct, print, graph
4376	<ampersand>		punct, print, graph
4377	<apostrophe>		punct, print, graph
4378	<left-parenthesis>		punct, print, graph
4379	<right-parenthesis>		punct, print, graph
4380	<asterisk>		punct, print, graph
4381	<plus-sign>		punct, print, graph
4382	<comma>		punct, print, graph
4383	<hyphen-minus>		punct, print, graph
4384	<period>		punct, print, graph
4385	<slash>		punct, print, graph
4386	<zero>		digit, xdigit, print, graph
4387	<one>		digit, xdigit, print, graph
4388	<two>		digit, xdigit, print, graph
4389	<three>		digit, xdigit, print, graph
4390	<four>		digit, xdigit, print, graph
4391	<five>		digit, xdigit, print, graph
4392	<six>		digit, xdigit, print, graph
4393	<seven>		digit, xdigit, print, graph
4394	<eight>		digit, xdigit, print, graph
4395	<nine>		digit, xdigit, print, graph
4396	<colon>		punct, print, graph
4397	<semicolon>		punct, print, graph
4398	<less-than-sign>		punct, print, graph
4399	<equals-sign>		punct, print, graph
4400	<greater-than-sign>		punct, print, graph
4401	<question-mark>		punct, print, graph
4402	<commercial-at>		punct, print, graph
4403	<A>	<a>	upper, xdigit, alpha, print, graph
4404	<B>	<b>	upper, xdigit, alpha, print, graph
4405	<C>	<c>	upper, xdigit, alpha, print, graph
4406	<D>	<d>	upper, xdigit, alpha, print, graph
4407	<E>	<e>	upper, xdigit, alpha, print, graph
4408	<F>	<f>	upper, xdigit, alpha, print, graph
4409	<G>	<g>	upper, alpha, print, graph
4410	<H>	<h>	upper, alpha, print, graph
4411	<I>	<i>	upper, alpha, print, graph
4412	<J>	<j>	upper, alpha, print, graph
4413	<K>	<k>	upper, alpha, print, graph
4414	<L>	<l>	upper, alpha, print, graph
4415	<M>	<m>	upper, alpha, print, graph
4416	<N>	<n>	upper, alpha, print, graph
4417	<O>	<o>	upper, alpha, print, graph
4418	<P>	<p>	upper, alpha, print, graph

	Symbolic Name	Other Case	Character Classes
4420			
4421	<Q>	<q>	upper, alpha, print, graph
4422	<R>	<r>	upper, alpha, print, graph
4423	<S>	<s>	upper, alpha, print, graph
4424	<T>	<t>	upper, alpha, print, graph
4425	<U>	<u>	upper, alpha, print, graph
4426	<V>	<v>	upper, alpha, print, graph
4427	<W>	<w>	upper, alpha, print, graph
4428	<X>	<x>	upper, alpha, print, graph
4429	<Y>	<y>	upper, alpha, print, graph
4430	<Z>	<z>	upper, alpha, print, graph
4431	<left-square-bracket>		punct, print, graph
4432	<backslash>		punct, print, graph
4433	<right-square-bracket>		punct, print, graph
4434	<circumflex>		punct, print, graph
4435	<underscore>		punct, print, graph
4436	<grave-accent>		punct, print, graph
4437	<a>	<A>	lower, xdigit, alpha, print, graph
4438	<b>	<B>	lower, xdigit, alpha, print, graph
4439	<c>	<C>	lower, xdigit, alpha, print, graph
4440	<d>	<D>	lower, xdigit, alpha, print, graph
4441	<e>	<E>	lower, xdigit, alpha, print, graph
4442	<f>	<F>	lower, xdigit, alpha, print, graph
4443	<g>	<G>	lower, alpha, print, graph
4444	<h>	<H>	lower, alpha, print, graph
4445	<i>	<I>	lower, alpha, print, graph
4446	<j>	<J>	lower, alpha, print, graph
4447	<k>	<K>	lower, alpha, print, graph
4448	<l>	<L>	lower, alpha, print, graph
4449	<m>	<M>	lower, alpha, print, graph
4450	<n>	<N>	lower, alpha, print, graph
4451	<o>	<O>	lower, alpha, print, graph
4452	<p>	<P>	lower, alpha, print, graph
4453	<q>	<Q>	lower, alpha, print, graph
4454	<r>	<R>	lower, alpha, print, graph
4455	<s>	<S>	lower, alpha, print, graph
4456	<t>	<T>	lower, alpha, print, graph
4457	<u>	<U>	lower, alpha, print, graph
4458	<v>	<V>	lower, alpha, print, graph
4459	<w>	<W>	lower, alpha, print, graph
4460	<x>	<X>	lower, alpha, print, graph
4461	<y>	<Y>	lower, alpha, print, graph
4462	<z>	<Z>	lower, alpha, print, graph
4463	<left-curly-bracket>		punct, print, graph
4464	<vertical-line>		punct, print, graph
4465	<right-curly-bracket>		punct, print, graph
4466	<tilde>		punct, print, graph
4467	<DEL>		cntrl

### 7.3.2 LC\_COLLATE

The *LC\_COLLATE* category provides a collation sequence definition for numerous utilities in the Shell and Utilities volume of POSIX.1-2008 (*ls*, *sort*, and so on), regular expression matching (see Chapter 9, on page 181), and the *strcoll()*, *strxfrm()*, *wcscoll()*, and *wcsxfrm()* functions in the System Interfaces volume of POSIX.1-2008.

A collation sequence definition shall define the relative order between collating elements (characters and multi-character collating elements) in the locale. This order is expressed in terms of collation values; that is, by assigning each element one or more collation values (also known as collation weights). This does not imply that implementations shall assign such values, but that ordering of strings using the resultant collation definition in the locale behaves as if such assignment is done and used in the collation process. At least the following capabilities are provided:

1. **Multi-character collating elements.** Specification of multi-character collating elements (that is, sequences of two or more characters to be collated as an entity).
2. **User-defined ordering of collating elements.** Each collating element shall be assigned a collation value defining its order in the character (or basic) collation sequence. This ordering is used by regular expressions and pattern matching and, unless collation weights are explicitly specified, also as the collation weight to be used in sorting.
3. **Multiple weights and equivalence classes.** Collating elements can be assigned one or more (up to the limit {*COLL\_WEIGHTS\_MAX*}, as defined in *<limits.h>*) collating weights for use in sorting. The first weight is hereafter referred to as the primary weight.
4. **One-to-many mapping.** A single character is mapped into a string of collating elements.
5. **Equivalence class definition.** Two or more collating elements have the same collation value (primary weight).
6. **Ordering by weights.** When two strings are compared to determine their relative order, the two strings are first broken up into a series of collating elements; the elements in each successive pair of elements are then compared according to the relative primary weights for the elements. If equal, and more than one weight has been assigned, then the pairs of collating elements are re-compared according to the relative subsequent weights, until either a pair of collating elements compare unequal or the weights are exhausted.

All implementation-provided locales (either preinstalled or provided as locale definitions which can be installed later) should define a collation sequence that has a total ordering of all characters unless the locale name has an '@' modifier indicating that it has a special collation sequence (for example, @*icase* could indicate that each upper and lowercase character pair collates equally).

#### Notes:

1. A future version of this standard may require these locales to define a collation sequence that has a total ordering of all characters (by changing “should” to “shall”).
2. Users installing their own locales should ensure that they define a collation sequence with a total ordering of all characters unless an '@' modifier in the locale name (such as @*icase*) indicates that it has a special collation sequence.

The following keywords shall be recognized in a collation sequence definition. They are described in detail in the following sections.

**copy** Specify the name of an existing locale which shall be used as the definition of this category. If this keyword is specified, no other keyword shall be specified.



4514	<b>collating-element</b>	Define a collating-element symbol representing a multi-character collating element. This keyword is optional.
4515		
4516	<b>collating-symbol</b>	Define a collating symbol for use in collation order statements. This keyword is optional.
4517		
4518	<b>order_start</b>	Define collation rules. This statement shall be followed by one or more collation order statements, assigning character collation values and collation weights to collating elements.
4519		
4520		
4521	<b>order_end</b>	Specify the end of the collation-order statements.

#### 4522 7.3.2.1 The collating-element Keyword

4523 In addition to the collating elements in the character set, the **collating-element** keyword can be  
 4524 used to define multi-character collating elements. The syntax is as follows:

```
4525 "collating-element %s from \"%s\"\\n", <collating-symbol>, <string>
```

4526 The <collating-symbol> operand shall be a symbolic name, enclosed between angle brackets ('<' and '>'), and shall not duplicate any symbolic name in the current charmap file (if any), or any other symbolic name defined in this collation definition. The string operand is a string of two or more characters that collates as an entity. A <collating-element> defined via this keyword is only recognized with the LC\_COLLATE category.

4531 For example:

```
4532 collating-element <ch> from "<c><h>"
4533 collating-element <e-acute> from "<acute><e>"
4534 collating-element <ll> from "ll"
```

#### 4535 7.3.2.2 The collating-symbol Keyword

4536 This keyword shall be used to define symbols for use in collation sequence statements; that is,  
 4537 between the **order\_start** and the **order\_end** keywords. The syntax is as follows:

```
4538 "collating-symbol %s\\n", <collating-symbol>
```

4539 The <collating-symbol> shall be a symbolic name, enclosed between angle brackets ('<' and '>'), and shall not duplicate any symbolic name in the current charmap file (if any), or any other symbolic name defined in this collation definition. A <collating-symbol> defined via this keyword is only recognized within the LC\_COLLATE category.

4543 For example:

```
4544 collating-symbol <UPPER_CASE>
4545 collating-symbol <HIGH>
```

4546 The **collating-symbol** keyword defines a symbolic name that can be associated with a relative position in the character order sequence. While such a symbolic name does not represent any collating element, it can be used as a weight.



7.3.2.3 The *order\_start* Keyword

The **order\_start** keyword shall precede collation order entries and also define the number of weights for this collation sequence definition and other collation rules. The syntax is as follows:

```
"order_start %s;%s:...;%s\n", <sort-rules>, <sort-rules> ...
```

The operands to the **order\_start** keyword are optional. If present, the operands define rules to be applied when strings are compared. The number of operands define how many weights each element is assigned; if no operands are present, one **forward** operand is assumed. If present, the first operand defines rules to be applied when comparing strings using the first (primary) weight; the second when comparing strings using the second weight, and so on. Operands shall be separated by <semicolon> characters (';'). Each operand shall consist of one or more collation directives, separated by <comma> characters (',''). If the number of operands exceeds the {COLL\_WEIGHTS\_MAX} limit, the utility shall issue a warning message. The following directives shall be supported:

**forward** Specifies that comparison operations for the weight level shall proceed from start of string towards the end of string.

**backward** Specifies that comparison operations for the weight level shall proceed from end of string towards the beginning of string.

**position** Specifies that comparison operations for the weight level shall consider the relative position of elements in the strings not subject to **IGNORE**. The string containing an element not subject to **IGNORE** after the fewest collating elements subject to **IGNORE** from the start of the compare shall collate first. If both strings contain a character not subject to **IGNORE** in the same relative position, the collating values assigned to the elements shall determine the ordering. In case of equality, subsequent characters not subject to **IGNORE** shall be considered in the same manner.

The directives **forward** and **backward** are mutually-exclusive.

If no operands are specified, a single **forward** operand shall be assumed.

For example:

```
order_start    forward;backward
```

## 7.3.2.4 Collation Order

The **order\_start** keyword shall be followed by collating identifier entries. The syntax for the collating element entries is as follows:

```
"%s %s;%s:...;%s\n", <collating-identifier>, <weight>, <weight>, ...
```

Each *collating-identifier* shall consist of either a character (in any of the forms defined in [Section 7.3](#), on page 136), a *collating-element*, a *collating-symbol*, an ellipsis, or the special symbol **UNDEFINED**. The order in which collating elements are specified determines the character order sequence, such that each collating element shall compare less than the elements following it.

A *collating-element* shall be used to specify multi-character collating elements, and indicates that the character sequence specified via the *collating-element* is to be collated as a unit and in the relative order specified by its place.

A *collating-symbol* can be used to define a position in the relative order for use in weights. No weights shall be specified with a *collating-symbol*.

The ellipsis symbol specifies that a sequence of characters shall collate according to their encoded character values. It shall be interpreted as indicating that all characters with a coded character set value higher than the value of the character in the preceding line, and lower than the coded character set value for the character in the following line, in the current coded character set, shall be placed in the character collation order between the previous and the following character in ascending order according to their coded character set values. An initial ellipsis shall be interpreted as if the preceding line specified the NUL character, and a trailing ellipsis as if the following line specified the highest coded character set value in the current coded character set. An ellipsis shall be treated as invalid if the preceding or following lines do not specify characters in the current coded character set. The use of the ellipsis symbol ties the definition to a specific coded character set and may preclude the definition from being portable between implementations.

The symbol **UNDEFINED** shall be interpreted as including all coded character set values not specified explicitly or via the ellipsis symbol. Such characters shall be inserted in the character collation order at the point indicated by the symbol, and in ascending order according to their coded character set values. If no **UNDEFINED** symbol is specified, and the current coded character set contains characters not specified in this section, the utility shall issue a warning message and place such characters at the end of the character collation order.

The optional operands for each collation-element shall be used to define the primary, secondary, or subsequent weights for the collating element. The first operand specifies the relative primary weight, the second the relative secondary weight, and so on. Two or more collation-elements can be assigned the same weight; they belong to the same “equivalence class” if they have the same primary weight. Collation shall behave as if, for each weight level, elements subject to **IGNORE** are removed, unless the **position** collation directive is specified for the corresponding level with the **order\_start** keyword. Then each successive pair of elements shall be compared according to the relative weights for the elements. If the two strings compare equal, the process shall be repeated for the next weight level, up to the limit {**COLL\_WEIGHTS\_MAX**}.

Weights should be assigned such that the collation sequence has a total ordering of all characters unless an '@' modifier in the locale name indicates that it has a special collation sequence.

**Note:** A future version of this standard may require a total ordering of all characters for implementation-provided locales that do not have an '@' modifier in the locale name. See [Section 7.3.2](#) (on page 147).

Weights shall be expressed as characters (in any of the forms specified in [Section 7.3](#), on page 136), *<collating-symbol>*s, *<collating-element>*s, an ellipsis, or the special symbol **IGNORE**. A single character, a *<collating-symbol>*, or a *<collating-element>* shall represent the relative position in the character collating sequence of the character or symbol, rather than the character or characters themselves. Thus, rather than assigning absolute values to weights, a particular weight is expressed using the relative order value assigned to a collating element based on its order in the character collation sequence.

One-to-many mapping is indicated by specifying two or more concatenated characters or symbolic names. For example, if the *<eszet>* is given the string "*<s><s>*" as a weight, comparisons are performed as if all occurrences of the *<eszet>* are replaced by "*<s><s>*" (assuming that "*<s>*" has the collating weight "*<s>*"). If it is necessary to define *<eszet>* and "*<s><s>*" as an equivalence class, then a collating element must be defined for the string "*ss*".

All characters specified via an ellipsis shall by default be assigned unique weights, equal to the relative order of characters. Characters specified via an explicit or implicit **UNDEFINED** special symbol shall by default be assigned the same primary weight (that is, they belong to the same equivalence class) if the collation order has more than one weight level. If the collation order has only one weight level, these characters should be assigned unique primary weights, equal to the relative order of their character in the character collation sequence, but may be assigned the

same primary weight.

**Note:** A future version of this standard may require these characters to be assigned unique primary weights if the collation order has only one weight level.

An ellipsis symbol as a weight shall be interpreted to mean that each character in the sequence shall have unique weights, equal to the relative order of their character in the character collation sequence. The use of the ellipsis as a weight shall be treated as an error if the collating element is neither an ellipsis nor the special symbol **UNDEFINED**.

The special keyword **IGNORE** as a weight shall indicate that when strings are compared using the weights at the level where **IGNORE** is specified, the collating element shall be ignored; that is, as if the string did not contain the collating element. In regular expressions and pattern matching, all characters that are subject to **IGNORE** in their primary weight form an equivalence class.

An empty operand shall be interpreted as the collating element itself.

For example, the order statement:

```
<a>      <a> i <a>
```

is equal to:

```
<a>
```

An ellipsis can be used as an operand if the collating element was an ellipsis, and shall be interpreted as the value of each character defined by the ellipsis.

The collation order as defined in this section affects the interpretation of bracket expressions in regular expressions (see [Section 9.3.5](#), on page 184).

For example:

```
order_start  forward;backward
<LOW>
<space>      <LOW> i <space>
...          <LOW> i ...
<a>          <a> i <a>
<a-acute>    <a> i <a-acute>
<a-grave>    <a> i <a-grave>
<A>          <a> i <A>
<A-acute>    <a> i <A-acute>
<A-grave>    <a> i <A-grave>
<ch>         <ch> i <ch>
<Ch>         <ch> i <Ch>
<s>          <s> i <s>
<eszet>      "<s><s>" i "<eszet><eszet>"
UNDEFINED    IGNORE; ...
order_end
```

This example is interpreted as follows:

1. All characters between <space> and 'a' shall have the same primary equivalence class and individual secondary weights based on their ordinal encoded values.
2. All characters based on the uppercase or lowercase character 'a' belong to the same primary equivalence class.

- 4685           3. The multi-character collating element <ch> is represented by the collating symbol <ch>  
 4686           and belongs to the same primary equivalence class as the multi-character collating  
 4687           element <Ch>.
- 4688           4. The **UNDEFINED** means that all characters not specified in this definition (explicitly or  
 4689           via the ellipsis) shall be ignored when comparing primary weights, and have individual  
 4690           secondary weights based on their ordinal encoded values.

#### 4691   7.3.2.5   The order\_end Keyword

4692           The collating order entries shall be terminated with an **order\_end** keyword.

#### 4693   7.3.2.6   LC\_COLLATE Category in the POSIX Locale

4694           The minimum collation sequence definition of the POSIX locale follows; the code listing depicts  
 4695           the *localedef* input. All characters not explicitly listed here shall be inserted in the character  
 4696           collation order after the listed characters and shall be assigned unique primary weights. If the  
 4697           listed characters have ASCII encoding, the other characters shall be in ascending order according  
 4698           to their coded character set values; otherwise, the order of the other characters is unspecified.  
 4699           The collation sequence shall not include any multi-character collating elements.

```

4700   LC_COLLATE
4701   # This is the minimum input for the POSIX locale definition for the
4702   # LC_COLLATE category. Characters in this list are in the same order
4703   # as in the ASCII codeset.
4704   order_start forward
4705   <NUL>
4706   <SOH>
4707   <STX>
4708   <ETX>
4709   <EOT>
4710   <ENQ>
4711   <ACK>
4712   <alert>
4713   <backspace>
4714   <tab>
4715   <newline>
4716   <vertical-tab>
4717   <form-feed>
4718   <carriage-return>
4719   <SO>
4720   <SI>
4721   <DLE>
4722   <DC1>
4723   <DC2>
4724   <DC3>
4725   <DC4>
4726   <NAK>
4727   <SYN>
4728   <ETB>
4729   <CAN>
4730   <EM>
4731   <SUB>

```

4732	<ESC>
4733	<IS4>
4734	<IS3>
4735	<IS2>
4736	<IS1>
4737	<space>
4738	<exclamation-mark>
4739	<quotation-mark>
4740	<number-sign>
4741	<dollar-sign>
4742	<percent-sign>
4743	<ampersand>
4744	<apostrophe>
4745	<left-parenthesis>
4746	<right-parenthesis>
4747	<asterisk>
4748	<plus-sign>
4749	<comma>
4750	<hyphen-minus>
4751	<period>
4752	<slash>
4753	<zero>
4754	<one>
4755	<two>
4756	<three>
4757	<four>
4758	<five>
4759	<six>
4760	<seven>
4761	<eight>
4762	<nine>
4763	<colon>
4764	<semicolon>
4765	<less-than-sign>
4766	<equals-sign>
4767	<greater-than-sign>
4768	<question-mark>
4769	<commercial-at>
4770	<A>
4771	<B>
4772	<C>
4773	<D>
4774	<E>
4775	<F>
4776	<G>
4777	<H>
4778	<I>
4779	<J>
4780	<K>
4781	<L>
4782	<M>
4783	<N>
4784	<O>

```

4785      <P>
4786      <Q>
4787      <R>
4788      <S>
4789      <T>
4790      <U>
4791      <V>
4792      <W>
4793      <X>
4794      <Y>
4795      <Z>
4796      <left-square-bracket>
4797      <backslash>
4798      <right-square-bracket>
4799      <circumflex>
4800      <underscore>
4801      <grave-accent>
4802      <a>
4803      <b>
4804      <c>
4805      <d>
4806      <e>
4807      <f>
4808      <g>
4809      <h>
4810      <i>
4811      <j>
4812      <k>
4813      <l>
4814      <m>
4815      <n>
4816      <o>
4817      <p>
4818      <q>
4819      <r>
4820      <s>
4821      <t>
4822      <u>
4823      <v>
4824      <w>
4825      <x>
4826      <y>
4827      <z>
4828      <left-curly-bracket>
4829      <vertical-line>
4830      <right-curly-bracket>
4831      <tilde>
4832      <DEL>
4833      order_end
4834      #
4835      END LC_COLLATE

```

### 7.3.3 LC\_MONETARY

The *LC\_MONETARY* category shall define the rules and symbols that are used to format monetary numeric information.

This information is available through the *localeconv()* function and is used by the *strfmon()* function.

Some of the information is also available in an alternative form via the *nl\_langinfo()* function (see CRNCYSTR in [<langinfo.h>](#)).

The following items are defined in this category of the locale. The item names are the keywords recognized by the *localedef* utility when defining a locale. They are also similar to the member names of the *lconv* structure defined in [<locale.h>](#); see [<locale.h>](#) for the exact symbols in the header. The *localeconv()* function returns {CHAR\_MAX} for unspecified integer items and the empty string (" ") for unspecified or size zero string items.

In a locale definition file, the operands are strings, formatted as indicated by the grammar in [Section 7.4](#) (on page 166). For some keywords, the strings can contain only integers. Keywords that are not provided, string values set to the empty string (" "), or integer keywords set to -1, are used to indicate that the value is not available in the locale. The following keywords shall be recognized:

**copy** Specify the name of an existing locale which shall be used as the definition of this category. If this keyword is specified, no other keyword shall be specified.

**Note:** This is a *localedef* utility keyword, unavailable through *localeconv()*.

**int\_curr\_symbol** The international currency symbol. The operand shall be a four-character string, with the first three characters containing the alphabetic international currency symbol. The international currency symbol should be chosen in accordance with those specified in the ISO 4217 standard. The fourth character shall be the character used to separate the international currency symbol from the monetary quantity.

**currency\_symbol** The string that shall be used as the local currency symbol.

**mon\_decimal\_point** The operand is a string containing the symbol that shall be used as the decimal delimiter (radix character) in monetary formatted quantities.

**mon\_thousands\_sep** The operand is a string containing the symbol that shall be used as a separator for groups of digits to the left of the decimal delimiter in formatted monetary quantities.

**mon\_grouping** Define the size of each group of digits in formatted monetary quantities. The operand is a sequence of integers separated by <semicolon> characters. Each integer specifies the number of digits in each group, with the initial integer defining the size of the group immediately preceding the decimal delimiter, and the following integers defining the preceding groups. If the last integer is not -1, then the size of the previous group (if any) shall be repeatedly used for the remainder of the digits. If the last integer is -1, then no further grouping shall be performed.

**positive\_sign** A string that shall be used to indicate a non-negative-valued formatted monetary quantity.

**negative\_sign** A string that shall be used to indicate a negative-valued formatted monetary quantity.



4881	<b>int_frac_digits</b>	An integer representing the number of fractional digits (those to the right of the decimal delimiter) to be written in a formatted monetary quantity using <b>int_curr_symbol</b> .
4882		
4883		
4884	<b>frac_digits</b>	An integer representing the number of fractional digits (those to the right of the decimal delimiter) to be written in a formatted monetary quantity using <b>currency_symbol</b> .
4885		
4886		
4887	<b>p_cs_precedes</b>	An integer set to 1 if the <b>currency_symbol</b> precedes the value for a monetary quantity with a non-negative value, and set to 0 if the symbol succeeds the value.
4888		
4889		
4890	<b>p_sep_by_space</b>	Set to a value indicating the separation of the <b>currency_symbol</b> , the sign string, and the value for a non-negative formatted monetary quantity.
4891		
4892		The values of <b>p_sep_by_space</b> , <b>n_sep_by_space</b> , <b>int_p_sep_by_space</b> , and <b>int_n_sep_by_space</b> are interpreted according to the following:
4893		
4894		0 No <space> separates the currency symbol and value.
4895		1 If the currency symbol and sign string are adjacent, a <space> separates them from the value; otherwise, a <space> separates the currency symbol from the value.
4896		
4897		
4898		2 If the currency symbol and sign string are adjacent, a <space> separates them; otherwise, a <space> separates the sign string from the value.
4899		
4900		
4901	<b>n_cs_precedes</b>	An integer set to 1 if the <b>currency_symbol</b> precedes the value for a monetary quantity with a negative value, and set to 0 if the symbol succeeds the value.
4902		
4903		
4904	<b>n_sep_by_space</b>	Set to a value indicating the separation of the <b>currency_symbol</b> , the sign string, and the value for a negative formatted monetary quantity.
4905		
4906	<b>p_sign_posn</b>	An integer set to a value indicating the positioning of the <b>positive_sign</b> for a monetary quantity with a non-negative value. The following integer values shall be recognized for <b>int_n_sign_posn</b> , <b>int_p_sign_posn</b> , <b>n_sign_posn</b> , and <b>p_sign_posn</b> :
4907		
4908		
4909		
4910		0 Parentheses enclose the quantity and the <b>currency_symbol</b> .
4911		1 The sign string precedes the quantity and the <b>currency_symbol</b> .
4912		2 The sign string succeeds the quantity and the <b>currency_symbol</b> .
4913		3 The sign string precedes the <b>currency_symbol</b> .
4914		4 The sign string succeeds the <b>currency_symbol</b> .
4915	<b>n_sign_posn</b>	An integer set to a value indicating the positioning of the <b>negative_sign</b> for a negative formatted monetary quantity.
4916		
4917	<b>int_p_cs_precedes</b>	An integer set to 1 if the <b>int_curr_symbol</b> precedes the value for a monetary quantity with a non-negative value, and set to 0 if the symbol succeeds the value.
4918		
4919		
4920	<b>int_n_cs_precedes</b>	An integer set to 1 if the <b>int_curr_symbol</b> precedes the value for a monetary quantity with a negative value, and set to 0 if the symbol succeeds the value.
4921		
4922		



4923	<b>int_p_sep_by_space</b>	Set to a value indicating the separation of the <b>int_curr_symbol</b> , the sign string, and the value for a non-negative internationally formatted monetary quantity.
4924		
4925		
4926	<b>int_n_sep_by_space</b>	Set to a value indicating the separation of the <b>int_curr_symbol</b> , the sign string, and the value for a negative internationally formatted monetary quantity.
4927		
4928		
4929	<b>int_p_sign_posn</b>	An integer set to a value indicating the positioning of the <b>positive_sign</b> for a positive monetary quantity formatted with the international format.
4930		
4931	<b>int_n_sign_posn</b>	An integer set to a value indicating the positioning of the <b>negative_sign</b> for a negative monetary quantity formatted with the international format.
4932		

### 4933 7.3.3.1 LC\_MONETARY Category in the POSIX Locale

4934 The monetary formatting definitions for the POSIX locale follow; the code listing depicting the  
 4935 *localedef* input, the table representing the same information with the addition of *localeconv()* and  
 4936 *nl\_langinfo()* formats. All values are unspecified in the POSIX locale.

```

4937 LC_MONETARY
4938 # This is the POSIX locale definition for
4939 # the LC_MONETARY category.
4940 #
4941 int_curr_symbol      " "
4942 currency_symbol      " "
4943 mon_decimal_point    " "
4944 mon_thousands_sep   " "
4945 mon_grouping         -1
4946 positive_sign        " "
4947 negative_sign        " "
4948 int_frac_digits      -1
4949 frac_digits          -1
4950 p_cs_precedes        -1
4951 p_sep_by_space       -1
4952 n_cs_precedes        -1
4953 n_sep_by_space       -1
4954 p_sign_posn          -1
4955 n_sign_posn          -1
4956 int_p_cs_precedes    -1
4957 int_p_sep_by_space   -1
4958 int_n_cs_precedes    -1
4959 int_n_sep_by_space   -1
4960 int_p_sign_posn      -1
4961 int_n_sign_posn      -1
4962 #
4963 END LC_MONETARY

```

	Item	langinfo Constant	POSIX Locale Value	localeconv() Value	localedef Value
4964	<b>int_curr_symbol</b>	—	N/A	" "	" "
4965	<b>currency_symbol</b>	CRNCYSTR	N/A	" "	" "
4966	<b>mon_decimal_point</b>	—	N/A	" "	" "
4967	<b>mon_thousands_sep</b>	—	N/A	" "	" "
4968	<b>mon_grouping</b>	—	N/A	" "	–1
4969	<b>positive_sign</b>	—	N/A	" "	" "
4970	<b>negative_sign</b>	—	N/A	" "	" "
4971	<b>int_frac_digits</b>	—	N/A	{CHAR_MAX}	–1
4972	<b>frac_digits</b>	—	N/A	{CHAR_MAX}	–1
4973	<b>p_cs_precedes</b>	CRNCYSTR	N/A	{CHAR_MAX}	–1
4974	<b>p_sep_by_space</b>	—	N/A	{CHAR_MAX}	–1
4975	<b>n_cs_precedes</b>	CRNCYSTR	N/A	{CHAR_MAX}	–1
4976	<b>n_sep_by_space</b>	—	N/A	{CHAR_MAX}	–1
4977	<b>p_sign_posn</b>	—	N/A	{CHAR_MAX}	–1
4978	<b>n_sign_posn</b>	—	N/A	{CHAR_MAX}	–1
4979	<b>int_p_cs_precedes</b>	—	N/A	{CHAR_MAX}	–1
4980	<b>int_p_sep_by_space</b>	—	N/A	{CHAR_MAX}	–1
4981	<b>int_n_cs_precedes</b>	—	N/A	{CHAR_MAX}	–1
4982	<b>int_n_sep_by_space</b>	—	N/A	{CHAR_MAX}	–1
4983	<b>int_p_sign_posn</b>	—	N/A	{CHAR_MAX}	–1
4984	<b>int_n_sign_posn</b>	—	N/A	{CHAR_MAX}	–1

The entry N/A indicates that the value is not available in the POSIX locale.

### 7.3.4 LC\_NUMERIC

The *LC\_NUMERIC* category shall define the rules and symbols that are used to format non-monetary numeric information. This information is available through the *localeconv()* function.

Some of the information is also available in an alternative form via the *nl\_langinfo()* function.

The following items are defined in this category of the locale. The item names are the keywords recognized by the *localedef* utility when defining a locale. They are also similar to the member names of the **lconv** structure defined in [<locale.h>](#); see [<locale.h>](#) for the exact symbols in the header. The *localeconv()* function returns {CHAR\_MAX} for unspecified integer items and the empty string (" ") for unspecified or size zero string items.

In a locale definition file, the operands are strings, formatted as indicated by the grammar in [Section 7.4](#) (on page 166). For some keywords, the strings can only contain integers. Keywords that are not provided, string values set to the empty string (" "), or integer keywords set to –1, shall be used to indicate that the value is not available in the locale. The following keywords shall be recognized:

**copy** Specify the name of an existing locale which shall be used as the definition of this category. If this keyword is specified, no other keyword shall be specified.

**Note:** This is a *localedef* utility keyword, unavailable through *localeconv()*.

**decimal\_point** The operand is a string containing the symbol that shall be used as the decimal delimiter (radix character) in numeric, non-monetary formatted quantities. This keyword cannot be omitted and cannot be set to the empty string. In contexts where standards limit the **decimal\_point** to a single byte, the result of specifying a multi-byte operand shall be unspecified.

5010       **thousands\_sep**   The operand is a string containing the symbol that shall be used as a separator  
 5011                           for groups of digits to the left of the decimal delimiter in numeric, non-  
 5012                           monetary formatted monetary quantities. In contexts where standards limit  
 5013                           the **thousands\_sep** to a single byte, the result of specifying a multi-byte  
 5014                           operand shall be unspecified.

5015       **grouping**        Define the size of each group of digits in formatted non-monetary quantities.  
 5016                           The operand is a sequence of integers separated by <semicolon> characters.  
 5017                           Each integer specifies the number of digits in each group, with the initial  
 5018                           integer defining the size of the group immediately preceding the decimal  
 5019                           delimiter, and the following integers defining the preceding groups. If the last  
 5020                           integer is not -1, then the size of the previous group (if any) shall be  
 5021                           repeatedly used for the remainder of the digits. If the last integer is -1, then no  
 5022                           further grouping shall be performed.

#### 5023 7.3.4.1 LC\_NUMERIC Category in the POSIX Locale

5024       The non-monetary numeric formatting definitions for the POSIX locale follow; the code listing  
 5025       depicting the *localedef* input, the table representing the same information with the addition of  
 5026       *localeconv()* values, and *nl\_langinfo()* constants.

```
5027 LC_NUMERIC
5028 # This is the POSIX locale definition for
5029 # the LC_NUMERIC category.
5030 #
5031 decimal_point      "<period>"
5032 thousands_sep      " "
5033 grouping           -1
5034 #
5035 END LC_NUMERIC
```

Item	langinfo Constant	POSIX Locale Value	localeconv() Value	localedef Value
<b>decimal_point</b>	RADIXCHAR	"."	"."	.
<b>thousands_sep</b>	THOUSEP	N/A	" "	" "
<b>grouping</b>	—	N/A	" "	-1

5041       The entry N/A indicates that the value is not available in the POSIX locale.

### 5042 7.3.5 LC\_TIME

5043       The *LC\_TIME* category shall define the interpretation of the conversion specifications supported  
 5044       by the *date* utility and shall affect the behavior of the *strftime()*, *wcsftime()*, *strptime()*, and  
 5045       *nl\_langinfo()* functions. Since the interfaces for C-language access and locale definition differ  
 5046       significantly, they are described separately.

#### 5047 7.3.5.1 LC\_TIME Locale Definition

5048       In a locale definition, the following mandatory keywords shall be recognized:

5049       **copy**           Specify the name of an existing locale which shall be used as the definition of  
 5050                           this category. If this keyword is specified, no other keyword shall be specified.

5051	<b>abday</b>	Define the abbreviated weekday names, corresponding to the %a conversion specification (conversion specification in the <i>strptime()</i> , <i>wcsftime()</i> , and <i>strptime()</i> functions). The operand shall consist of seven
5052		<semicolon>-separated strings, each surrounded by double-quotes. The first
5053		string shall be the abbreviated name of the day corresponding to Sunday, the
5054		second the abbreviated name of the day corresponding to Monday, and so on.
5055		
5056		
5057	<b>day</b>	Define the full weekday names, corresponding to the %A conversion
5058		specification. The operand shall consist of seven <semicolon>-separated
5059		strings, each surrounded by double-quotes. The first string is the full name of
5060		the day corresponding to Sunday, the second the full name of the day
5061		corresponding to Monday, and so on.
5062	<b>abmon</b>	Define the abbreviated month names, corresponding to the %b conversion
5063		specification. The operand shall consist of twelve <semicolon>-separated
5064		strings, each surrounded by double-quotes. The first string shall be the
5065		abbreviated name of the first month of the year (January), the second the
5066		abbreviated name of the second month, and so on.
5067	<b>mon</b>	Define the full month names, corresponding to the %B conversion
5068		specification. The operand shall consist of twelve <semicolon>-separated
5069		strings, each surrounded by double-quotes. The first string shall be the full
5070		name of the first month of the year (January), the second the full name of the
5071		second month, and so on.
5072	<b>d_t_fmt</b>	Define the appropriate date and time representation, corresponding to the %c
5073		conversion specification. The operand shall consist of a string containing any
5074		combination of characters and conversion specifications. In addition, the
5075		string can contain escape sequences defined in the table in <a href="#">Table 5-1</a> (on page
5076		121) ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v').
5077	<b>d_fmt</b>	Define the appropriate date representation, corresponding to the %x
5078		conversion specification. The operand shall consist of a string containing any
5079		combination of characters and conversion specifications. In addition, the
5080		string can contain escape sequences defined in <a href="#">Table 5-1</a> (on page 121).
5081	<b>t_fmt</b>	Define the appropriate time representation, corresponding to the %X
5082		conversion specification. The operand shall consist of a string containing any
5083		combination of characters and conversion specifications. In addition, the
5084		string can contain escape sequences defined in <a href="#">Table 5-1</a> (on page 121).
5085	<b>am_pm</b>	Define the appropriate representation of the <i>ante-meridiem</i> and <i>post-meridiem</i>
5086		strings, corresponding to the %p conversion specification. The operand shall
5087		consist of two strings, separated by a <semicolon>, each surrounded by
5088		double-quotes. The first string shall represent the <i>ante-meridiem</i> designation,
5089		the last string the <i>post-meridiem</i> designation.
5090	<b>t_fmt_ampm</b>	Define the appropriate time representation in the 12-hour clock format with
5091		<b>am_pm</b> , corresponding to the %r conversion specification. The operand shall
5092		consist of a string and can contain any combination of characters and
5093		conversion specifications. If the string is empty, the 12-hour format is not
5094		supported in the locale.
5095	<b>era</b>	Define how years are counted and displayed for each era in a locale. The
5096		operand shall consist of <semicolon>-separated strings. Each string shall be an
5097		era description segment with the format:
5098		<i>direction:offset:start_date:end_date:era_name:era_format</i>

5099		according to the definitions below. There can be as many era description
5100		segments as are necessary to describe the different eras.
5101	<b>Note:</b>	The start of an era might not be the earliest point in the era—it may be the
5102		latest. For example, the Christian era BC starts on the day before January 1,
5103		AD 1, and increases with earlier time.
5104	<i>direction</i>	Either a '+' or a '-' character. The '+' character shall indicate
5105		that years closer to the <i>start_date</i> have lower numbers than those
5106		closer to the <i>end_date</i> . The '-' character shall indicate that years
5107		closer to the <i>start_date</i> have higher numbers than those closer to
5108		the <i>end_date</i> .
5109	<i>offset</i>	The number of the year closest to the <i>start_date</i> in the era,
5110		corresponding to the %Ey conversion specification.
5111	<i>start_date</i>	A date in the form <i>yyyy/mm/dd</i> , where <i>yyyy</i> , <i>mm</i> , and <i>dd</i> are the
5112		year, month, and day numbers respectively of the start of the era.
5113		Years prior to AD 1 shall be represented as negative numbers.
5114	<i>end_date</i>	The ending date of the era, in the same format as the <i>start_date</i> ,
5115		or one of the two special values "-*" or "+*". The value "-*" shall
5116		indicate that the ending date is the beginning of time. The
5117		value "+*" shall indicate that the ending date is the end of time.
5118	<i>era_name</i>	A string representing the name of the era, corresponding to the
5119		%EC conversion specification.
5120	<i>era_format</i>	A string for formatting the year in the era, corresponding to the
5121		%EY conversion specification.
5122	<b>era_d_fmt</b>	Define the format of the date in alternative era notation, corresponding to the
5123		%Ex conversion specification.
5124	<b>era_t_fmt</b>	Define the locale's appropriate alternative time format, corresponding to the
5125		%EX conversion specification.
5126	<b>era_d_t_fmt</b>	Define the locale's appropriate alternative date and time format,
5127		corresponding to the %Ec conversion specification.
5128	<b>alt_digits</b>	Define alternative symbols for digits, corresponding to the %O modified
5129		conversion specification. The operand shall consist of <semicolon>-separated
5130		strings, each surrounded by double-quotes. The first string shall be the
5131		alternative symbol corresponding with zero, the second string the symbol
5132		corresponding with one, and so on. Up to 100 alternative symbol strings can
5133		be specified. The %O modifier shall indicate that the string corresponding to
5134		the value specified via the conversion specification shall be used instead of the
5135		value.
5136	7.3.5.2	<i>LC_TIME</i> C-Language Access
5137		The following constants used to identify items of <i>langinfo</i> data can be used as arguments to the
5138		<i>nl_langinfo()</i> function to access information in the <i>LC_TIME</i> category. These constants are
5139		defined in the <b>&lt;langinfo.h&gt;</b> header.
5140	<b>ABDAY_x</b>	The abbreviated weekday names (for example, Sun), where <i>x</i> is a number
5141		from 1 to 7.

5142	DAY_x	The full weekday names (for example, Sunday), where <i>x</i> is a number from 1 to 7.
5143		
5144	ABMON_x	The abbreviated month names (for example, Jan), where <i>x</i> is a number from 1 to 12.
5145		
5146	MON_x	The full month names (for example, January), where <i>x</i> is a number from 1 to 12.
5147		
5148	D_T_FMT	The appropriate date and time representation.
5149	D_FMT	The appropriate date representation.
5150	T_FMT	The appropriate time representation.
5151	AM_STR	The appropriate ante-meridiem affix.
5152	PM_STR	The appropriate post-meridiem affix.
5153	T_FMT_AMPM	The appropriate time representation in the 12-hour clock format with AM_STR and PM_STR.
5154		
5155	ERA	The era description segments, which describe how years are counted and displayed for each era in a locale. Each era description segment shall have the format:
5156		
5157		
5158		<i>direction:offset:start_date:end_date:era_name:era_format</i>
5159		according to the definitions below. There can be as many era description segments as are necessary to describe the different eras. Era description segments are separated by <semicolon> characters.
5160		
5161		
5162	<i>direction</i>	Either a '+' or a '-' character. The '+' character shall indicate that years closer to the <i>start_date</i> have lower numbers than those closer to the <i>end_date</i> . The '-' character shall indicate that years closer to the <i>start_date</i> have higher numbers than those closer to the <i>end_date</i> .
5163		
5164		
5165		
5166		
5167	<i>offset</i>	The number of the year closest to the <i>start_date</i> in the era.
5168	<i>start_date</i>	A date in the form <i>yyyy/mm/dd</i> , where <i>yyyy</i> , <i>mm</i> , and <i>dd</i> are the year, month, and day numbers respectively of the start of the era. Years prior to AD 1 shall be represented as negative numbers.
5169		
5170		
5171	<i>end_date</i>	The ending date of the era, in the same format as the <i>start_date</i> , or one of the two special values "-*" or "+*". The value "-*" shall indicate that the ending date is the beginning of time. The value "+*" shall indicate that the ending date is the end of time.
5172		
5173		
5174		
5175	<i>era_name</i>	The era, corresponding to the %EC conversion specification.
5176	<i>era_format</i>	The format of the year in the era, corresponding to the %EY conversion specification.
5177		
5178	ERA_D_FMT	The era date format.
5179	ERA_T_FMT	The locale's appropriate alternative time format, corresponding to the %EX conversion specification.
5180		
5181	ERA_D_T_FMT	The locale's appropriate alternative date and time format, corresponding to the %Ec conversion specification.
5182		

5183 ALT\_DIGITS The alternative symbols for digits, corresponding to the %O conversion  
 5184 specification modifier. The value consists of <semicolon>-separated symbols.  
 5185 The first is the alternative symbol corresponding to zero, the second is the  
 5186 symbol corresponding to one, and so on. Up to 100 alternative symbols may  
 5187 be specified.

### 5188 7.3.5.3 LC\_TIME Category in the POSIX Locale

5189 The LC\_TIME category definition of the POSIX locale follows; the code listing depicts the  
 5190 *localedef* input; the table represents the same information with the addition of *localedef* keywords,  
 5191 conversion specifiers used by the *date* utility and the *strftime()*, *wcsftime()*, and *strptime()*  
 5192 functions, and *nl\_langinfo()* constants.

```

5193 LC_TIME
5194 # This is the POSIX locale definition for
5195 # the LC_TIME category.
5196 #
5197 # Abbreviated weekday names (%a)
5198 abday      "<S><u><n>" ; "<M><o><n>" ; "<T><u><e>" ; "<W><e><d>" ; \
5199            "<T><h><u>" ; "<F><r><i>" ; "<S><a><t>"
5200 #
5201 # Full weekday names (%A)
5202 day        "<S><u><n><d><a><y>" ; "<M><o><n><d><a><y>" ; \
5203            "<T><u><e><s><d><a><y>" ; "<W><e><d><n><e><s><d><a><y>" ; \
5204            "<T><h><u><r><s><d><a><y>" ; "<F><r><i><d><a><y>" ; \
5205            "<S><a><t><u><r><d><a><y>"
5206 #
5207 # Abbreviated month names (%b)
5208 abmon      "<J><a><n>" ; "<F><e><b>" ; "<M><a><r>" ; \
5209            "<A><p><r>" ; "<M><a><y>" ; "<J><u><n>" ; \
5210            "<J><u><l>" ; "<A><u><g>" ; "<S><e><p>" ; \
5211            "<O><c><t>" ; "<N><o><v>" ; "<D><e><c>"
5212 #
5213 # Full month names (%B)
5214 mon        "<J><a><n><u><a><r><y>" ; "<F><e><b><r><u><a><r><y>" ; \
5215            "<M><a><r><c><h>" ; "<A><p><r><i><l>" ; \
5216            "<M><a><y>" ; "<J><u><n><e>" ; \
5217            "<J><u><l><y>" ; "<A><u><g><u><s><t>" ; \
5218            "<S><e><p><t><e><m><b><e><r>" ; "<O><c><t><o><b><e><r>" ; \
5219            "<N><o><v><e><m><b><e><r>" ; "<D><e><c><e><m><b><e><r>"
5220 #
5221 # Equivalent of AM/PM (%p)      "AM" ; "PM"
5222 am_pm      "<A><M>" ; "<P><M>"
5223 #
5224 # Appropriate date and time representation (%c)
5225 #      "%a %b %e %H:%M:%S %Y"
5226 d_t_fmt    "<percent-sign><a><space><percent-sign><b>\
5227            <space><percent-sign><e><space><percent-sign><H>\
5228            <colon><percent-sign><M><colon><percent-sign><S>\
5229            <space><percent-sign><Y>"
5230 #
5231 # Appropriate date representation (%x)      "%m/%d/%y"
5232 d_fmt      "<percent-sign><m><slash><percent-sign><d>\

```



```

5233 <slash><percent-sign><y>"
5234 #
5235 # Appropriate time representation (%X) "%H:%M:%S"
5236 t_fmt "<percent-sign><H><colon><percent-sign><M>\
5237 <colon><percent-sign><S>"
5238 #
5239 # Appropriate 12-hour time representation (%r) "%I:%M:%S %p"
5240 t_fmt_ampm "<percent-sign><I><colon><percent-sign><M><colon>\
5241 <percent-sign><S><space><percent-sign><p>"
5242 #
5243 END LC_TIME

```

	localedef Keyword	langinfo Constant	Conversion Specification	POSIX Locale Value
5244	d_t_fmt	D_T_FMT	%c	"%a %b %e %H:%M:%S %Y"
5245	d_fmt	D_FMT	%x	"%m/%d/%y"
5246	t_fmt	T_FMT	%X	"%H:%M:%S"
5247	am_pm	AM_STR	%p	"AM"
5248	am_pm	PM_STR	%p	"PM"
5249	t_fmt_ampm	T_FMT_AMP	%r	"%I:%M:%S %p"
5250	day	DAY_1	%A	"Sunday"
5251	day	DAY_2	%A	"Monday"
5252	day	DAY_3	%A	"Tuesday"
5253	day	DAY_4	%A	"Wednesday"
5254	day	DAY_5	%A	"Thursday"
5255	day	DAY_6	%A	"Friday"
5256	day	DAY_7	%A	"Saturday"
5257	abday	ABDAY_1	%a	"Sun"
5258	abday	ABDAY_2	%a	"Mon"
5259	abday	ABDAY_3	%a	"Tue"
5260	abday	ABDAY_4	%a	"Wed"
5261	abday	ABDAY_5	%a	"Thu"
5262	abday	ABDAY_6	%a	"Fri"
5263	abday	ABDAY_7	%a	"Sat"
5264	mon	MON_1	%B	"January"
5265	mon	MON_2	%B	"February"
5266	mon	MON_3	%B	"March"
5267	mon	MON_4	%B	"April"
5268	mon	MON_5	%B	"May"
5269	mon	MON_6	%B	"June"
5270	mon	MON_7	%B	"July"
5271	mon	MON_8	%B	"August"
5272	mon	MON_9	%B	"September"
5273	mon	MON_10	%B	"October"
5274	mon	MON_11	%B	"November"
5275	mon	MON_12	%B	"December"
5276	abmon	ABMON_1	%b	"Jan"
5277	abmon	ABMON_2	%b	"Feb"
5278	abmon	ABMON_3	%b	"Mar"
5279	abmon	ABMON_4	%b	"Apr"
5280	abmon	ABMON_5	%b	"May"
5281	abmon	ABMON_6	%b	"Jun"



localedef Keyword	langinfo Constant	Conversion Specification	POSIX Locale Value
<b>abmon</b>	ABMON_7	%b	"Jul "
<b>abmon</b>	ABMON_8	%b	"Aug "
<b>abmon</b>	ABMON_9	%b	"Sep "
<b>abmon</b>	ABMON_10	%b	"Oct "
<b>abmon</b>	ABMON_11	%b	"Nov "
<b>abmon</b>	ABMON_12	%b	"Dec "
<b>era</b>	ERA	%EC, %Ey, %EY	N/A
<b>era_d_fmt</b>	ERA_D_FMT	%Ex	N/A
<b>era_t_fmt</b>	ERA_T_FMT	%EX	N/A
<b>era_d_t_fmt</b>	ERA_D_T_FMT	%Ec	N/A
<b>alt_digits</b>	ALT_DIGITS	%O	N/A

The entry N/A indicates the value is not available in the POSIX locale.

### 7.3.6 LC\_MESSAGES

The *LC\_MESSAGES* category shall define the format and values used by various utilities for affirmative and negative responses. This information is available through the *nl\_langinfo()* function.

The message catalog used by the standard utilities and selected by the *catopen()* function shall be determined by the setting of *NLSPATH*; see [Chapter 8](#) (on page 173). The *LC\_MESSAGES* category can be specified as part of an *NLSPATH* substitution field.

The following keywords shall be recognized as part of the locale definition file.

**copy** Specify the name of an existing locale which shall be used as the definition of this category. If this keyword is specified, no other keyword shall be specified.

**Note:** This is a *localedef* keyword, unavailable through *nl\_langinfo()*.

**yesexpr** The operand consists of an extended regular expression (see [Section 9.4](#), on page 188) that describes acceptable affirmative responses to a question expecting an affirmative or negative response.

**noexpr** The operand consists of an extended regular expression that describes acceptable negative responses to a question expecting an affirmative or negative response.

#### 7.3.6.1 LC\_MESSAGES Category in the POSIX Locale

The format and values for affirmative and negative responses of the POSIX locale follow; the code listing depicting the *localedef* input, the table representing the same information with the addition of *nl\_langinfo()* constants.

```
LC_MESSAGES
# This is the POSIX locale definition for
# the LC_MESSAGES category.
#
yesexpr "<circumflex><left-square-bracket><y><Y><right-square-bracket>"
#
noexpr "<circumflex><left-square-bracket><n><N><right-square-bracket>"
#
END LC_MESSAGES
```

5327	<b>localedef Keyword</b>	<b>langinfo Constant</b>	<b>POSIX Locale Value</b>
5328	<b>yesexpr</b>	YESEXPR	"^[yY]"
5329	<b>noexpr</b>	NOEXPR	"^[nN]"

## 7.4 Locale Definition Grammar

The grammar and lexical conventions in this section shall together describe the syntax for the locale definition source. The general conventions for this style of grammar are described in XCU [Section 1.3](#) (on page 2325). The grammar shall take precedence over the text in this chapter.

### 7.4.1 Locale Lexical Conventions

The lexical conventions for the locale definition grammar are described in this section.

The following tokens shall be processed (in addition to those string constants shown in the grammar):

<b>LOC_NAME</b>	A string of characters representing the name of a locale.
<b>CHAR</b>	Any single character.
<b>NUMBER</b>	A decimal number, represented by one or more decimal digits.
<b>COLLSYMBOL</b>	A symbolic name, enclosed between angle brackets. The string cannot duplicate any charmap symbol defined in the current charmap (if any), or a <b>COLLELEMENT</b> symbol.
<b>COLLELEMENT</b>	A symbolic name, enclosed between angle brackets, which cannot duplicate either any charmap symbol or a <b>COLLSYMBOL</b> symbol.
<b>CHARCLASS</b>	A string of alphanumeric characters from the portable character set, the first of which is not a digit, consisting of at least one and at most {CHARCLASS_NAME_MAX} bytes, and optionally surrounded by double-quotes.
<b>CHARSYMBOL</b>	A symbolic name, enclosed between angle brackets, from the current charmap (if any).
<b>OCTAL_CHAR</b>	One or more octal representations of the encoding of each byte in a single character. The octal representation consists of an escape character (normally a <backslash>) followed by two or more octal digits.
<b>HEX_CHAR</b>	One or more hexadecimal representations of the encoding of each byte in a single character. The hexadecimal representation consists of an escape character followed by the constant <i>x</i> and two or more hexadecimal digits.
<b>DECIMAL_CHAR</b>	One or more decimal representations of the encoding of each byte in a single character. The decimal representation consists of an escape character followed by a character 'd' and two or more decimal digits.

5364	<b>ELLIPSIS</b>	The string "...".
5365	<b>EXTENDED_REG_EXP</b>	An extended regular expression as defined in the grammar in <a href="#">Section 9.5</a> (on page 192).
5366		
5367	<b>EOL</b>	The line termination character <newline>.

#### 5368 7.4.2 Locale Grammar

5369 This section presents the grammar for the locale definition.

5370	%token	LOC_NAME
5371	%token	CHAR
5372	%token	NUMBER
5373	%token	COLLSYMBOL COLLELEMENT
5374	%token	CHARSYMBOL OCTAL_CHAR HEX_CHAR DECIMAL_CHAR
5375	%token	ELLIPSIS
5376	%token	EXTENDED_REG_EXP
5377	%token	EOL
5378	%start	locale_definition
5379	%%	
5380	locale_definition	: global_statements locale_categories
5381		locale_categories
5382		;
5383	global_statements	: global_statements symbol_redefine
5384		symbol_redefine
5385		;
5386	symbol_redefine	: 'escape_char' CHAR EOL
5387		'comment_char' CHAR EOL
5388		;
5389	locale_categories	: locale_categories locale_category
5390		locale_category
5391		;
5392	locale_category	: lc_ctype   lc_collate   lc_messages
5393		lc_monetary   lc_numeric   lc_time
5394		;
5395	/* The following grammar rules are common to all categories */	
5396	char_list	: char_list char_symbol
5397		char_symbol
5398		;
5399	char_symbol	: CHAR   CHARSYMBOL
5400		OCTAL_CHAR   HEX_CHAR   DECIMAL_CHAR
5401		;
5402	elem_list	: elem_list char_symbol
5403		elem_list COLLSYMBOL
5404		elem_list COLLELEMENT
5405		char_symbol
5406		COLLSYMBOL

```

5407             | COLLELEMENT
5408             ;
5409     symb_list      : symb_list COLLSYMBOL
5410             | COLLSYMBOL
5411             ;
5412     locale_name     : LOC_NAME
5413             | ' ' LOC_NAME ' '
5414             ;
5415     /* The following is the LC_CTYPE category grammar */
5416     lc_ctype        : ctype_hdr ctype_keywords      ctype_tlr
5417             | ctype_hdr 'copy' locale_name EOL ctype_tlr
5418             ;
5419     ctype_hdr       : 'LC_CTYPE' EOL
5420             ;
5421     ctype_keywords  : ctype_keywords ctype_keyword
5422             | ctype_keyword
5423             ;
5424     ctype_keyword   : charclass_keyword charclass_list EOL
5425             | charconv_keyword charconv_list EOL
5426             | 'charclass' charclass_namelist EOL
5427             ;
5428     charclass_namelist : charclass_namelist ';' CHARCLASS
5429             | CHARCLASS
5430             ;
5431     charclass_keyword : 'upper' | 'lower' | 'alpha' | 'digit'
5432             | 'punct' | 'xdigit' | 'space' | 'print'
5433             | 'graph' | 'blank' | 'cntrl' | 'alnum'
5434             | CHARCLASS
5435             ;
5436     charclass_list   : charclass_list ';' char_symbol
5437             | charclass_list ';' ELLIPSIS ';' char_symbol
5438             | char_symbol
5439             ;
5440     charconv_keyword : 'toupper'
5441             | 'tolower'
5442             ;
5443     charconv_list    : charconv_list ';' charconv_entry
5444             | charconv_entry
5445             ;
5446     charconv_entry   : '(' char_symbol ',' char_symbol ')'
5447             ;
5448     ctype_tlr        : 'END' 'LC_CTYPE' EOL
5449             ;
5450     /* The following is the LC_COLLATE category grammar */
5451     lc_collate       : collate_hdr collate_keywords      collate_tlr

```

```

5452             | collate_hdr 'copy' locale_name EOL collate_tlr
5453             ;
5454     collate_hdr      : 'LC_COLLATE' EOL
5455             ;
5456     collate_keywords :                order_statements
5457             | opt_statements order_statements
5458             ;
5459     opt_statements   : opt_statements collating_symbols
5460             | opt_statements collating_elements
5461             | collating_symbols
5462             | collating_elements
5463             ;
5464     collating_symbols : 'collating-symbol' COLLSYMBOL EOL
5465             ;
5466     collating_elements : 'collating-element' COLLELEMENT
5467             | 'from' '"' elem_list '"' EOL
5468             ;
5469     order_statements : order_start collation_order order_end
5470             ;
5471     order_start       : 'order_start' EOL
5472             | 'order_start' order_opts EOL
5473             ;
5474     order_opts        : order_opts ';' order_opt
5475             | order_opt
5476             ;
5477     order_opt         : order_opt ',' opt_word
5478             | opt_word
5479             ;
5480     opt_word          : 'forward' | 'backward' | 'position'
5481             ;
5482     collation_order    : collation_order collation_entry
5483             | collation_entry
5484             ;
5485     collation_entry    : COLLSYMBOL EOL
5486             | collation_element weight_list EOL
5487             | collation_element EOL
5488             ;
5489     collation_element  : char_symbol
5490             | COLLELEMENT
5491             | ELLIPSIS
5492             | 'UNDEFINED'
5493             ;
5494     weight_list        : weight_list ';' weight_symbol
5495             | weight_list ';'
5496             | weight_symbol
5497             ;

```

```

5498     weight_symbol      : /* empty */
5499                          | char_symbol
5500                          | COLLSYMBOL
5501                          | ''' elem_list '''
5502                          | ''' symb_list '''
5503                          | ELLIPSIS
5504                          | 'IGNORE'
5505                          ;
5506     order_end            : 'order_end' EOL
5507                          ;
5508     collate_tlr           : 'END' 'LC_COLLATE' EOL
5509                          ;
5510     /* The following is the LC_MESSAGES category grammar */
5511     lc_messages           : messages_hdr messages_keywords      messages_tlr
5512                          | messages_hdr 'copy' locale_name EOL messages_tlr
5513                          ;
5514     messages_hdr          : 'LC_MESSAGES' EOL
5515                          ;
5516     messages_keywords     : messages_keywords messages_keyword
5517                          | messages_keyword
5518                          ;
5519     messages_keyword      : 'yesexpr' ''' EXTENDED_REG_EXP ''' EOL
5520                          | 'noexpr'  ''' EXTENDED_REG_EXP ''' EOL
5521                          ;
5522     messages_tlr          : 'END' 'LC_MESSAGES' EOL
5523                          ;
5524     /* The following is the LC_MONETARY category grammar */
5525     lc_monetary           : monetary_hdr monetary_keywords      monetary_tlr
5526                          | monetary_hdr 'copy' locale_name EOL monetary_tlr
5527                          ;
5528     monetary_hdr          : 'LC_MONETARY' EOL
5529                          ;
5530     monetary_keywords     : monetary_keywords monetary_keyword
5531                          | monetary_keyword
5532                          ;
5533     monetary_keyword      : mon_keyword_string mon_string EOL
5534                          | mon_keyword_char NUMBER EOL
5535                          | mon_keyword_char '-1' EOL
5536                          | mon_keyword_grouping mon_group_list EOL
5537                          ;
5538     mon_keyword_string    : 'int_curr_symbol' | 'currency_symbol'
5539                          | 'mon_decimal_point' | 'mon_thousands_sep'
5540                          | 'positive_sign' | 'negative_sign'
5541                          ;
5542     mon_string            : ''' char_list '''

```

```

5543         | '""'
5544         ;
5545     mon_keyword_char    : 'int_frac_digits' | 'frac_digits'
5546         | 'p_cs_precedes' | 'p_sep_by_space'
5547         | 'n_cs_precedes' | 'n_sep_by_space'
5548         | 'p_sign_posn' | 'n_sign_posn'
5549         | 'int_p_cs_precedes' | 'int_p_sep_by_space'
5550         | 'int_n_cs_precedes' | 'int_n_sep_by_space'
5551         | 'int_p_sign_posn' | 'int_n_sign_posn'
5552         ;
5553     mon_keyword_grouping : 'mon_grouping'
5554         ;
5555     mon_group_list       : NUMBER
5556         | mon_group_list ';' NUMBER
5557         ;
5558     monetary_tlr        : 'END' 'LC_MONETARY' EOL
5559         ;
5560     /* The following is the LC_NUMERIC category grammar */
5561     lc_numeric           : numeric_hdr numeric_keywords      numeric_tlr
5562         | numeric_hdr 'copy' locale_name EOL numeric_tlr
5563         ;
5564     numeric_hdr          : 'LC_NUMERIC' EOL
5565         ;
5566     numeric_keywords     : numeric_keywords numeric_keyword
5567         | numeric_keyword
5568         ;
5569     numeric_keyword      : num_keyword_string num_string EOL
5570         | num_keyword_grouping num_group_list EOL
5571         ;
5572     num_keyword_string   : 'decimal_point'
5573         | 'thousands_sep'
5574         ;
5575     num_string           : '"' char_list '"'
5576         | '""'
5577         ;
5578     num_keyword_grouping : 'grouping'
5579         ;
5580     num_group_list       : NUMBER
5581         | num_group_list ';' NUMBER
5582         ;
5583     numeric_tlr          : 'END' 'LC_NUMERIC' EOL
5584         ;
5585     /* The following is the LC_TIME category grammar */
5586     lc_time              : time_hdr time_keywords          time_tlr
5587         | time_hdr 'copy' locale_name EOL time_tlr

```

```

5588                                     ;
5589     time_hdr                        : 'LC_TIME' EOL
5590                                     ;
5591     time_keywords                    : time_keywords time_keyword
5592                                     | time_keyword
5593                                     ;
5594     time_keyword                     : time_keyword_name time_list EOL
5595                                     | time_keyword_fmt time_string EOL
5596                                     | time_keyword_opt time_list EOL
5597                                     ;
5598     time_keyword_name                : 'abday' | 'day' | 'abmon' | 'mon'
5599                                     ;
5600     time_keyword_fmt                 : 'd_t_fmt' | 'd_fmt' | 't_fmt'
5601                                     | 'am_pm' | 't_fmt_ampm'
5602                                     ;
5603     time_keyword_opt                 : 'era' | 'era_d_fmt' | 'era_t_fmt'
5604                                     | 'era_d_t_fmt' | 'alt_digits'
5605                                     ;
5606     time_list                        : time_list ';' time_string
5607                                     | time_string
5608                                     ;
5609     time_string                      : '"' char_list '"'
5610                                     ;
5611     time_tlr                         : 'END' 'LC_TIME' EOL
5612                                     ;

```



# Environment Variables

## 8.1 Environment Variable Definition

Environment variables defined in this chapter affect the operation of multiple utilities, functions, and applications. There are other environment variables that are of interest only to specific utilities. Environment variables that apply to a single utility only are defined as part of the utility description. See the ENVIRONMENT VARIABLES section of the utility descriptions in the Shell and Utilities volume of POSIX.1-2008 for information on environment variable usage.

The value of an environment variable is a string of characters. For a C-language program, an array of strings called the environment shall be made available when a process begins. The array is pointed to by the external variable *environ*, which is defined as:

```
extern char **environ;
```

These strings have the form *name=value*; *names* shall not contain the character '='. For values to be portable across systems conforming to POSIX.1-2008, the value shall be composed of characters from the portable character set (except NUL and as indicated below). There is no meaning associated with the order of strings in the environment. If more than one string in an environment of a process has the same *name*, the consequences are undefined.

Environment variable names used by the utilities in the Shell and Utilities volume of POSIX.1-2008 consist solely of uppercase letters, digits, and the <underscore> ('\_') from the characters defined in Table 6-1 (on page 125) and do not begin with a digit. Other characters may be permitted by an implementation; applications shall tolerate the presence of such names. Uppercase and lowercase letters shall retain their unique identities and shall not be folded together. The name space of environment variable names containing lowercase letters is reserved for applications. Applications can define any environment variables with names from this name space without modifying the behavior of the standard utilities.

**Note:** Other applications may have difficulty dealing with environment variable names that start with a digit. For this reason, use of such names is not recommended anywhere.

The *values* that the environment variables may be assigned are not restricted except that they are considered to end with a null byte and the total space used to store the environment and the arguments to the process is limited to {ARG\_MAX} bytes.

Other *name=value* pairs may be placed in the environment by, for example, calling any of the *setenv()*, *unsetenv()*, or *putenv()* functions, assigning a new value to the *environ* variable, or by using *envp* arguments when creating a process; see *exec* in the System Interfaces volume of POSIX.1-2008.

If the application modifies the pointers to which *environ* points, the behavior of all interfaces described in the System Interfaces volume of POSIX.1-2008 is undefined.

It is unwise to conflict with certain variables that are frequently exported by widely used command interpreters and applications:

5651	ARFLAGS	IFS	MAILPATH	PS1
5652	CC	LANG	MAILRC	PS2
5653	CDPATH	LC_ALL	MAKEFLAGS	PS3
5654	CFLAGS	LC_COLLATE	MAKESHELL	PS4
5655	CHARSET	LC_CTYPE	MANPATH	PWD
5656	COLUMNS	LC_MESSAGES	MBOX	RANDOM
5657	DATMSK	LC_MONETARY	MORE	SECONDS
5658	DEAD	LC_NUMERIC	MSGVERB	SHELL
5659	EDITOR	LC_TIME	NLSPATH	TERM
5660	ENV	LDFLAGS	NPROC	TERMCAP
5661	EXINIT	LEX	OLDPWD	TERMINFO
5662	FC	LFLAGS	OPTARG	TMPDIR
5663	FCEDIT	LINENO	OPTERR	TZ
5664	FFLAGS	LINES	OPTIND	USER
5665	GET	LISTER	PAGER	VISUAL
5666	GFLAGS	LOGNAME	PATH	YACC
5667	HISTFILE	LPDEST	PPID	YFLAGS
5668	HISTORY	MAIL	PRINTER	
5669	HISTSIZE	MAILCHECK	PROCLANG	
5670	HOME	MAILER	PROJECTDIR	

If the variables in the following two sections are present in the environment during the execution of an application or utility, they shall be given the meaning described below. Some are placed into the environment by the implementation at the time the user logs in; all can be added or changed by the user or any ancestor of the current process. The implementation adds or changes environment variables named in POSIX.1-2008 only as specified in POSIX.1-2008. If they are defined in the application's environment, the utilities in the Shell and Utilities volume of POSIX.1-2008 and the functions in the System Interfaces volume of POSIX.1-2008 assume they have the specified meaning. Conforming applications shall not set these environment variables to have meanings other than as described. See *getenv()* (on page 1024) and XCU Section 2.12 (on page 2371) for methods of accessing these variables.

## 8.2 Internationalization Variables

This section describes environment variables that are relevant to the operation of internationalized interfaces described in POSIX.1-2008.

Users may use the following environment variables to announce specific localization requirements to applications. Applications can retrieve this information using the *setlocale()* function to initialize the correct behavior of the internationalized interfaces. The descriptions of the internationalization environment variables describe the resulting behavior only when the application locale is initialized in this way. The use of the internationalization variables by utilities described in the Shell and Utilities volume of POSIX.1-2008 is described in the ENVIRONMENT VARIABLES section for those utilities in addition to the global effects described in this section.

**LANG** This variable shall determine the locale category for native language, local customs, and coded character set in the absence of the *LC\_ALL* and other *LC\_\** (*LC\_COLLATE*, *LC\_CTYPE*, *LC\_MESSAGES*, *LC\_MONETARY*, *LC\_NUMERIC*, *LC\_TIME*) environment variables. This can be used by applications to determine the language to use for error messages and instructions, collating sequences, date formats, and so on.

5698	<i>LC_ALL</i>	This variable shall determine the values for all locale categories. The value of
5699		the <i>LC_ALL</i> environment variable has precedence over any of the other
5700		environment variables starting with <i>LC_</i> ( <i>LC_COLLATE</i> , <i>LC_CTYPE</i> ,
5701		<i>LC_MESSAGES</i> , <i>LC_MONETARY</i> , <i>LC_NUMERIC</i> , <i>LC_TIME</i> ) and the <i>LANG</i>
5702		environment variable.
5703	<i>LC_COLLATE</i>	This variable shall determine the locale category for character collation. It
5704		determines collation information for regular expressions and sorting,
5705		including equivalence classes and multi-character collating elements, in
5706		various utilities and the <i>strcoll()</i> and <i>strxfrm()</i> functions. Additional semantics
5707		of this variable, if any, are implementation-defined.
5708	<i>LC_CTYPE</i>	This variable shall determine the locale category for character handling
5709		functions, such as <i>tolower()</i> , <i>toupper()</i> , and <i>isalpha()</i> . This environment
5710		variable determines the interpretation of sequences of bytes of text data as
5711		characters (for example, single as opposed to multi-byte characters), the
5712		classification of characters (for example, alpha, digit, graph), and the behavior
5713		of character classes. Additional semantics of this variable, if any, are
5714		implementation-defined.
5715	<i>LC_MESSAGES</i>	This variable shall determine the locale category for processing affirmative
5716		and negative responses and the language and cultural conventions in which
5717		messages should be written. It also affects the behavior of the <i>catopen()</i>
5718		function in determining the message catalog. Additional semantics of this
5719		variable, if any, are implementation-defined. The language and cultural
5720		conventions of diagnostic and informative messages whose format is
5721		unspecified by POSIX.1-2008 should be affected by the setting of
5722		<i>LC_MESSAGES</i> .
5723	<i>LC_MONETARY</i>	This variable shall determine the locale category for monetary-related numeric
5724		formatting information. Additional semantics of this variable, if any, are
5725		implementation-defined.
5726	<i>LC_NUMERIC</i>	This variable shall determine the locale category for numeric formatting (for
5727		example, thousands separator and radix character) information in various
5728		utilities as well as the formatted I/O operations in <i>printf()</i> and <i>scanf()</i> and the
5729		string conversion functions in <i>strtod()</i> . Additional semantics of this variable,
5730		if any, are implementation-defined.
5731	<i>LC_TIME</i>	This variable shall determine the locale category for date and time formatting
5732		information. It affects the behavior of the time functions in <i>strptime()</i> .
5733		Additional semantics of this variable, if any, are implementation-defined.
5734	<i>NLSPATH</i>	This variable shall contain a sequence of templates that the <i>catopen()</i> function
5735		uses when attempting to locate message catalogs. Each template consists of an
5736		optional prefix, one or more conversion specifications, a pathname, and an
5737		optional suffix.
5738		For example:
5739		<i>NLSPATH</i> ="/system/nlslib/%N.cat "
5740		defines that <i>catopen()</i> should look for all message catalogs in the directory
5741		<b>/system/nlslib</b> , where the catalog name should be constructed from the <i>name</i>
5742		parameter passed to <i>catopen()</i> (%N), with the suffix <b>.cat</b> .
5743		Conversion specifications consist of a ' % ' symbol, followed by a single-letter
5744		keyword. The following keywords are currently defined:

5745                    %N   The value of the *name* parameter passed to *catopen()*.

5746                    %L   The value of the *LC\_MESSAGES* category.

5747                    %l   The *language* element from the *LC\_MESSAGES* category.

5748                    %t   The *territory* element from the *LC\_MESSAGES* category.

5749                    %c   The *codeset* element from the *LC\_MESSAGES* category.

5750                    %%   A single '%' character.

5751                    An empty string is substituted if the specified value is not currently defined.

5752                    The separators <underscore> ('\_') and <period> ('.') are not included in

5753                    the %t and %c conversion specifications.

5754                    Templates defined in *NLSPATH* are separated by <colon> characters (':'). A

5755                    leading or two adjacent <colon> characters ("::") is equivalent to specifying

5756                    %N. For example:

5757                    `NLSPATH=":%N.cat:/nlslib/%L/%N.cat"`

5758                    indicates to *catopen()* that it should look for the requested message catalog in

5759                    *name*, *name.cat*, and */nlslib/category/name.cat*, where *category* is the value of the

5760                    *LC\_MESSAGES* category of the current locale.

5761                    Users should not set the *NLSPATH* variable unless they have a specific reason

5762                    to override the default system path. Setting *NLSPATH* to override the default

5763                    system path produces undefined results in the standard utilities and in

5764                    applications with appropriate privileges.

5765                    The environment variables *LANG*, *LC\_ALL*, *LC\_COLLATE*, *LC\_CTYPE*, *LC\_MESSAGES*,

5766                    *LC\_MONETARY*, *LC\_NUMERIC*, *LC\_TIME*, and *NLSPATH* provide for the support of

5767                    internationalized applications. The standard utilities shall make use of these environment

5768                    variables as described in this section and the individual ENVIRONMENT VARIABLES sections

5769                    for the utilities. If these variables specify locale categories that are not based upon the same

5770                    underlying codeset, the results are unspecified.

5771                    The values of locale categories shall be determined by a precedence order; the first condition met

5772                    below determines the value:

- 5773                    1. If the *LC\_ALL* environment variable is defined and is not null, the value of *LC\_ALL* shall
- 5774                    be used.
- 5775                    2. If the *LC\_\** environment variable (*LC\_COLLATE*, *LC\_CTYPE*, *LC\_MESSAGES*,
- 5776                    *LC\_MONETARY*, *LC\_NUMERIC*, *LC\_TIME*) is defined and is not null, the value of the
- 5777                    environment variable shall be used to initialize the category that corresponds to the
- 5778                    environment variable.
- 5779                    3. If the *LANG* environment variable is defined and is not null, the value of the *LANG*
- 5780                    environment variable shall be used.
- 5781                    4. If the *LANG* environment variable is not set or is set to the empty string, the
- 5782                    implementation-defined default locale shall be used.

5783                    If the locale value is "C" or "POSIX", the POSIX locale shall be used and the standard utilities

5784                    behave in accordance with the rules in [Section 7.2](#) (on page 136) for the associated category.

5785                    If the locale value begins with a <slash>, it shall be interpreted as the pathname of a file that was

5786                    created in the output format used by the *localedef* utility; see OUTPUT FILES under *localedef*.

5787                    Referencing such a pathname shall result in that locale being used for the indicated category.

5788 XSI If the locale value has the form:

5789 `language[_territory][.codeset]`

5790 it refers to an implementation-provided locale, where settings of language, territory, and codeset

5791 are implementation-defined.

5792 `LC_COLLATE`, `LC_CTYPE`, `LC_MESSAGES`, `LC_MONETARY`, `LC_NUMERIC`, and `LC_TIME` are

5793 defined to accept an additional field `@modifier`, which allows the user to select a specific instance

5794 of localization data within a single category (for example, for selecting the dictionary as opposed

5795 to the character ordering of data). The syntax for these environment variables is thus defined as:

5796 `[language[_territory][.codeset][@modifier]]`

5797 For example, if a user wanted to interact with the system in French, but required to sort German

5798 text files, `LANG` and `LC_COLLATE` could be defined as:

5799 `LANG=Fr_FR`

5800 `LC_COLLATE=De_DE`

5801 This could be extended to select dictionary collation (say) by use of the `@modifier` field; for

5802 example:

5803 `LC_COLLATE=De_DE@dict`

5804 An implementation may support other formats.

5805 If the locale value is not recognized by the implementation, the behavior is unspecified.

5806 These environment variables are used by the `newlocale()` and `setlocale()` functions, and by the

5807 standard utilities.

5808 Additional criteria for determining a valid locale name are implementation-defined.

### 5809 8.3 Other Environment Variables

5810 **COLUMNS** This variable shall represent a decimal integer >0 used to indicate the user's

5811 preferred width in column positions for the terminal screen or window; see

5812 [Section 3.103](#) (on page 50). If this variable is unset or null, the implementation

5813 determines the number of columns, appropriate for the terminal or window,

5814 in an unspecified manner. When `COLUMNS` is set, any terminal-width

5815 information implied by `TERM` is overridden. Users and conforming

5816 applications should not set `COLUMNS` unless they wish to override the

5817 system selection and produce output unrelated to the terminal characteristics.

5818 Users should not need to set this variable in the environment unless there is a

5819 specific reason to override the implementation's default behavior, such as to

5820 display data in an area arbitrarily smaller than the terminal or window.

5821 XSI **DATEMSK** Indicates the pathname of the template file used by `getdate()`.

5822 **HOME** The system shall initialize this variable at the time of login to be a pathname of

5823 the user's home directory. See [<pwd.h>](#).

5824 **LINES** This variable shall represent a decimal integer >0 used to indicate the user's

5825 preferred number of lines on a page or the vertical screen or window size in

5826 lines. A line in this case is a vertical measure large enough to hold the tallest

5827 character in the character set being displayed. If this variable is unset or null,



5828		the implementation determines the number of lines, appropriate for the
5829		terminal or window (size, terminal baud rate, and so on), in an unspecified
5830		manner. When <i>LINES</i> is set, any terminal-height information implied by
5831		<i>TERM</i> is overridden. Users and conforming applications should not set <i>LINES</i>
5832		unless they wish to override the system selection and produce output
5833		unrelated to the terminal characteristics.
5834		Users should not need to set this variable in the environment unless there is a
5835		specific reason to override the implementation's default behavior, such as to
5836		display data in an area arbitrarily smaller than the terminal or window.
5837	<i>LOGNAME</i>	The system shall initialize this variable at the time of login to be the user's
5838		login name. See <a href="#">&lt;pwd.h&gt;</a> . For a value of <i>LOGNAME</i> to be portable across
5839		implementations of POSIX.1-2008, the value should be composed of characters
5840		from the portable filename character set.
5841	XSI	<i>MSGVERB</i> Describes which message components shall be used in writing messages by
5842		<i>fntmsg()</i> .
5843	<i>PATH</i>	This variable shall represent the sequence of path prefixes that certain
5844		functions and utilities apply in searching for an executable file known only by
5845		a filename. The prefixes shall be separated by a <colon> (':'). When a non-
5846		zero-length prefix is applied to this filename, a <slash> shall be inserted
5847		between the prefix and the filename if the prefix did not end in <slash>. A
5848		zero-length prefix is a legacy feature that indicates the current working
5849		directory. It appears as two adjacent <colon> characters (": :"), as an initial
5850		<colon> preceding the rest of the list, or as a trailing <colon> following the
5851		rest of the list. A strictly conforming application shall use an actual pathname
5852		(such as .) to represent the current working directory in <i>PATH</i> . The list shall
5853		be searched from beginning to end, applying the filename to each prefix, until
5854		an executable file with the specified name and appropriate execution
5855		permissions is found. If the pathname being sought contains a <slash>, the
5856		search through the path prefixes shall not be performed. If the pathname
5857		begins with a <slash>, the specified path is resolved (see <a href="#">Section 4.13</a> , on page
5858		111). If <i>PATH</i> is unset or is set to null, the path search is implementation-
5859		defined.
5860		Since <colon> is a separator in this context, directory names that might be
5861		used in <i>PATH</i> should not include a <colon> character.
5862	<i>PWD</i>	This variable shall represent an absolute pathname of the current working
5863		directory. It shall not contain any components that are dot or dot-dot. The
5864		value is set by the <i>cd</i> utility, and by the <i>sh</i> utility during initialization.
5865	<i>SHELL</i>	This variable shall represent a pathname of the user's preferred command
5866		language interpreter. If this interpreter does not conform to the Shell
5867		Command Language in XCU <a href="#">Chapter 2</a> (on page 2335), utilities may behave
5868		differently from those described in POSIX.1-2008.
5869	<i>TMPDIR</i>	This variable shall represent a pathname of a directory made available for
5870		programs that need a place to create temporary files.
5871	<i>TERM</i>	This variable shall represent the terminal type for which output is to be
5872		prepared. This information is used by utilities and application programs
5873		wishing to exploit special capabilities specific to a terminal. The format and
5874		allowable values of this environment variable are unspecified.

5875            *TZ*            This variable shall represent timezone information. The contents of the  
 5876 environment variable named *TZ* shall be used by the *ctime()*, *ctime\_r()*,  
 5877 *localtime()*, *localtime\_r()*, *strptime()*, *mktime()*, functions, and by various  
 5878 utilities, to override the default timezone. The value of *TZ* has one of the two  
 5879 forms (spaces inserted for clarity):

5880                    *:characters*

5881                    or:

5882                    *std offset dst offset, rule*

5883            If *TZ* is of the first format (that is, if the first character is a <colon>), the  
 5884 characters following the <colon> are handled in an implementation-defined  
 5885 manner.

5886            The expanded format (for all *TZ*s whose value does not have a <colon> as the  
 5887 first character) is as follows:

5888                    *stdoffset[dst[offset][,start[/time],end[/time]]]*

5889            Where:

5890            *std* and *dst*    Indicate no less than three, nor more than {TZNAME\_MAX},  
 5891 bytes that are the designation for the standard (*std*) or the  
 5892 alternative (*dst*—such as Daylight Savings Time) timezone. Only  
 5893 *std* is required; if *dst* is missing, then the alternative time does  
 5894 not apply in this locale.

5895            Each of these fields may occur in either of two formats quoted or  
 5896 unquoted:

5897                    — In the quoted form, the first character shall be the <less-  
 5898 than-sign> ('<') character and the last character shall be  
 5899 the <greater-than-sign> ('>') character. All characters  
 5900 between these quoting characters shall be alphanumeric  
 5901 characters from the portable character set in the current  
 5902 locale, the <plus-sign> ('+') character, or the <hyphen-  
 5903 minus>('−') character. The *std* and *dst* fields in this case  
 5904 shall not include the quoting characters.

5905                    — In the unquoted form, all characters in these fields shall be  
 5906 alphabetic characters from the portable character set in the  
 5907 current locale.

5908            The interpretation of these fields is unspecified if either field is  
 5909 less than three bytes (except for the case when *dst* is missing),  
 5910 more than {TZNAME\_MAX} bytes, or if they contain characters  
 5911 other than those specified.

5912                    *offset*            Indicates the value added to the local time to arrive at  
 5913 Coordinated Universal Time. The *offset* has the form:

5914                    *hh[:mm[:ss]]*

5915            The minutes (*mm*) and seconds (*ss*) are optional. The hour (*hh*)  
 5916 shall be required and may be a single digit. The *offset* following  
 5917 *std* shall be required. If no *offset* follows *dst*, the alternative time  
 5918 is assumed to be one hour ahead of standard time. One or more  
 5919 digits may be used; the value is always interpreted as a decimal

5920		number. The hour shall be between zero and 24, and the minutes
5921		(and seconds)—if present—between zero and 59. The result of
5922		using values outside of this range is unspecified. If preceded by
5923		a '−', the timezone shall be east of the Prime Meridian;
5924		otherwise, it shall be west (which may be indicated by an
5925		optional preceding '+').
5926	<i>rule</i>	Indicates when to change to and back from the alternative time.
5927		The <i>rule</i> has the form:
5928		<i>date</i> [/ <i>time</i> ], <i>date</i> [/ <i>time</i> ]
5929		where the first <i>date</i> describes when the change from standard to
5930		alternative time occurs and the second <i>date</i> describes when the
5931		change back happens. Each <i>time</i> field describes when, in current
5932		local time, the change to the other time is made.
5933		The format of <i>date</i> is one of the following:
5934	<i>Jn</i>	The Julian day <i>n</i> ( $1 \leq n \leq 365$ ). Leap days shall not be
5935		counted. That is, in all years—including leap years—
5936		February 28 is day 59 and March 1 is day 60. It is
5937		impossible to refer explicitly to the occasional February
5938		29.
5939	<i>n</i>	The zero-based Julian day ( $0 \leq n \leq 365$ ). Leap days shall
5940		be counted, and it is possible to refer to February 29.
5941	<i>Mm.n.d</i>	The <i>d</i> 'th day ( $0 \leq d \leq 6$ ) of week <i>n</i> of month <i>m</i> of the
5942		year ( $1 \leq n \leq 5$ , $1 \leq m \leq 12$ , where week 5 means "the last
5943		<i>d</i> day in month <i>m</i> " which may occur in either the fourth
5944		or the fifth week). Week 1 is the first week in which the
5945		<i>d</i> 'th day occurs. Day zero is Sunday.
5946		The <i>time</i> has the same format as <i>offset</i> except that no leading sign
5947		('−' or '+') is allowed. The default, if <i>time</i> is not given, shall be
5948		02:00:00.



# Regular Expressions

Regular Expressions (REs) provide a mechanism to select specific strings from a set of character strings.

Regular expressions are a context-independent syntax that can represent a wide variety of character sets and character set orderings, where these character sets are interpreted according to the current locale. While many regular expressions can be interpreted differently depending on the current locale, many features, such as character class expressions, provide for contextual invariance across locales.

The Basic Regular Expression (BRE) notation and construction rules in [Section 9.3](#) (on page 183) shall apply to most utilities supporting regular expressions. Some utilities, instead, support the Extended Regular Expressions (ERE) described in [Section 9.4](#) (on page 188); any exceptions for both cases are noted in the descriptions of the specific utilities using regular expressions. Both BREs and EREs are supported by the Regular Expression Matching interface in the System Interfaces volume of POSIX.1-2008 under *regcomp()*, *regex()*, and related functions.

## 9.1 Regular Expression Definitions

For the purposes of this section, the following definitions shall apply:

### entire regular expression

The concatenated set of one or more BREs or EREs that make up the pattern specified for string selection.

### matched

A sequence of zero or more characters shall be said to be matched by a BRE or ERE when the characters in the sequence correspond to a sequence of characters defined by the pattern.

Matching shall be based on the bit pattern used for encoding the character, not on the graphic representation of the character. This means that if a character set contains two or more encodings for a graphic symbol, or if the strings searched contain text encoded in more than one codeset, no attempt is made to search for any other representation of the encoded symbol. If that is required, the user can specify equivalence classes containing all variations of the desired graphic symbol.

The search for a matching sequence starts at the beginning of a string and stops when the first sequence matching the expression is found, where “first” is defined to mean “begins earliest in the string”. If the pattern permits a variable number of matching characters and thus there is more than one such sequence starting at that point, the longest such sequence is matched. For example, the BRE “bb\*” matches the second to fourth characters of the string “abbbc”, and the ERE “(wee|week)(knights|night)” matches all ten characters of the string “weeknights”.

Consistent with the whole match being the longest of the leftmost matches, each subpattern, from left to right, shall match the longest possible string. For this purpose, a null string shall be considered to be longer than no match at all. For example, matching the BRE

"\(.\*\).\*" against "abcdef", the subexpression "(\1)" is "abcdef", and matching the BRE "\ (a\*\)" against "bc", the subexpression "(\1)" is the null string.

When a multi-character collating element in a bracket expression (see [Section 9.3.5](#), on page 184) is involved, the longest sequence shall be measured in characters consumed from the string to be matched; that is, the collating element counts not as one element, but as the number of characters it matches.

#### **BRE (ERE) matching a single character**

A BRE or ERE that shall match either a single character or a single collating element.

Only a BRE or ERE of this type that includes a bracket expression (see [Section 9.3.5](#), on page 184) can match a collating element.

#### **BRE (ERE) matching multiple characters**

A BRE or ERE that shall match a concatenation of single characters or collating elements.

Such a BRE or ERE is made up from a BRE (ERE) matching a single character and BRE (ERE) special characters.

#### **invalid**

This section uses the term “invalid” for certain constructs or conditions. Invalid REs shall cause the utility or function using the RE to generate an error condition. When invalid is not used, violations of the specified syntax or semantics for REs produce undefined results: this may entail an error, enabling an extended syntax for that RE, or using the construct in error as literal characters to be matched. For example, the BRE construct "\{1,2,3\}" does not comply with the grammar. A conforming application cannot rely on it producing an error nor matching the literal characters "\{1,2,3\}".

## **9.2 Regular Expression General Requirements**

The requirements in this section shall apply to both basic and extended regular expressions.

The use of regular expressions is generally associated with text processing. REs (BREs and EREs) operate on text strings; that is, zero or more characters followed by an end-of-string delimiter (typically NUL). Some utilities employing regular expressions limit the processing to lines; that is, zero or more characters followed by a <newline>.

In the functions processing regular expressions described in System Interfaces volume of POSIX.1-2008, the <newline> is regarded as an ordinary character and both a <period> and a non-matching list can match one. The Shell and Utilities volume of POSIX.1-2008 specifies within the individual descriptions of those standard utilities employing regular expressions whether they permit matching of <newline> characters; if not stated otherwise, the use of literal <newline> characters or any escape sequence equivalent in either patterns or matched text produces undefined results. Those utilities (like *grep*) that do not allow <newline> characters to match are responsible for eliminating any <newline> from strings before matching against the RE. The *regcomp()* function in the System Interfaces volume of POSIX.1-2008, however, can provide support for such processing without violating the rules of this section.

The interfaces specified in POSIX.1-2008 do not permit the inclusion of a NUL character in an RE or in the string to be matched. If during the operation of a standard utility a NUL is included in the text designated to be matched, that NUL may designate the end of the text string for the purposes of matching.

When a standard utility or function that uses regular expressions specifies that pattern matching shall be performed without regard to the case (uppercase or lowercase) of either data or

patterns, then when each character in the string is matched against the pattern, not only the character, but also its case counterpart (if any), shall be matched. This definition of case-insensitive processing is intended to allow matching of multi-character collating elements as well as characters, as each character in the string is matched using both its cases. For example, in a locale where "Ch" is a multi-character collating element and where a matching list expression matches such elements, the RE "[.Ch.]" when matched against the string "char" is in reality matched against "ch", "Ch", "cH", and "CH".

The implementation shall support any regular expression that does not exceed 256 bytes in length.

## 9.3 Basic Regular Expressions

### 9.3.1 BREs Matching a Single Character or Collating Element

A BRE ordinary character, a special character preceded by a <backslash>, or a <period> shall match a single character. A bracket expression shall match a single character or a single collating element.

### 9.3.2 BRE Ordinary Characters

An ordinary character is a BRE that matches itself: any character in the supported character set, except for the BRE special characters listed in [Section 9.3.3](#).

The interpretation of an ordinary character preceded by an unescaped <backslash> ('\\') is undefined, except for:

- The characters ' ) ' , ' ( ' , ' { ' , and ' } '
- The digits 1 to 9 inclusive (see [Section 9.3.6](#), on page 186)
- A character inside a bracket expression

### 9.3.3 BRE Special Characters

A BRE special character has special properties in certain contexts. Outside those contexts, or when preceded by a <backslash>, such a character is a BRE that matches the special character itself. The BRE special characters and the contexts in which they have their special meaning are as follows:

. [ \ The <period>, <left-square-bracket>, and <backslash> shall be special except when used in a bracket expression (see [Section 9.3.5](#), on page 184). An expression containing a '[' that is unescaped and is not part of a bracket expression produces undefined results.

\* The <asterisk> shall be special except when used:

— In a bracket expression

— As the first character of an entire BRE (after an initial '<sup>^</sup>', if any)

— As the first character of a subexpression (after an initial '<sup>^</sup>', if any); see [Section 9.3.6](#) (on page 186)

<sup>^</sup> The <circumflex> shall be special when used as an anchor (see [Section 9.3.8](#), on page 188). The <circumflex> shall signify a non-matching list expression when it occurs first in a list, immediately following a <left-square-bracket> (see [Section 9.3.5](#)).

<sup>\$</sup> The <dollar-sign> shall be special when used as an anchor.

### 9.3.4 Periods in BREs

A <period> ('<sup>.</sup>'), when used outside a bracket expression, is a BRE that shall match any character in the supported character set except NUL.

### 9.3.5 RE Bracket Expression

A bracket expression (an expression enclosed in square brackets, "[ ]") is an RE that shall match a specific set of single characters, and may match a specific set of multi-character collating elements, based on the non-empty set of list expressions contained in the bracket expression.

The following rules and definitions apply to bracket expressions:

1. A bracket expression is either a matching list expression or a non-matching list expression. It consists of one or more expressions: ordinary characters, collating elements, collating symbols, equivalence classes, character classes, or range expressions. The <right-square-bracket> ('<sup>]</sup>') shall lose its special meaning and represent itself in a bracket expression if it occurs first in the list (after an initial <circumflex> ('<sup>^</sup>'), if any). Otherwise, it shall terminate the bracket expression, unless it appears in a collating symbol (such as "[<sup>.</sup>]<sup>.</sup>"] or is the ending <right-square-bracket> for a collating symbol, equivalence class, or character class. The special characters '<sup>.</sup>', '<sup>\*</sup>', '<sup>[</sup>', and '\\<sup>'</sup>' (<period>, <asterisk>, <left-square-bracket>, and <backslash>, respectively) shall lose their special meaning within a bracket expression.

The character sequences "[<sup>.</sup>]", "[<sup>=</sup>]", and "[<sup>:</sup>]" (<left-square-bracket> followed by a <period>, <equals-sign>, or <colon>) shall be special inside a bracket expression and are used to delimit collating symbols, equivalence class expressions, and character class expressions. These symbols shall be followed by a valid expression and the matching terminating sequence "<sup>]</sup>", "<sup>=</sup>]", or "<sup>:</sup>]", as described in the following items.

2. A matching list expression specifies a list that shall match any single character that is matched by one of the expressions represented in the list. The first character in the list cannot be the <circumflex>. An ordinary character in the list should only match that character, but may match any single character that collates equally with that character; for example, "[abc]" is an RE that should only match one of the characters '<sup>a</sup>', '<sup>b</sup>', or '<sup>c</sup>'.

**Note:** A future version of this standard may require that an ordinary character in the list only matches that character.

It is unspecified whether a matching list expression matches a multi-character collating element that is matched by one of the expressions.

3. A non-matching list expression begins with a <circumflex> ('<sup>^</sup>'), and the matching behavior shall be the logical inverse of the corresponding matching list expression (the same bracket expression but without the leading <circumflex>). For example, if the RE

"[abc]" only matches 'a', 'b', or 'c', then "[^abc]" is an RE that matches any character except 'a', 'b', or 'c'. It is unspecified whether a non-matching list expression matches a multi-character collating element that is not matched by any of the expressions. The <circumflex> shall have this special meaning only when it occurs first in the list, immediately following the <left-square-bracket>.

4. A collating symbol is a collating element enclosed within bracket-period ("[" and ".]") delimiters. Collating elements are defined as described in [Section 7.3.2.4](#) (on page 149). Conforming applications shall represent multi-character collating elements as collating symbols when it is necessary to distinguish them from a list of the individual characters that make up the multi-character collating element. For example, if the string "ch" is a collating element defined using the line:

```
collating-element <ch-digraph> from "<c><h>"
```

in the locale definition, the expression "[.ch.]" shall be treated as an RE containing the collating symbol 'ch', while "[ch]" shall be treated as an RE matching 'c' or 'h'. Collating symbols are recognized only inside bracket expressions. If the string is not a collating element in the current locale, the expression is invalid.

5. An equivalence class expression shall represent the set of collating elements belonging to an equivalence class, as described in [Section 7.3.2.4](#) (on page 149). Only primary equivalence classes shall be recognized. The class shall be expressed by enclosing any one of the collating elements in the equivalence class within bracket-equal ("[" and "=") delimiters. For example, if 'a', 'à', and '^' belong to the same equivalence class, then "[[a=]b]", "[[à=]b]", and "[[^=]b]" are each equivalent to "[aâ^b]". If the collating element does not belong to an equivalence class, the equivalence class expression shall be treated as a collating symbol.

6. A character class expression shall represent the union of two sets:

- a. The set of single characters that belong to the character class, as defined in the *LC\_CTYPE* category in the current locale.
- b. An unspecified set of multi-character collating elements.

All character classes specified in the current locale shall be recognized. A character class expression is expressed as a character class name enclosed within bracket-colon ("[" and " : "]") delimiters.

The following character class expressions shall be supported in all locales:

```
[ :alnum: ]    [ :cntrl: ]    [ :lower: ]    [ :space: ]
[ :alpha: ]    [ :digit: ]    [ :print: ]    [ :upper: ]
[ :blank: ]    [ :graph: ]    [ :punct: ]    [ :xdigit: ]
```

In addition, character class expressions of the form:

```
[ :name: ]
```

are recognized in those locales where the *name* keyword has been given a **charclass** definition in the *LC\_CTYPE* category.

7. In the POSIX locale, a range expression represents the set of collating elements that fall between two elements in the collation sequence, inclusive. In other locales, a range expression has unspecified behavior: strictly conforming applications shall not rely on whether the range expression is valid, or on the set of collating elements matched. A range expression shall be expressed as the starting point and the ending point separated by a <hyphen-minus> ('-').

In the following, all examples assume the POSIX locale.

The starting range point and the ending range point shall be a collating element or collating symbol. An equivalence class expression used as a starting or ending point of a range expression produces unspecified results. An equivalence class can be used portably within a bracket expression, but only outside the range. If the represented set of collating elements is empty, it is unspecified whether the expression matches nothing, or is treated as invalid.

The interpretation of range expressions where the ending range point is also the starting range point of a subsequent range expression (for example, "[a-m-o]") is undefined.

The <hyphen-minus> character shall be treated as itself if it occurs first (after an initial '^', if any) or last in the list, or as an ending range point in a range expression. As examples, the expressions "[ac]" and "[ac-]" are equivalent and match any of the characters 'a', 'c', or '-'; "[^ac]" and "[^ac-]" are equivalent and match any characters except 'a', 'c', or '-'; the expression "[%-]" matches any of the characters between '%' and '-' inclusive; the expression "[-@]" matches any of the characters between '-' and '@' inclusive; and the expression "[a--@]" is either invalid or equivalent to '@', because the letter 'a' follows the symbol '-' in the POSIX locale. To use a <hyphen-minus> as the starting range point, it shall either come first in the bracket expression or be specified as a collating symbol; for example, "[[. . ]-0]", which matches either a <right-square-bracket> or any character or collating element that collates between <hyphen-minus> and 0, inclusive.

If a bracket expression specifies both '-' and ']', the ']' shall be placed first (after the '^', if any) and the '-' last within the bracket expression.

8. If a bracket expression contains at least three list elements, where the first and last list elements are the same single-character element of <period>, <equals-sign>, or <colon>, then it is unspecified whether the bracket expression will be treated as a collating symbol, equivalence class, or character class, respectively; treated as a matching list expression; or rejected as an error.

### 9.3.6 BREs Matching Multiple Characters

The following rules can be used to construct BREs matching multiple characters from BREs matching a single character:

1. The concatenation of BREs shall match the concatenation of the strings matched by each component of the BRE.
2. A subexpression can be defined within a BRE by enclosing it between the character pairs "\(" and "\)". Such a subexpression shall match whatever it would have matched without the "\(" and "\)", except that anchoring within subexpressions is optional behavior; see [Section 9.3.8](#) (on page 188). Subexpressions can be arbitrarily nested.
3. The back-reference expression '\n' shall match the same (possibly empty) string of characters as was matched by a subexpression enclosed between "\(" and "\)" preceding the '\n'. The character 'n' shall be a digit from 1 through 9, specifying the *n*th subexpression (the one that begins with the *n*th "\(" from the beginning of the pattern and ends with the corresponding paired "\)"). The expression is invalid if less than *n* subexpressions precede the '\n'. The string matched by a contained subexpression shall be within the string matched by the containing subexpression. If the containing subexpression does not match, or if there is no match for the contained subexpression within the string matched by the containing subexpression, then back-



reference expressions corresponding to the contained subexpression shall not match. When a subexpression matches more than one string, a back-reference expression corresponding to the subexpression shall refer to the last matched string. For example, the expression `"^\. *\) \1$"` matches strings consisting of two adjacent appearances of the same substring, and the expression `"^(a\) * \1"` fails to match `'a'`, the expression `"\ (a \(b\) *\) * \2"` fails to match `'abab'`, and the expression `"^ \(ab *\) * \1$"` matches `'ababbabb'`, but fails to match `'ababbab'`.

4. When a BRE matching a single character, a subexpression, or a back-reference is followed by the special character `<asterisk>` (`' * '`), together with that `<asterisk>` it shall match what zero or more consecutive occurrences of the BRE would match. For example, `"[ab] *"` and `"[ab][ab]"` are equivalent when matching the string `"ab"`.

5. When a BRE matching a single character, a subexpression, or a back-reference is followed by an interval expression of the format `"\{m\}"`, `"\{m,\}"`, or `"\{m,n\}"`, together with that interval expression it shall match what repeated consecutive occurrences of the BRE would match. The values of  $m$  and  $n$  are decimal integers in the range  $0 \leq m \leq n \leq \{RE\_DUP\_MAX\}$ , where  $m$  specifies the exact or minimum number of occurrences and  $n$  specifies the maximum number of occurrences. The expression `"\{m\}"` shall match exactly  $m$  occurrences of the preceding BRE, `"\{m,\}"` shall match at least  $m$  occurrences, and `"\{m,n\}"` shall match any number of occurrences between  $m$  and  $n$ , inclusive.

For example, in the string `"abababcccccd"` the BRE `"c\{3\}"` is matched by characters seven to nine, the BRE `"\ (ab\) \{4,\}"` is not matched at all, and the BRE `"c\{1,3\}d"` is matched by characters ten to thirteen.

The behavior of multiple adjacent duplication symbols (`' * '` and intervals) produces undefined results.

A subexpression repeated by an `<asterisk>` (`' * '`) or an interval expression shall not match a null expression unless this is the only match for the repetition or it is necessary to satisfy the exact or minimum number of occurrences for the interval expression.

### 9.3.7 BRE Precedence

The order of precedence shall be as shown in the following table:

BRE Precedence (from high to low)	
Collation-related bracket symbols	<code>[==]</code> <code>[::]</code> <code>[..]</code>
Escaped characters	<code>\&lt;special character&gt;</code>
Bracket expression	<code>[]</code>
Subexpressions/back-references	<code>\( \)</code> <code>\n</code>
Single-character-BRE duplication	<code>* \{m,n\}</code>
Concatenation	
Anchoring	<code>^</code> <code>\$</code>

### 9.3.8 BRE Expression Anchoring

A BRE can be limited to matching expressions that begin or end a string; this is called “anchoring”. The <circumflex> and <dollar-sign> special characters shall be considered BRE anchors in the following contexts:

1. A <circumflex> ( ' ^ ' ) shall be an anchor when used as the first character of an entire BRE. The implementation may treat the <circumflex> as an anchor when used as the first character of a subexpression. The <circumflex> shall anchor the expression (or optionally subexpression) to the beginning of a string; only sequences starting at the first character of a string shall be matched by the BRE. For example, the BRE " ^ ab " matches " ab " in the string " abcdef ", but fails to match in the string " cdefab ". The BRE " \ ( ^ ab \ ) " may match the former string. A portable BRE shall escape a leading <circumflex> in a subexpression to match a literal circumflex.
2. A <dollar-sign> ( ' \$ ' ) shall be an anchor when used as the last character of an entire BRE. The implementation may treat a <dollar-sign> as an anchor when used as the last character of a subexpression. The <dollar-sign> shall anchor the expression (or optionally subexpression) to the end of the string being matched; the <dollar-sign> can be said to match the end-of-string following the last character.
3. A BRE anchored by both ' ^ ' and ' \$ ' shall match only an entire string. For example, the BRE " ^ abcdef \$ " matches strings consisting only of " abcdef ".

## 9.4 Extended Regular Expressions

The extended regular expression (ERE) notation and construction rules shall apply to utilities defined as using extended regular expressions; any exceptions to the following rules are noted in the descriptions of the specific utilities using EREs.

### 9.4.1 EREs Matching a Single Character or Collating Element

An ERE ordinary character, a special character preceded by a <backslash> or a <period> shall match a single character. A bracket expression shall match a single character or a single collating element. An ERE matching a single character enclosed in parentheses shall match the same as the ERE without parentheses would have matched.

### 9.4.2 ERE Ordinary Characters

An ordinary character is an ERE that matches itself. An ordinary character is any character in the supported character set, except for the ERE special characters listed in [Section 9.4.3](#) (on page 189). The interpretation of an ordinary character preceded by an unescaped <backslash> ( ' \ ' ) is undefined, except in the context of a bracket expression (see [Section 9.4.5](#), on page 189).



### 9.4.3 ERE Special Characters

An ERE special character has special properties in certain contexts. Outside those contexts, or when preceded by a <backslash>, such a character shall be an ERE that matches the special character itself. The extended regular expression special characters and the contexts in which they shall have their special meaning are as follows:

- . [ \ ( The <period>, <left-square-bracket>, <backslash>, and <left-parenthesis> shall be special except when used in a bracket expression (see [Section 9.3.5](#), on page 184). Outside a bracket expression, a <left-parenthesis> immediately followed by a <right-parenthesis> produces undefined results. A <left-square-bracket> that is unescaped + and is not part of a bracket expression also produces undefined results.
- ) The <right-parenthesis> shall be special when matched with a preceding <left-parenthesis>, both outside a bracket expression.
- \* + ? { The <asterisk>, <plus-sign>, <question-mark>, and <left-brace> shall be special except when used in a bracket expression (see [Section 9.3.5](#), on page 184). Any of the following uses produce undefined results:
  - If these characters appear first in an ERE, or immediately following an unescaped | <vertical-line>, <circumflex>, <dollar-sign>, or <left-parenthesis>
  - If a <left-brace> is not part of a valid interval expression (see [Section 9.4.6](#), on page 190)
- | The <vertical-line> is special except when used in a bracket expression (see [Section 9.3.5](#), on page 184). A <vertical-line> appearing first or last in an ERE, or immediately following a <vertical-line> or a <left-parenthesis>, or immediately preceding a <right-parenthesis>, produces undefined results.
- ^ The <circumflex> shall be special when used as an anchor (see [Section 9.4.9](#), on page 191). The <circumflex> shall signify a non-matching list expression when it occurs first | in a list, immediately following a <left-square-bracket> (see [Section 9.3.5](#), on page 184).
- \$ The <dollar-sign> shall be special when used as an anchor.

### 9.4.4 Periods in EREs

A <period> ( ' . ' ), when used outside a bracket expression, is an ERE that shall match any character in the supported character set except NUL.

### 9.4.5 ERE Bracket Expression

The rules for ERE Bracket Expressions are the same as for Basic Regular Expressions; see [Section 9.3.5](#) (on page 184).

#### 9.4.6 EREs Matching Multiple Characters

The following rules shall be used to construct EREs matching multiple characters from EREs matching a single character:

1. A concatenation of EREs shall match the concatenation of the character sequences matched by each component of the ERE. A concatenation of EREs enclosed in parentheses shall match whatever the concatenation without the parentheses matches. For example, both the ERE "cd" and the ERE "(cd)" are matched by the third and fourth character of the string "abcdefabcdef".
2. When an ERE matching a single character or an ERE enclosed in parentheses is followed by the special character <plus-sign> ('+'), together with that <plus-sign> it shall match what one or more consecutive occurrences of the ERE would match. For example, the ERE "b+(bc)" matches the fourth to seventh characters in the string "acabbbbcde". And, "[ab]+" and "[ab][ab]\*" are equivalent.
3. When an ERE matching a single character or an ERE enclosed in parentheses is followed by the special character <asterisk> ('\*'), together with that <asterisk> it shall match what zero or more consecutive occurrences of the ERE would match. For example, the ERE "b\*c" matches the first character in the string "cabbbbcde", and the ERE "b\*cd" matches the third to seventh characters in the string "cabbbbcdebbbbbbbcdbc". And, "[ab]\*" and "[ab][ab]" are equivalent when matching the string "ab".
4. When an ERE matching a single character or an ERE enclosed in parentheses is followed by the special character <question-mark> ('?'), together with that <question-mark> it shall match what zero or one consecutive occurrences of the ERE would match. For example, the ERE "b?c" matches the second character in the string "acabbbbcde".
5. When an ERE matching a single character or an ERE enclosed in parentheses is followed by an interval expression of the format "{m}", "{m,}", or "{m,n}", together with that interval expression it shall match what repeated consecutive occurrences of the ERE would match. The values of *m* and *n* are decimal integers in the range  $0 \leq m \leq n \leq \text{RE\_DUP\_MAX}$ , where *m* specifies the exact or minimum number of occurrences and *n* specifies the maximum number of occurrences. The expression "{m}" matches exactly *m* occurrences of the preceding ERE, "{m,}" matches at least *m* occurrences, and "{m,n}" matches any number of occurrences between *m* and *n*, inclusive.  
  
For example, in the string "abababcccccccd" the ERE "c{3}" is matched by characters seven to nine and the ERE "(ab){2,}" is matched by characters one to six.

The behavior of multiple adjacent duplication symbols ('+', '\*', '?', and intervals) produces undefined results.

An ERE matching a single character repeated by an '\*', '?', or an interval expression shall not match a null expression unless this is the only match for the repetition or it is necessary to satisfy the exact or minimum number of occurrences for the interval expression.

#### 9.4.7 ERE Alternation

Two EREs separated by the special character <vertical-line> ( ' | ' ) shall match a string that is matched by either. For example, the ERE "a( (bc) | d )" matches the string "abc" and the string "ad". Single characters, or expressions matching single characters, separated by the <vertical-line> and enclosed in parentheses, shall be treated as an ERE matching a single character.

#### 9.4.8 ERE Precedence

The order of precedence shall be as shown in the following table:

ERE Precedence (from high to low)	
Collation-related bracket symbols	[==] [::] [..]
Escaped characters	\<special character>
Bracket expression	[ ]
Grouping	( )
Single-character-ERE duplication	* + ? {m,n}
Concatenation	
Anchoring	^ \$
Alternation	

For example, the ERE "abba|cde" matches either the string "abba" or the string "cde" (rather than the string "abbade" or "abbcde", because concatenation has a higher order of precedence than alternation).

#### 9.4.9 ERE Expression Anchoring

An ERE can be limited to matching expressions that begin or end a string; this is called "anchoring". The <circumflex> and <dollar-sign> special characters shall be considered ERE anchors when used anywhere outside a bracket expression. This shall have the following effects:

1. A <circumflex> ( ' ^ ' ) outside a bracket expression shall anchor the expression or subexpression it begins to the beginning of a string; such an expression or subexpression can match only a sequence starting at the first character of a string. For example, the EREs " ^ ab " and " ( ^ ab )" match "ab" in the string "abcdef", but fail to match in the string "cdefab", and the ERE "a ^ b" is valid, but can never match because the 'a' prevents the expression " ^ b" from matching starting at the first character.
2. A <dollar-sign> ( ' \$ ' ) outside a bracket expression shall anchor the expression or subexpression it ends to the end of a string; such an expression or subexpression can match only a sequence ending at the last character of a string. For example, the EREs " ef \$ " and " ( ef \$ )" match "ef" in the string "abcdef", but fail to match in the string "cdefab", and the ERE "e \$ f" is valid, but can never match because the 'f' prevents the expression "e \$ " from matching ending at the last character.

## 9.5 Regular Expression Grammar

Grammars describing the syntax of both basic and extended regular expressions are presented in this section. The grammar takes precedence over the text. See XCU [Section 1.3](#) (on page 2325).

### 9.5.1 BRE/ERE Grammar Lexical Conventions

The lexical conventions for regular expressions are as described in this section.

Except as noted, the longest possible token or delimiter beginning at a given point is recognized.

The following tokens are processed (in addition to those string constants shown in the grammar):

<b>COLL_ELEM_SINGLE</b>	Any single-character collating element, unless it is a <b>META_CHAR</b> .
<b>COLL_ELEM_MULTI</b>	Any multi-character collating element.
<b>BACKREF</b>	Applicable only to basic regular expressions. The character string consisting of a <backslash> character followed by a single-digit numeral, '1' to '9'.
<b>DUP_COUNT</b>	Represents a numeric constant. It shall be an integer in the range 0 ≤ <b>DUP_COUNT</b> ≤ { <b>RE_DUP_MAX</b> }. This token is only recognized when the context of the grammar requires it. At all other times, digits not preceded by a <backslash> character are treated as <b>ORD_CHAR</b> .
<b>META_CHAR</b>	One of the characters: <ul style="list-style-type: none"> <li>^ When found first in a bracket expression</li> <li>– When found anywhere but first (after an initial '^', if any) or last in a bracket expression, or as the ending range point in a range expression</li> <li>] When found anywhere but first (after an initial '^', if any) in a bracket expression</li> </ul>
<b>L_ANCHOR</b>	Applicable only to basic regular expressions. The character '^' when it appears as the first character of a basic regular expression and when not <b>QUOTED_CHAR</b> . The '^' may be recognized as an anchor elsewhere; see <a href="#">Section 9.3.8</a> (on page 188).
<b>ORD_CHAR</b>	A character, other than one of the special characters in <b>SPEC_CHAR</b> .
<b>QUOTED_CHAR</b>	In a BRE, one of the character sequences: <ul style="list-style-type: none"> <li>\^ \. \* \[ \\$ \\</li> </ul> In an ERE, one of the character sequences: <ul style="list-style-type: none"> <li>\^ \. \[ \\$ \( \) \ </li> <li>\* \+ \? \{ \\</li> </ul>
<b>R_ANCHOR</b>	(Applicable only to basic regular expressions.) The character '\$' when it appears as the last character of a basic regular expression and when not <b>QUOTED_CHAR</b> . The '\$' may be recognized as an anchor elsewhere; see <a href="#">Section 9.3.8</a> (on page 188).

6416       **SPEC\_CHAR**                   For basic regular expressions, one of the following special characters:

6417                   .           Anywhere outside bracket expressions

6418                   \           Anywhere outside bracket expressions

6419                   [           Anywhere outside bracket expressions

6420                   ^           When used as an anchor (see [Section 9.3.8](#), on page 188)       -

6421                   \$           When used as an anchor

6422                   \*           Anywhere except first in an entire RE, anywhere in a

6423                               bracket expression, directly following "\(", directly

6424                               following an anchoring '^'

6425                   For extended regular expressions, shall be one of the following

6426                   special characters found anywhere outside bracket expressions:

6427                   ^       .       [       \$       (       )       |

6428                   \*       +       ?       {       \

6429                   (The close-parenthesis shall be considered special in this context only

6430                   if matched with a preceding open-parenthesis.)

## 6431   **9.5.2 RE and Bracket Expression Grammar**

6432       This section presents the grammar for basic regular expressions, including the bracket

6433       expression grammar that is common to both BREs and EREs.

6434       %token     ORD\_CHAR QUOTED\_CHAR DUP\_COUNT

6435       %token     BACKREF L\_ANCHOR R\_ANCHOR

6436       %token     Back\_open\_paren   Back\_close\_paren

6437       /\*         '\('                '\)'               \*/

6438       %token     Back\_open\_brace    Back\_close\_brace

6439       /\*         '\{'                '\}'               \*/

6440       /\* The following tokens are for the Bracket Expression

6441       grammar common to both REs and EREs. \*/

6442       %token     COLL\_ELEM\_SINGLE COLL\_ELEM\_MULTI META\_CHAR

6443       %token     Open\_equal   Equal\_close   Open\_dot   Dot\_close   Open\_colon   Colon\_close

6444       /\*         '['        '='        '['        '.'        ']'        ':'        ']'        \*/

6445       %token     class\_name

6446       /\* class\_name is a keyword to the LC\_CTYPE locale category \*/

6447       /\* (representing a character class) in the current locale \*/

6448       /\* and is only recognized between [: and :] \*/

6449       %start     basic\_reg\_exp

6450       %%

6451       /\* -----

6452       Basic Regular Expression

6453       -----

6454       \*/

6455       basic\_reg\_exp :               RE\_expression

6456       | L\_ANCHOR

```

6457             | R_ANCHOR
6458             | L_ANCHOR R_ANCHOR
6459             | L_ANCHOR RE_expression
6460             | RE_expression R_ANCHOR
6461             | L_ANCHOR RE_expression R_ANCHOR
6462             ;
6463 RE_expression : simple_RE
6464             | RE_expression simple_RE
6465             ;
6466 simple_RE    : nondupl_RE
6467             | nondupl_RE RE_dupl_symbol
6468             ;
6469 nondupl_RE   : one_char_or_coll_elem_RE
6470             | Back_open_paren RE_expression Back_close_paren
6471             | BACKREF
6472             ;
6473 one_char_or_coll_elem_RE : ORD_CHAR
6474             | QUOTED_CHAR
6475             | '.'
6476             | bracket_expression
6477             ;
6478 RE_dupl_symbol : '*'
6479             | Back_open_brace DUP_COUNT Back_close_brace
6480             | Back_open_brace DUP_COUNT ',' Back_close_brace
6481             | Back_open_brace DUP_COUNT ',' DUP_COUNT Back_close_brace
6482             ;
6483 /* -----
6484    Bracket Expression
6485    -----
6486 */
6487 bracket_expression : '[' matching_list ']'
6488                 | '[' nonmatching_list ']'
6489                 ;
6490 matching_list    : bracket_list
6491                 ;
6492 nonmatching_list : '^' bracket_list
6493                 ;
6494 bracket_list     : follow_list
6495                 | follow_list '-'
6496                 ;
6497 follow_list      : expression_term
6498                 | follow_list expression_term
6499                 ;
6500 expression_term  : single_expression
6501                 | range_expression
6502                 ;
6503 single_expression : end_range
6504                 | character_class
6505                 | equivalence_class
6506                 ;
6507 range_expression : start_range end_range
6508                 | start_range '-'

```

```

6509         ;
6510     start_range      : end_range '-'
6511         ;
6512     end_range        : COLL_ELEM_SINGLE
6513         | collating_symbol
6514         ;
6515     collating_symbol : Open_dot COLL_ELEM_SINGLE Dot_close
6516         | Open_dot COLL_ELEM_MULTI Dot_close
6517         | Open_dot META_CHAR Dot_close
6518         ;
6519     equivalence_class : Open_equal COLL_ELEM_SINGLE Equal_close
6520         | Open_equal COLL_ELEM_MULTI Equal_close
6521         ;
6522     character_class  : Open_colon class_name Colon_close
6523         ;

```

The BRE grammar does not permit **L\_ANCHOR** or **R\_ANCHOR** inside "**\(**" and "**\)**" (which implies that '**^**' and '**\$**' are ordinary characters). This reflects the semantic limits on the application, as noted in [Section 9.3.8](#) (on page 188). Implementations are permitted to extend the language to interpret '**^**' and '**\$**' as anchors in these locations, and as such, conforming applications cannot use unescaped '**^**' and '**\$**' in positions inside "**\(**" and "**\)**" that might be interpreted as anchors.

### 9.5.3 ERE Grammar

This section presents the grammar for extended regular expressions, excluding the bracket expression grammar.

**Note:** The bracket expression grammar and the associated **%token** lines are identical between BREs and EREs. It has been omitted from the ERE section to avoid unnecessary editorial duplication.

```

6535 %token  ORD_CHAR QUOTED_CHAR DUP_COUNT
6536 %start  extended_reg_exp
6537 %%
6538 /* -----
6539     Extended Regular Expression
6540     -----
6541 */
6542 extended_reg_exp      : ERE_branch
6543         | extended_reg_exp '|' ERE_branch
6544         ;
6545 ERE_branch            : ERE_expression
6546         | ERE_branch ERE_expression
6547         ;
6548 ERE_expression        : one_char_or_coll_elem_ERE
6549         | '^'
6550         | '$'
6551         | '(' extended_reg_exp ')'
6552         | ERE_expression ERE_dupl_symbol
6553         ;
6554 one_char_or_coll_elem_ERE : ORD_CHAR
6555         | QUOTED_CHAR
6556         | '.'

```

```

6557         | bracket_expression
6558         ;
6559 ERE_dupl_symbol : ' * '
6560                 | ' + '
6561                 | ' ? '
6562                 | ' { ' DUP_COUNT ' } '
6563                 | ' { ' DUP_COUNT ' , ' ' } '
6564                 | ' { ' DUP_COUNT ' , ' DUP_COUNT ' } '
6565                 ;

```

The ERE grammar does not permit several constructs that previous sections specify as having undefined results. Additionally, there are some constructs which the grammar permits but which still give undefined results:

- **ORD\_CHAR** preceded by an unescaped <backslash> character
- One or more *ERE\_dupl\_symbols* appearing first in an ERE, or immediately following ' | ', '^ ', ' ( ', or ' \$ '
- ' { ' not part of a valid *ERE\_dupl\_symbol*
- ' | ' appearing first or last in an ERE, or immediately following ' | ' or ' ( ', or immediately preceding ' ) '

Implementations are permitted to extend the language to allow these. Strictly Conforming applications cannot use such constructs.



6577

Chapter 10

6578

# Directory Structure and Devices

6579

## 10.1 Directory Structure and Files

6580

6581

6582

The following directories shall exist on conforming systems and conforming applications shall make use of them only as described. Strictly conforming applications shall not assume the ability to create files in any of these directories, unless specified below.

6583

**/** The root directory.

6584

**/dev** Contains **/dev/console**, **/dev/null**, and **/dev/tty**, described below.

6585

The following directory shall exist on conforming systems and shall be used as described:

6586

6587

6588

**/tmp** A directory made available for applications that need a place to create temporary files. Applications shall be allowed to create files in this directory, but shall not assume that such files are preserved between invocations of the application.

6589

The following files shall exist on conforming systems and shall be both readable and writable:

6590

6591

**/dev/null** An empty data source and infinite data sink. Data written to **/dev/null** shall be discarded. Reads from **/dev/null** shall always return end-of-file (EOF).

6592

6593

6594

6595

6596

6597

**/dev/tty** In each process, a synonym for the controlling terminal associated with the process group of that process, if any. It is useful for programs or shell procedures that wish to be sure of writing messages to or reading data from the terminal no matter how output has been redirected. It can also be used for applications that demand the name of a file for output, when typed output is desired and it is tiresome to find out what terminal is currently in use.

6598

The following file shall exist on conforming systems and need not be readable or writable:

6599

6600

6601

6602

**/dev/console** The **/dev/console** file is a generic name given to the system console (see [Section 3.392](#), on page 97). It is usually linked to an implementation-defined special file. It shall provide an interface to the system console conforming to the requirements of [Chapter 11](#) (on page 199).

## 10.2 Output Devices and Terminal Types

The utilities in the Shell and Utilities volume of POSIX.1-2008 historically have been implemented on a wide range of terminal types, but a conforming implementation need not support all features of all utilities on every conceivable terminal. POSIX.1-2008 states which features are optional for certain classes of terminals in the individual utility description sections. The implementation shall document in the system documentation which terminal types it supports and which of these features and utilities are not supported by each terminal.

When a feature or utility is not supported on a specific terminal type, as allowed by POSIX.1-2008, and the implementation considers such a condition to be an error preventing use of the feature or utility, the implementation shall indicate such conditions through diagnostic messages or exit status values or both (as appropriate to the specific utility description) that inform the user that the terminal type lacks the appropriate capability.

POSIX.1-2008 uses a notational convention based on historical practice that identifies some of the control characters defined in [Section 7.3.1](#) (on page 139) in a manner easily remembered by users on many terminals. The correspondence between this “<control>-char” notation and the actual control characters is shown in the following table. When POSIX.1-2008 refers to a character by its <control>-name, it is referring to the actual control character shown in the Value column of the table, which is not necessarily the exact control key sequence on all terminals. Some terminals have keyboards that do not allow the direct transmission of all the non-alphanumeric characters shown. In such cases, the system documentation shall describe which data sequences transmitted by the terminal are interpreted by the system as representing the special characters.

**Table 10-1** Control Character Names

Name	Value	Name	Value
<control>-A	<SOH>	<control>-Q	<DC1>
<control>-B	<STX>	<control>-R	<DC2>
<control>-C	<ETX>	<control>-S	<DC3>
<control>-D	<EOT>	<control>-T	<DC4>
<control>-E	<ENQ>	<control>-U	<NAK>
<control>-F	<ACK>	<control>-V	<SYN>
<control>-G	<BEL>	<control>-W	<ETB>
<control>-H	<BS>	<control>-X	<CAN>
<control>-I	<HT>	<control>-Y	<EM>
<control>-J	<LF>	<control>-Z	<SUB>
<control>-K	<VT>	<control>-[	<ESC>
<control>-L	<FF>	<control>-\ <control>-]	<FS>
<control>-M	<CR>	<control>-^	<GS>
<control>-N	<SO>	<control>-_	<RS>
<control>-O	<SI>	<control>-?	<US>
<control>-P	<DLE>		<DEL>

**Note:** The notation uses uppercase letters for arbitrary editorial reasons. There is no implication that the keystrokes represent control-shift-letter sequences.

# General Terminal Interface

This chapter describes a general terminal interface that shall be provided. It shall be supported on any asynchronous communications ports if the implementation provides them. It is implementation-defined whether it supports network connections or synchronous ports, or both.

## 11.1 Interface Characteristics

### 11.1.1 Opening a Terminal Device File

When a terminal device file is opened, it normally causes the thread to wait until a connection is established. In practice, application programs seldom open these files; they are opened by special programs and become an application's standard input, output, and error files.

Cases where applications do open a terminal device are as follows:

1. Opening `/dev/tty`, or the pathname returned by `ctermid()`, in order to obtain a file descriptor for the controlling terminal; see [Section 11.1.3](#) (on page 200).
2. Opening the slave side of a pseudo-terminal; see XSH [ptsname\(\)](#).
3. Opening a modem or similar piece of equipment connected by a serial line. In this case, the terminal parameters (see [Section 11.2](#), on page 205) may be initialized to default settings by the implementation in between the last close of the device by any process and the next open of the device, or they may persist from one use to the next. The terminal parameters can be set to values that ensure the terminal behaves in a conforming manner by means of the `O_TTY_INIT` open flag when opening a terminal device that is not already open in any process, or by executing the `stty` utility with the operand `sane`.

As described in `open()`, opening a terminal device file with the `O_NONBLOCK` flag clear shall cause the thread to block until the terminal device is ready and available. If `CLOCAL` mode is not set, this means blocking until a connection is established. If `CLOCAL` mode is set in the terminal, or the `O_NONBLOCK` flag is specified in the `open()`, the `open()` function shall return a file descriptor without waiting for a connection to be established.

### 11.1.2 Process Groups

A terminal may have a foreground process group associated with it. This foreground process group plays a special role in handling signal-generating input characters, as discussed in [Section 11.1.9](#) (on page 204).

A command interpreter process supporting job control can allocate the terminal to different jobs, or process groups, by placing related processes in a single process group and associating this process group with the terminal. A terminal's foreground process group may be set or examined by a process, assuming the permission requirements are met; see *tcgetpgrp()* and *tcsetpgrp()*. The terminal interface aids in this allocation by restricting access to the terminal by processes that are not in the current process group; see [Section 11.1.4](#) (on page 201).

When there is no longer any process whose process ID or process group ID matches the foreground process group ID, the terminal shall have no foreground process group. It is unspecified whether the terminal has a foreground process group when there is a process whose process ID matches the foreground process group ID, but whose process group ID does not. No actions defined in POSIX.1-2008, other than allocation of a controlling terminal or a successful call to *tcsetpgrp()*, shall cause a process group to become the foreground process group of the terminal.

### 11.1.3 The Controlling Terminal

A terminal may belong to a process as its controlling terminal. Each process of a session that has a controlling terminal has the same controlling terminal. A terminal may be the controlling terminal for at most one session. The controlling terminal for a session is allocated by the session leader in an implementation-defined manner. If a session leader has no controlling terminal, and opens a terminal device file that is not already associated with a session without using the *O\_NOCTTY* option (see *open()*), it is implementation-defined whether the terminal becomes the controlling terminal of the session leader. If a process which is not a session leader opens a terminal file, or the *O\_NOCTTY* option is used on *open()*, then that terminal shall not become the controlling terminal of the calling process. When a controlling terminal becomes associated with a session, its foreground process group shall be set to the process group of the session leader.

The controlling terminal is inherited by a child process during a *fork()* function call. A process relinquishes its controlling terminal when it creates a new session with the *setsid()* function; other processes remaining in the old session that had this terminal as their controlling terminal continue to have it. Upon the close of the last file descriptor in the system (whether or not it is in the current session) associated with the controlling terminal, it is unspecified whether all processes that had that terminal as their controlling terminal cease to have any controlling terminal. Whether and how a session leader can reacquire a controlling terminal after the controlling terminal has been relinquished in this fashion is unspecified. A process does not relinquish its controlling terminal simply by closing all of its file descriptors associated with the controlling terminal if other processes continue to have it open.

When a controlling process terminates, the controlling terminal is dissociated from the current session, allowing it to be acquired by a new session leader. Subsequent access to the terminal by other processes in the earlier session may be denied, with attempts to access the terminal treated as if a modem disconnect had been sensed.

#### 11.1.4 Terminal Access Control

If a process is in the foreground process group of its controlling terminal, read operations shall be allowed, as described in [Section 11.1.5](#). Any attempts by a process in a background process group to read from its controlling terminal cause its process group to be sent a SIGTTIN signal unless one of the following special cases applies: if the reading process is ignoring the SIGTTIN signal or the reading thread is blocking the SIGTTIN signal, or if the process group of the reading process is orphaned, the *read()* shall return -1, with *errno* set to [EIO] and no signal shall be sent. The default action of the SIGTTIN signal shall be to stop the process to which it is sent. See [<signal.h>](#).

If a process is in the foreground process group of its controlling terminal, write operations shall be allowed as described in [Section 11.1.8](#) (on page 203). Attempts by a process in a background process group to write to its controlling terminal shall cause the process group to be sent a SIGTTOU signal unless one of the following special cases applies: if TOSTOP is not set, or if TOSTOP is set and the process is ignoring the SIGTTOU signal or the writing thread is blocking the SIGTTOU signal, the process is allowed to write to the terminal and the SIGTTOU signal is not sent. If TOSTOP is set, the process group of the writing process is orphaned, the writing process is not ignoring the SIGTTOU signal, and the writing thread is not blocking the SIGTTOU signal, the *write()* shall return -1, with *errno* set to [EIO] and no signal shall be sent.

Certain calls that set terminal parameters are treated in the same fashion as *write()*, except that TOSTOP is ignored; that is, the effect is identical to that of terminal writes when TOSTOP is set (see [Section 11.2.5](#) (on page 210), *tcdrain()*, *tcflow()*, *tcflush()*, *tcsendbreak()*, *tcsetattr()*, and *tcsetpgrp()*).

#### 11.1.5 Input Processing and Reading Data

A terminal device associated with a terminal device file may operate in full-duplex mode, so that data may arrive even while output is occurring. Each terminal device file has an input queue associated with it, into which incoming data is stored by the system before being read by a process. The system may impose a limit, {MAX\_INPUT}, on the number of bytes that may be stored in the input queue. The behavior of the system when this limit is exceeded is implementation-defined.

Two general kinds of input processing are available, determined by whether the terminal device file is in canonical mode or non-canonical mode. These modes are described in [Section 11.1.6](#) (on page 202) and [Section 11.1.7](#) (on page 202). Additionally, input characters are processed according to the *c\_iflag* (see [Section 11.2.2](#), on page 206) and *c\_lflag* (see [Section 11.2.5](#), on page 210) fields. Such processing can include “echoing”, which in general means transmitting input characters immediately back to the terminal when they are received from the terminal. This is useful for terminals that can operate in full-duplex mode.

The manner in which data is provided to a process reading from a terminal device file is dependent on whether the terminal file is in canonical or non-canonical mode, and on whether or not the O\_NONBLOCK flag is set by *open()* or *fcntl()*.

If the O\_NONBLOCK flag is clear, then the read request shall be blocked until data is available or a signal has been received. If the O\_NONBLOCK flag is set, then the read request shall be completed, without blocking, in one of three ways:

1. If there is enough data available to satisfy the entire request, the *read()* shall complete successfully and shall return the number of bytes read.

2. If there is not enough data available to satisfy the entire request, the `read()` shall complete successfully, having read as much data as possible, and shall return the number of bytes it was able to read.

3. If there is no data available, the `read()` shall return `-1`, with `errno` set to `[EAGAIN]`.

When data is available depends on whether the input processing mode is canonical or non-canonical. [Section 11.1.6](#) and [Section 11.1.7](#) describe each of these input processing modes.

### 11.1.6 Canonical Mode Input Processing

In canonical mode input processing, terminal input is processed in units of lines. A line is delimited by a <newline> character (NL), an end-of-file character (EOF), or an end-of-line (EOL) character. See [Section 11.1.9](#) (on page 204) for more information on EOF and EOL. This means that a read request shall not return until an entire line has been typed or a signal has been received. Also, no matter how many bytes are requested in the `read()` call, at most one line shall be returned. It is not, however, necessary to read a whole line at once; any number of bytes, even one, may be requested in a `read()` without losing information.

If `{MAX_CANON}` is defined for this terminal device, it shall be a limit on the number of bytes in a line. The behavior of the system when this limit is exceeded is implementation-defined. If `{MAX_CANON}` is not defined, there shall be no such limit; see `pathconf()`.

Erase and kill processing occur when either of two special characters, the ERASE and KILL characters (see [Section 11.1.9](#), on page 204), is received. This processing shall affect data in the input queue that has not yet been delimited by an NL, EOF, or EOL character. This un-delimited data makes up the current line. The ERASE character shall delete the last character in the current line, if there is one. The KILL character shall delete all data in the current line, if there is any. The ERASE and KILL characters shall have no effect if there is no data in the current line. The ERASE and KILL characters themselves shall not be placed in the input queue.

### 11.1.7 Non-Canonical Mode Input Processing

In non-canonical mode input processing, input bytes are not assembled into lines, and erase and kill processing shall not occur. The values of the MIN and TIME members of the `c_cc` array are used to determine how to process the bytes received. POSIX.1-2008 does not specify whether the setting of `O_NONBLOCK` takes precedence over MIN or TIME settings. Therefore, if `O_NONBLOCK` is set, `read()` may return immediately, regardless of the setting of MIN or TIME. Also, if no data is available, `read()` may either return 0, or return `-1` with `errno` set to `[EAGAIN]`.

MIN represents the minimum number of bytes that should be received when the `read()` function returns successfully. TIME is a timer of 0.1 second granularity that is used to time out bursty and short-term data transmissions. If MIN is greater than `{MAX_INPUT}`, the response to the request is undefined. The four possible values for MIN and TIME and their interactions are described below.



**Case A: MIN>0, TIME>0**

In case A, TIME serves as an inter-byte timer which shall be activated after the first byte is received. Since it is an inter-byte timer, it shall be reset after a byte is received. The interaction between MIN and TIME is as follows. As soon as one byte is received, the inter-byte timer shall be started. If MIN bytes are received before the inter-byte timer expires (remember that the timer is reset upon receipt of each byte), the read shall be satisfied. If the timer expires before MIN bytes are received, the characters received to that point shall be returned to the user. Note that if TIME expires at least one byte shall be returned because the timer would not have been enabled unless a byte was received. In this case (MIN>0, TIME>0) the read shall block until the MIN and TIME mechanisms are activated by the receipt of the first byte, or a signal is received. If data is in the buffer at the time of the *read()*, the result shall be as if data has been received immediately after the *read()*.

**Case B: MIN>0, TIME=0**

In case B, since the value of TIME is zero, the timer plays no role and only MIN is significant. A pending read shall not be satisfied until MIN bytes are received (that is, the pending read shall block until MIN bytes are received), or a signal is received. A program that uses case B to read record-based terminal I/O may block indefinitely in the read operation.

**Case C: MIN=0, TIME>0**

In case C, since MIN=0, TIME no longer represents an inter-byte timer. It now serves as a read timer that shall be activated as soon as the *read()* function is processed. A read shall be satisfied as soon as a single byte is received or the read timer expires. Note that in case C if the timer expires, no bytes shall be returned. If the timer does not expire, the only way the read can be satisfied is if a byte is received. If bytes are not received, the read shall not block indefinitely waiting for a byte; if no byte is received within TIME\*0.1 seconds after the read is initiated, the *read()* shall return a value of zero, having read no data. If data is in the buffer at the time of the *read()*, the timer shall be started as if data has been received immediately after the *read()*.

**Case D: MIN=0, TIME=0**

The minimum of either the number of bytes requested or the number of bytes currently available shall be returned without waiting for more bytes to be input. If no characters are available, *read()* shall return a value of zero, having read no data.

### 11.1.8 Writing Data and Output Processing

When a process writes one or more bytes to a terminal device file, they are processed according to the *c\_oflag* field (see [Section 11.2.3](#), on page 207). The implementation may provide a buffering mechanism; as such, when a call to *write()* completes, all of the bytes written have been scheduled for transmission to the device, but the transmission has not necessarily completed. See *write()* for the effects of O\_NONBLOCK on *write()*.

**11.1.9 Special Characters**

Certain characters have special functions on input or output or both. These functions are summarized as follows:

**INTR** Special character on input, which is recognized if the ISIG flag is set. Generates a SIGINT signal which is sent to all processes in the foreground process group for which the terminal is the controlling terminal. If ISIG is set, the INTR character shall be discarded when processed.

**QUIT** Special character on input, which is recognized if the ISIG flag is set. Generates a SIGQUIT signal which is sent to all processes in the foreground process group for which the terminal is the controlling terminal. If ISIG is set, the QUIT character shall be discarded when processed.

**ERASE** Special character on input, which is recognized if the ICANON flag is set. Erases the last character in the current line; see [Section 11.1.6](#) (on page 202). It shall not erase beyond the start of a line, as delimited by an NL, EOF, or EOL character. If ICANON is set, the ERASE character shall be discarded when processed.

**KILL** Special character on input, which is recognized if the ICANON flag is set. Deletes the entire line, as delimited by an NL, EOF, or EOL character. If ICANON is set, the KILL character shall be discarded when processed.

**EOF** Special character on input, which is recognized if the ICANON flag is set. When received, all the bytes waiting to be read are immediately passed to the process without waiting for a <newline>, and the EOF is discarded. Thus, if there are no bytes waiting (that is, the EOF occurred at the beginning of a line), a byte count of zero shall be returned from the *read()*, representing an end-of-file indication. If ICANON is set, the EOF character shall be discarded when processed.

**NL** Special character on input, which is recognized if the ICANON flag is set. It is the line delimiter <newline>. It cannot be changed.

**EOL** Special character on input, which is recognized if the ICANON flag is set. It is an additional line delimiter, like NL.

**SUSP** If the ISIG flag is set, receipt of the SUSP character shall cause a SIGTSTP signal to be sent to all processes in the foreground process group for which the terminal is the controlling terminal, and the SUSP character shall be discarded when processed.

**STOP** Special character on both input and output, which is recognized if the IXON (output control) or IXOFF (input control) flag is set. Can be used to suspend output temporarily. It is useful with CRT terminals to prevent output from disappearing before it can be read. If IXON is set, the STOP character shall be discarded when processed.

**START** Special character on both input and output, which is recognized if the IXON (output control) or IXOFF (input control) flag is set. Can be used to resume output that has been suspended by a STOP character. If IXON is set, the START character shall be discarded when processed.

**CR** Special character on input, which is recognized if the ICANON flag is set; it is the <carriage-return> character. When ICANON and ICRNL are set and IGNCR is not set, this character shall be translated into an NL, and shall have the same effect as an NL character. It cannot be changed.

The NL and CR characters cannot be changed. It is implementation-defined whether the START and STOP characters can be changed. The values for INTR, QUIT, ERASE, KILL, EOF, EOL, and SUSP shall be changeable to suit individual tastes. Special character functions associated with



6877 changeable special control characters can be disabled individually.

6878 If two or more special characters have the same value, the function performed when that

6879 character is received is undefined.

6880 A special character is recognized not only by its value, but also by its context; for example, an

6881 implementation may support multi-byte sequences that have a meaning different from the

6882 meaning of the bytes when considered individually. Implementations may also support

6883 additional single-byte functions. These implementation-defined multi-byte or single-byte

6884 functions shall be recognized only if the IEXTEN flag is set; otherwise, data is received without

6885 interpretation, except as required to recognize the special characters defined in this section.

6886 XSI If IEXTEN is set, the ERASE, KILL, and EOF characters can be escaped by a preceding

6887 <backslash> character, in which case no special function shall occur.

#### 6888 11.1.10 Modem Disconnect

6889 If a modem disconnect is detected by the terminal interface for a controlling terminal, and if

6890 CLOCAL is not set in the *c\_cflag* field for the terminal (see [Section 11.2.4](#), on page 209), the

6891 SIGHUP signal shall be sent to the controlling process for which the terminal is the controlling

6892 terminal. Unless other arrangements have been made, this shall cause the controlling process to

6893 terminate (see *exit()*). Any subsequent read from the terminal device shall return the value of

6894 zero, indicating end-of-file; see *read()*. Thus, processes that read a terminal file and test for end-

6895 of-file can terminate appropriately after a disconnect. If the EIO condition as specified in *read()*

6896 also exists, it is unspecified whether on EOF condition or [EIO] is returned. Any subsequent

6897 *write()* to the terminal device shall return -1, with *errno* set to [EIO], until the device is closed.

#### 6898 11.1.11 Closing a Terminal Device File

6899 The last process to close a terminal device file shall cause any output to be sent to the device and

6900 shall cause any input to be discarded. If HUPCL is set in the control modes and the

6901 communications port supports a disconnect function, the terminal device shall perform a

6902 disconnect.

### 6903 11.2 Parameters that Can be Set

#### 6904 11.2.1 The **termios** Structure

6905 Routines that need to control certain terminal I/O characteristics shall do so by using the

6906 **termios** structure as defined in the [<termios.h>](#) header.

6907 Since the **termios** structure may include additional members, and the standard members may

6908 include both standard and non-standard modes, the structure should never be initialized

6909 directly by the application as this may cause the terminal to behave in a non-conforming

6910 manner. When opening a terminal device (other than a pseudo-terminal) that is not already open

6911 in any process, it should be opened with the O\_TTY\_INIT flag before initializing the structure

6912 using *tcgetattr()* to ensure that any non-standard elements of the **termios** structure are set to

6913 values that result in conforming behavior of the terminal interface.

The members of the **termios** structure include (but are not limited to):

Member Type	Array Size	Member Name	Description
<b>tcflag_t</b>	NCCS	<i>c_iflag</i>	Input modes.
<b>tcflag_t</b>		<i>c_oflag</i>	Output modes.
<b>tcflag_t</b>		<i>c_cflag</i>	Control modes.
<b>tcflag_t</b>		<i>c_lflag</i>	Local modes.
<b>cc_t</b>		<i>c_cc[ ]</i>	Control characters.

The **tcflag\_t** and **cc\_t** types are defined in the **<termios.h>** header. They shall be unsigned integer types.

## 11.2.2 Input Modes

Values of the *c\_iflag* field describe the basic terminal input control, and are composed of the bitwise-inclusive OR of the masks shown, which shall be bitwise-distinct. The mask name symbols in this table are defined in **<termios.h>**:

Mask Name	Description
BRKINT	Signal interrupt on break.
ICRNL	Map CR to NL on input.
IGNBRK	Ignore break condition.
IGNCR	Ignore CR.
IGNPAR	Ignore characters with parity errors.
INLCR	Map NL to CR on input.
INPCK	Enable input parity check.
ISTRIP	Strip character.
IXANY	Enable any character to restart output.
IXOFF	Enable start/stop input control.
IXON	Enable start/stop output control.
PARMRK	Mark parity errors.

In the context of asynchronous serial data transmission, a break condition shall be defined as a sequence of zero-valued bits that continues for more than the time to send one byte. The entire sequence of zero-valued bits is interpreted as a single break condition, even if it continues for a time equivalent to more than one byte. In contexts other than asynchronous serial data transmission, the definition of a break condition is implementation-defined.

If IGNBRK is set, a break condition detected on input shall be ignored; that is, not put on the input queue and therefore not read by any process. If IGNBRK is not set and BRKINT is set, the break condition shall flush the input and output queues, and if the terminal is the controlling terminal of a foreground process group, the break condition shall generate a single SIGINT signal to that foreground process group. If neither IGNBRK nor BRKINT is set, a break condition shall be read as a single 0x00, or if PARMRK is set, as 0xff 0x00 0x00.

If IGNPAR is set, a byte with a framing or parity error (other than break) shall be ignored.

If PARMRK is set, and IGNPAR is not set, a byte with a framing or parity error (other than break) shall be given to the application as the three-byte sequence 0xff 0x00 X, where 0xff 0x00 is a two-byte flag preceding each sequence and X is the data of the byte received in error. To avoid ambiguity in this case, if ISTRIP is not set, a valid byte of 0xff is given to the application as 0xff 0xff. If neither PARMRK nor IGNPAR is set, a framing or parity error (other than break) shall be given to the application as a single byte 0x00.

If INPCK is set, input parity checking shall be enabled. If INPCK is not set, input parity checking shall be disabled, allowing output parity generation without input parity errors. Note that whether input parity checking is enabled or disabled is independent of whether parity detection is enabled or disabled (see [Section 11.2.4](#), on page 209). If parity detection is enabled but input parity checking is disabled, the hardware to which the terminal is connected shall recognize the parity bit, but the terminal special file shall not check whether or not this bit is correctly set.

If ISTRIP is set, valid input bytes shall first be stripped to seven bits; otherwise, all eight bits shall be processed.

If INLCR is set, a received NL character shall be translated into a CR character. If IGNCR is set, a received CR character shall be ignored (not read). If IGNCR is not set and ICRNL is set, a received CR character shall be translated into an NL character.

If IXANY is set, any input character shall restart output that has been suspended.

If IXON is set, start/stop output control shall be enabled. A received STOP character shall suspend output and a received START character shall restart output. When IXON is set, START and STOP characters are not read, but merely perform flow control functions. When IXON is not set, the START and STOP characters shall be read.

If IXOFF is set, start/stop input control shall be enabled. The system shall transmit STOP characters, which are intended to cause the terminal device to stop transmitting data, as needed to prevent the input queue from overflowing and causing implementation-defined behavior, and shall transmit START characters, which are intended to cause the terminal device to resume transmitting data, as soon as the device can continue transmitting data without risk of overflowing the input queue. The precise conditions under which STOP and START characters are transmitted are implementation-defined.

The initial input control value after *open()* is implementation-defined.

### 11.2.3 Output Modes

The *c\_oflag* field specifies the terminal interface's treatment of output, and is composed of the bitwise-inclusive OR of the masks shown, which shall be bitwise-distinct. The mask name symbols in the following table are defined in [<termios.h>](#):

Mask Name	Description
OPOST	Perform output processing.
ONLCR	Map NL to CR-NL on output.
OCRNL	Map CR to NL on output.
ONOCR	No CR output at column 0.
ONLRET	NL performs CR function.
OFILL	Use fill characters for delay.
OFDEL	Fill is DEL, else NUL.
NLDLY	Select newline delays:
NL0	Newline character type 0.
NL1	Newline character type 1.
CRDLY	Select carriage-return delays:
CR0	Carriage-return delay type 0.
CR1	Carriage-return delay type 1.
CR2	Carriage-return delay type 2.
CR3	Carriage-return delay type 3.
TABDLY	Select horizontal-tab delays:
TAB0	Horizontal-tab delay type 0.
TAB1	Horizontal-tab delay type 1.
TAB2	Horizontal-tab delay type 2.
TAB3	Expand tabs to spaces.
BSDLY	Select backspace delays:
BS0	Backspace-delay type 0.
BS1	Backspace-delay type 1.
VTDLY	Select vertical-tab delays:
VT0	Vertical-tab delay type 0.
VT1	Vertical-tab delay type 1.
FFDLY	Select form-feed delays:
FF0	Form-feed delay type 0.
FF1	Form-feed delay type 1.

If OPOST is set, output data shall be post-processed as described below, so that lines of text are modified to appear appropriately on the terminal device; otherwise, characters shall be transmitted without change.

If ONLCR is set, the NL character shall be transmitted as the CR-NL character pair. If OCRNL is set, the CR character shall be transmitted as the NL character. If ONOCR is set, no CR character shall be transmitted when at column 0 (first position). If ONLRET is set, the NL character is assumed to do the carriage-return function; the column pointer shall be set to 0 and the delays specified for CR shall be used. Otherwise, the NL character is assumed to do just the line-feed function; the column pointer remains unchanged. The column pointer shall also be set to 0 if the CR character is actually transmitted.

The delay bits specify how long transmission stops to allow for mechanical or other movement when certain characters are sent to the terminal. In all cases a value of 0 shall indicate no delay. If OFILL is set, fill characters shall be transmitted for delay instead of a timed delay. This is useful for high baud rate terminals which need only a minimal delay. If OFDEL is set, the fill character shall be DEL; otherwise, NUL.

If a <form-feed> or <vertical-tab> delay is specified, it shall last for about 2 seconds.

Newline delay shall last about 0.10 seconds. If ONLRET is set, the carriage-return delays shall be used instead of the newline delays. If OFILL is set, two fill characters shall be transmitted.

Carriage-return delay type 1 shall be dependent on the current column position, type 2 shall be

about 0.10 seconds, and type 3 shall be about 0.15 seconds. If OFILL is set, delay type 1 shall transmit two fill characters, and type 2 four fill characters.

Horizontal-tab delay type 1 shall be dependent on the current column position. Type 2 shall be about 0.10 seconds. Type 3 specifies that <tab> characters shall be expanded into <space> characters. If OFILL is set, two fill characters shall be transmitted for any delay.

Backspace delay shall last about 0.05 seconds. If OFILL is set, one fill character shall be transmitted.

The actual delays depend on line speed and system load.

The initial output control value after *open()* is implementation-defined.

#### 11.2.4 Control Modes

The *c\_cflag* field describes the hardware control of the terminal, and is composed of the bitwise-inclusive OR of the masks shown, which shall be bitwise-distinct. The mask name symbols in this table are defined in [<termios.h>](#); not all values specified are required to be supported by the underlying hardware (if any). If the terminal is a pseudo-terminal, it is unspecified whether non-default values are unsupported, or are supported and emulated in software, or are handled by *tcsetattr()*, *tcgetattr()*, and the *stty* utility as if they are supported but have no effect on the behavior of the terminal interface.

Mask Name	Description
CLOCAL	Ignore modem status lines.
CREAD	Enable receiver.
CSIZE	Number of bits transmitted or received per byte:
CS5	5 bits
CS6	6 bits
CS7	7 bits
CS8	8 bits.
CSTOPB	Send two stop bits, else one.
HUPCL	Hang up on last close.
PARENB	Parity enable.
PARODD	Odd parity, else even.

In addition, the input and output baud rates are stored in the **termios** structure. The symbols in the following table are defined in [<termios.h>](#). Not all values specified are required to be supported by the underlying hardware (if any). For pseudo-terminals, the input and output baud rates set in the **termios** structure need not affect the speed of data transmission through the terminal interface.

**Note:** The term “baud” is used historically here, but is not technically correct. This is properly “bits per second”, which may not be the same as baud. However, the term is used because of the historical usage and understanding.

Name	Description	Name	Description
B0	Hang up	B600	600 baud
B50	50 baud	B1200	1200 baud
B75	75 baud	B1800	1800 baud
B110	110 baud	B2400	2400 baud
B134	134.5 baud	B4800	4800 baud
B150	150 baud	B9600	9600 baud
B200	200 baud	B19200	19200 baud
B300	300 baud	B38400	38400 baud

The following functions are provided for getting and setting the values of the input and output baud rates in the **termios** structure: `cfgetispeed()`, `cfgetospeed()`, `cfsetispeed()`, and `cfsetospeed()`. The effects on the terminal device shall not become effective and not all errors need be detected until the `tcsetattr()` function is successfully called.

The CSIZE bits shall specify the number of transmitted or received bits per byte. If ISTRIP is not set, the value of all the other bits is unspecified. If ISTRIP is set, the value of all but the 7 low-order bits shall be zero, but the value of any other bits beyond CSIZE is unspecified when read. CSIZE shall not include the parity bit, if any. If CSTOPB is set, two stop bits shall be used; otherwise, one stop bit. For example, at 110 baud, two stop bits are normally used.

If CREAD is set, the receiver shall be enabled; otherwise, no characters shall be received.

If PARENB is set, parity generation and detection shall be enabled and a parity bit is added to each byte. If parity is enabled, PARODD shall specify odd parity if set; otherwise, even parity shall be used.

If HUPCL is set, the modem control lines for the port shall be lowered when the last process with the port open closes the port or the process terminates. The modem connection shall be broken.

If CLOCAL is set, a connection shall not depend on the state of the modem status lines. If CLOCAL is clear, the modem status lines shall be monitored.

Under normal circumstances, a call to the `open()` function shall wait for the modem connection to complete. However, if the `O_NONBLOCK` flag is set (see `open()`) or if CLOCAL has been set, the `open()` function shall return immediately without waiting for the connection.

If the object for which the control modes are set is not an asynchronous serial connection, some of the modes may be ignored; for example, if an attempt is made to set the baud rate on a network connection to a terminal on another host, the baud rate need not be set on the connection between that terminal and the machine to which it is directly connected.

The initial hardware control value after `open()` is implementation-defined.

### 11.2.5 Local Modes

The `c_lflag` field of the argument structure is used to control various functions. It is composed of the bitwise-inclusive OR of the masks shown, which shall be bitwise-distinct. The mask name symbols in this table are defined in [<termios.h>](#).



Mask Name	Description
ECHO	Enable echo.
ECHOE	Echo ERASE as an error correcting backspace.
ECHOK	Echo KILL.
ECHONL	Echo <newline>.
ICANON	Canonical input (erase and kill processing).
IEXTEN	Enable extended (implementation-defined) functions.
ISIG	Enable signals.
NOFLSH	Disable flush after interrupt, quit, or suspend.
TOSTOP	Send SIGTTOU for background output.

If ECHO is set, input characters shall be echoed back to the terminal. If ECHO is clear, input characters shall not be echoed.

If ECHOE and ICANON are set, the ERASE character shall cause the terminal to erase, if possible, the last character in the current line from the display. If there is no character to erase, an implementation may echo an indication that this was the case, or do nothing.

If ECHOK and ICANON are set, the KILL character shall either cause the terminal to erase the line from the display or shall echo the <newline> character after the KILL character.

If ECHONL and ICANON are set, the <newline> character shall be echoed even if ECHO is not set.

If ICANON is set, canonical processing shall be enabled. This enables the erase and kill edit functions, and the assembly of input characters into lines delimited by NL, EOF, and EOL, as described in [Section 11.1.6](#) (on page 202).

If ICANON is not set, read requests shall be satisfied directly from the input queue. A read shall not be satisfied until at least MIN bytes have been received or the timeout value TIME expired between bytes. The time value represents tenths of a second. See [Section 11.1.7](#) (on page 202) for more details.

If IEXTEN is set, implementation-defined functions shall be recognized from the input data. It is implementation-defined how IEXTEN being set interacts with ICANON, ISIG, IXON, or IXOFF. If IEXTEN is not set, implementation-defined functions shall not be recognized and the corresponding input characters are processed as described for ICANON, ISIG, IXON, and IXOFF.

If ISIG is set, each input character shall be checked against the special control characters INTR, QUIT, and SUSP. If an input character matches one of these control characters, the function associated with that character shall be performed. If ISIG is not set, no checking shall be done. Thus these special input functions are possible only if ISIG is set.

If NOFLSH is set, the normal flush of the input and output queues associated with the INTR, QUIT, and SUSP characters shall not be done.

If TOSTOP is set, the signal SIGTTOU shall be sent to the process group of a process that tries to write to its controlling terminal if it is not in the foreground process group for that terminal. This signal, by default, stops the members of the process group. Otherwise, the output generated by that process shall be output to the current output stream. If the writing process is ignoring the SIGTTOU signal or the writing thread is blocking the SIGTTOU signal, the process is allowed to produce output, and the SIGTTOU signal shall not be sent.

The initial local control value after *open()* is implementation-defined.

## 11.2.6 Special Control Characters

The special control character values shall be defined by the array `c_cc`. The subscript name and description for each element in both canonical and non-canonical modes are as follows:

Subscript Usage		Description
Canonical Mode	Non-Canonical Mode	
VEOF	VINTR	EOF character
VEOL		EOL character
VERASE		ERASE character
VINTR		INTR character
VKILL		KILL character
	VMIN	MIN value
VQUIT	VQUIT	QUIT character
VSUSP	VSUSP	SUSP character
	VTIME	TIME value
VSTART	VSTART	START character
VSTOP	VSTOP	STOP character

The subscript values are unique, except that the VMIN and VTIME subscripts may have the same values as the VEOF and VEOL subscripts, respectively.

Implementations that do not support changing the START and STOP characters may ignore the character values in the `c_cc` array indexed by the VSTART and VSTOP subscripts when `tcsetattr()` is called, but shall return the value in use when `tcgetattr()` is called.

The initial values of all control characters are implementation-defined.

If the value of one of the changeable special control characters (see [Section 11.1.9](#), on page 204) is `_POSIX_VDISABLE`, that function shall be disabled; that is, no input data is recognized as the disabled special character. If ICANON is not set, the value of `_POSIX_VDISABLE` has no special meaning for the VMIN and VTIME entries of the `c_cc` array.



7183

Chapter 12

7184

## Utility Conventions

7185

### 12.1 Utility Argument Syntax

7186

7187

This section describes the argument syntax of the standard utilities and introduces terminology used throughout POSIX.1-2008 for describing the arguments processed by the utilities.

7188

7189

7190

Within POSIX.1-2008, a special notation is used for describing the syntax of a utility's arguments. Unless otherwise noted, all utility descriptions use this notation, which is illustrated by this example (see XCU [Section 2.9.1](#), on page 2355):

7191

7192

```
utility_name[-a][-b][-c option_argument]
          [-d|-e][-f[option_argument]][operand...]
```

7193

7194

The notation used for the SYNOPSIS sections imposes requirements on the implementors of the standard utilities and provides a simple reference for the application developer or system user.

7195

7196

7197

7198

7199

7200

1. The utility in the example is named *utility\_name*. It is followed by options, option-arguments, and operands. The arguments that consist of <hyphen-minus> characters and single letters or digits, such as 'a', are known as "options" (or, historically, "flags"). Certain options are followed by an "option-argument", as shown with [-c *option\_argument*]. The arguments following the last options and option-arguments are named "operands".

7201

7202

7203

7204

7205

7206

7207

7208

2. Option-arguments are shown separated from their options by <blank> characters, except when the option-argument is enclosed in the '[' and ']' notation to indicate that it is optional. This reflects the situation in which an optional option-argument (if present) is included within the same argument string as the option; for a mandatory option-argument, it is the next argument. The Utility Syntax Guidelines in [Section 12.2](#) (on page 216) require that the option be a separate argument from its option-argument and that option-arguments not be optional, but there are some exceptions in POSIX.1-2008 to ensure continued operation of historical applications:

7209

7210

7211

7212

7213

7214

- a. If the SYNOPSIS of a standard utility shows an option with a mandatory option-argument (as with [-c *option\_argument*] in the example), a conforming application shall use separate arguments for that option and its option-argument. However, a conforming implementation shall also permit applications to specify the option and option-argument in the same argument string without intervening <blank> characters.

7215

7216

7217

7218

7219

7220

7221

- b. If the SYNOPSIS shows an optional option-argument (as with [-f[*option\_argument*]] in the example), a conforming application shall place any option-argument for that option directly adjacent to the option in the same argument string, without intervening <blank> characters. If the utility receives an argument containing only the option, it shall behave as specified in its description for an omitted option-argument; it shall not treat the next argument (if any) as the option-argument for that option.

3. Options are usually listed in alphabetical order unless this would make the utility description more confusing. There are no implied relationships between the options based upon the order in which they appear, unless otherwise stated in the OPTIONS section, or unless the exception in Guideline 11 of [Section 12.2](#) (on page 216) applies. If an option that does not have option-arguments is repeated, the results are undefined, unless otherwise stated.

4. Frequently, names of parameters that require substitution by actual values are shown with embedded <underscore> characters. Alternatively, parameters are shown as follows:

`<parameter name>`

The angle brackets are used for the symbolic grouping of a phrase representing a single parameter and conforming applications shall not include them in data submitted to the utility.

5. When a utility has only a few permissible options, they are sometimes shown individually, as in the example. Utilities with many flags generally show all of the individual flags (that do not take option-arguments) grouped, as in:

`utility_name [-abcDxyz][-p arg][operand]`

Utilities with very complex arguments may be shown as follows:

`utility_name [options][operands]`

6. Unless otherwise specified, whenever an operand or option-argument is, or contains, a numeric value:

- The number is interpreted as a decimal integer.
- Numerals in the range 0 to 2 147 483 647 are syntactically recognized as numeric values.
- When the utility description states that it accepts negative numbers as operands or option-arguments, numerals in the range -2 147 483 647 to 2 147 483 647 are syntactically recognized as numeric values.
- When the utility description states that the number is a file size-related value (such as a file size or offset, line number, or block count), numerals in the range 0 to the maximum file size supported by the implementation are syntactically recognized as numeric values (see XCU [Section 1.5](#), on page 2333). Where negative values are permitted, any value in the range -(maximum file size) to the maximum file size is accepted.

- Ranges greater than those listed here are allowed.

This does not mean that all numbers within the allowable range are necessarily semantically correct. A standard utility that accepts an option-argument or operand that is to be interpreted as a number, and for which a range of values smaller than that shown above is permitted by the POSIX.1-2008, describes that smaller range along with the description of the option-argument or operand. If an error is generated, the utility's diagnostic message shall indicate that the value is out of the supported range, not that it is syntactically incorrect.

7. Arguments or option-arguments enclosed in the '*[*' and '*]*' notation are optional and can be omitted. Conforming applications shall not include the '*[*' and '*]*' symbols in data submitted to the utility.

8. Arguments separated by the ' | ' (<vertical-line>) bar notation are mutually-exclusive. Conforming applications shall not include the ' | ' symbol in data submitted to the utility. Alternatively, mutually-exclusive options and operands may be listed with multiple synopsis lines.

For example:

```
utility_name -d[-a][-c option_argument][operand...]
utility_name[-a][-b][operand...]
```

When multiple synopsis lines are given for a utility, it is an indication that the utility has mutually-exclusive arguments. These mutually-exclusive arguments alter the functionality of the utility so that only certain other arguments are valid in combination with one of the mutually-exclusive arguments. Only one of the mutually-exclusive arguments is allowed for invocation of the utility. Unless otherwise stated in an accompanying OPTIONS section, the relationships between arguments depicted in the SYNOPSIS sections are mandatory requirements placed on conforming applications. The use of conflicting mutually-exclusive arguments produces undefined results, unless a utility description specifies otherwise. When an option is shown without the '[' and ']' brackets, it means that option is required for that version of the SYNOPSIS. However, it is not required to be the first argument, as shown in the example above, unless otherwise stated.

9. Ellipses ("...") are used to denote that one or more occurrences of an operand are allowed. When an option or an operand followed by ellipses is enclosed in brackets, zero or more options or operands can be specified. The form:

```
utility_name [-g option_argument]...[operand...]
```

indicates that multiple occurrences of the option and its option-argument preceding the ellipses are valid, with semantics as indicated in the OPTIONS section of the utility. (See also Guideline 11 in [Section 12.2](#) (on page 216).)

The form:

```
utility_name -f option_argument [-f option_argument]... [operand...]
```

indicates that the -f option is required to appear at least once and may appear multiple times.

10. When the synopsis line is too long to be printed on a single line in the Shell and Utilities volume of POSIX.1-2008, the indented lines following the initial line are continuation lines. An actual use of the command would appear on a single logical line.

## 12.2 Utility Syntax Guidelines

The following guidelines are established for the naming of utilities and for the specification of options, option-arguments, and operands. The *getopt()* function in the System Interfaces volume of POSIX.1-2008 assists utilities in handling options and operands that conform to these guidelines.

Operands and option-arguments can contain characters not specified in the portable character set.

The guidelines are intended to provide guidance to the authors of future utilities, such as those written specific to a local system or that are components of a larger application. Some of the standard utilities do not conform to all of these guidelines; in those cases, the OPTIONS sections describe the deviations.

**Guideline 1:** Utility names should be between two and nine characters, inclusive.

**Guideline 2:** Utility names should include lowercase letters (the **lower** character classification) and digits only from the portable character set.

**Guideline 3:** Each option name should be a single alphanumeric character (the **alnum** character classification) from the portable character set. The **-W** (capital-W) option shall be reserved for vendor options.

Multi-digit options should not be allowed.

**Guideline 4:** All options should be preceded by the **'-'** delimiter character.

**Guideline 5:** One or more options without option-arguments, followed by at most one option that takes an option-argument, should be accepted when grouped behind one **'-'** delimiter.

**Guideline 6:** Each option and option-argument should be a separate argument, except as noted in [Section 12.1](#) (on page 213), item (2).

**Guideline 7:** Option-arguments should not be optional.

**Guideline 8:** When multiple option-arguments are specified to follow a single option, they should be presented as a single argument, using **<comma>** characters within that argument or **<blank>** characters within that argument to separate them.

**Guideline 9:** All options should precede operands on the command line.

**Guideline 10:** The first **--** argument that is not an option-argument should be accepted as a delimiter indicating the end of options. Any following arguments should be treated as operands, even if they begin with the **'-'** character.

**Guideline 11:** The order of different options relative to one another should not matter, unless the options are documented as mutually-exclusive and such an option is documented to override any incompatible options preceding it. If an option that has option-arguments is repeated, the option and option-argument combinations should be interpreted in the order specified on the command line.

**Guideline 12:** The order of operands may matter and position-related interpretations should be determined on a utility-specific basis.

**Guideline 13:** For utilities that use operands to represent files to be opened for either reading or writing, the **'-'** operand should be used to mean only standard input (or standard output when it is clear from context that an output file is being specified) or a file named **-**.

**Guideline 14:** If an argument can be identified according to Guidelines 3 through 10 as an option, or as a group of options without option-arguments behind one ‘-’ delimiter, then it should be treated as such.

The utilities in the Shell and Utilities volume of POSIX.1-2008 that claim conformance to these guidelines shall conform completely to these guidelines as if these guidelines contained the term “shall” instead of “should”. On some implementations, the utilities accept usage in violation of these guidelines for backwards-compatibility as well as accepting the required form.

Where a utility described in the Shell and Utilities volume of POSIX.1-2008 as conforming to these guidelines is required to accept, or not to accept, the operand ‘-’ to mean standard input or output, this usage is explained in the OPERANDS section. Otherwise, if such a utility uses operands to represent files, it is implementation-defined whether the operand ‘-’ stands for standard input (or standard output), or for a file named -.

It is recommended that all future utilities and applications use these guidelines to enhance user portability. The fact that some historical utilities could not be changed (to avoid breaking existing applications) should not deter this future goal.



This chapter describes the contents of headers.

Headers contain function prototypes, the definition of symbolic constants, common structures, preprocessor macros, and defined types. Each function in the System Interfaces volume of POSIX.1-2008 specifies the headers that an application shall include in order to use that function. In most cases, only one header is required. These headers are present on an application development system; they need not be present on the target execution system.

### **Format of Entries**

The entries in this chapter are based on a common format as follows. The only sections relating to conformance are the SYNOPSIS and DESCRIPTION.

#### **NAME**

This section gives the name or names of the entry and briefly states its purpose.

#### **SYNOPSIS**

This section summarizes the use of the entry being described.

#### **DESCRIPTION**

This section describes the functionality of the header.

#### **APPLICATION USAGE**

This section is informative. This section gives warnings and advice to application developers about the entry. In the event of conflict between warnings and advice and a normative part of this volume of POSIX.1-2008, the normative material is to be taken as correct.

#### **RATIONALE**

This section is informative. This section contains historical information concerning the contents of this volume of POSIX.1-2008 and why features were included or discarded by the standard developers.

#### **FUTURE DIRECTIONS**

This section is informative. This section provides comments which should be used as a guide to current thinking; there is not necessarily a commitment to adopt these future directions.

#### **SEE ALSO**

This section is informative. This section gives references to related information.

#### **CHANGE HISTORY**

This section is informative. This section shows the derivation of the entry and any significant changes that have been made to it.

**NAME**

aio.h — asynchronous input and output

**SYNOPSIS**

```
#include <aio.h>
```

**DESCRIPTION**

The <aio.h> header shall define the **aio\_cb** structure, which shall include at least the following members:

int	aio_fildes	File descriptor.
off_t	aio_offset	File offset.
volatile void *	aio_buf	Location of buffer.
size_t	aio_nbytes	Length of transfer.
int	aio_reqprio	Request priority offset.
struct sigevent	aio_sigevent	Signal number and value.
int	aio_lio_opcode	Operation to be performed.

The <aio.h> header shall define the **off\_t**, **pthread\_attr\_t**, **size\_t**, and **ssize\_t** types as described in <sys/types.h>.

The <aio.h> header shall define the **struct timespec** structure as described in <time.h>.

The <aio.h> header shall define the **sigevent** structure and **sigval** union as described in <signal.h>.

The <aio.h> header shall define the following symbolic constants:

**AIO\_ALLDONE** A return value indicating that none of the requested operations could be canceled since they are already complete.

**AIO\_CANCELED** A return value indicating that all requested operations have been canceled.

**AIO\_NOTCANCELED** A return value indicating that some of the requested operations could not be canceled since they are in progress.

**LIO\_NOP** A *lio\_listio()* element operation option indicating that no transfer is requested.

**LIO\_NOWAIT** A *lio\_listio()* synchronization operation indicating that the calling thread is to continue execution while the *lio\_listio()* operation is being performed, and no notification is given when the operation is complete.

**LIO\_READ** A *lio\_listio()* element operation option requesting a read.

**LIO\_WAIT** A *lio\_listio()* synchronization operation indicating that the calling thread is to suspend until the *lio\_listio()* operation is complete.

**LIO\_WRITE** A *lio\_listio()* element operation option requesting a write.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
int      aio_cancel(int, struct aio_cb *);
int      aio_error(const struct aio_cb *);
FSC|SIO int      aio_fsync(int, struct aio_cb *);
int      aio_read(struct aio_cb *);
ssize_t  aio_return(struct aio_cb *);
int      aio_suspend(const struct aio_cb *const [], int,
```



```

7436             const struct timespec *);
7437 int         aio_write(struct aiocb *);
7438 int         lio_listio(int, struct aiocb *restrict const [restrict], int,
7439             struct sigevent *restrict);

```

7440 Inclusion of the <aio.h> header may make visible symbols defined in the headers <fcntl.h>, <signal.h>, and <time.h>.

7441

7442 **APPLICATION USAGE**

7443 None.

7444 **RATIONALE**

7445 None.

7446 **FUTURE DIRECTIONS**

7447 None.

7448 **SEE ALSO**

7449 <fcntl.h>, <signal.h>, <sys/types.h>, <time.h>

7450 XSH *aio\_cancel()*, *aio\_error()*, *aio\_fsync()*, *aio\_read()*, *aio\_return()*, *aio\_suspend()*, *aio\_write()*, *fsync()*, *lio\_listio()*, *lseek()*, *read()*, *write()*

7451

7452 **CHANGE HISTORY**

7453 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

7454 **Issue 6**

7455 The <aio.h> header is marked as part of the Asynchronous Input and Output option.

7456 The description of the constants is expanded.

7457 The **restrict** keyword is added to the prototype for *lio\_listio()*.

7458 **Issue 7**

7459 The <aio.h> header is moved from the Asynchronous Input and Output option to the Base.

7460 This reference page is clarified with respect to macros and symbolic constants, and type and structure declarations are added.

7461

7462 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0038 [98] is applied.

**NAME**

arpa/inet.h — definitions for internet operations

**SYNOPSIS**

```
#include <arpa/inet.h>
```

**DESCRIPTION**

The <arpa/inet.h> header shall define the **in\_port\_t** and **in\_addr\_t** types as described in <netinet/in.h>.

The <arpa/inet.h> header shall define the **in\_addr** structure as described in <netinet/in.h>.

IP6 The <arpa/inet.h> header shall define the **INET\_ADDRSTRLEN** and **INET6\_ADDRSTRLEN** macros as described in <netinet/in.h>.

The following shall be declared as functions, or defined as macros, or both. If functions are declared, function prototypes shall be provided.

```
uint32_t htonl(uint32_t);
uint16_t htons(uint16_t);
uint32_t ntohl(uint32_t);
uint16_t ntohs(uint16_t);
```

The <arpa/inet.h> header shall define the **uint32\_t** and **uint16\_t** types as described in <inttypes.h>.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
in_addr_t    inet_addr(const char *);
char         *inet_ntoa(struct in_addr);
const char   *inet_ntop(int, const void *restrict, char *restrict,
                        socklen_t);
int          inet_pton(int, const char *restrict, void *restrict);
```

Inclusion of the <arpa/inet.h> header may also make visible all symbols from <netinet/in.h> and <inttypes.h>.

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

<inttypes.h>, <netinet/in.h>

XSH *htonl()*, *inet\_addr()*, *inet\_ntop()*

**CHANGE HISTORY**

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

The **restrict** keyword is added to the prototypes for *inet\_ntop()* and *inet\_pton()*.

**Issue 7**

SD5-XBD-ERN-6 is applied.

**NAME**

assert.h — verify program assertion

**SYNOPSIS**

```
#include <assert.h>
```

**DESCRIPTION**

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The <assert.h> header shall define the *assert()* macro. It refers to the macro NDEBUB which is not defined in the header. If NDEBUB is defined as a macro name before the inclusion of this header, the *assert()* macro shall be defined simply as:

```
#define assert(ignore)((void) 0)
```

Otherwise, the macro behaves as described in *assert()*.

The *assert()* macro shall be redefined according to the current state of NDEBUB each time <assert.h> is included.

The *assert()* macro shall be implemented as a macro, not as a function. If the macro definition is suppressed in order to access an actual function, the behavior is undefined.

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

XSH *assert()*

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 6**

The definition of the *assert()* macro is changed for alignment with the ISO/IEC 9899:1999 standard.

## NAME

complex.h — complex arithmetic

## SYNOPSIS

```
#include <complex.h>
```

## DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The <complex.h> header shall define the following macros:

**complex**           Expands to **\_Complex**.

**\_Complex\_I**       Expands to a constant expression of type **const float \_Complex**, with the value of the imaginary unit (that is, a number  $i$  such that  $i^2=-1$ ).

**imaginary**       Expands to **\_Imaginary**.

**\_Imaginary\_I**   Expands to a constant expression of type **const float \_Imaginary** with the value of the imaginary unit.

**I**               Expands to either **\_Imaginary\_I** or **\_Complex\_I**. If **\_Imaginary\_I** is not defined, **I** expands to **\_Complex\_I**.

The macros **imaginary** and **\_Imaginary\_I** shall be defined if and only if the implementation supports imaginary types.

An application may undefine and then, perhaps, redefine the **complex**, **imaginary**, and **I** macros.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
double           cabs(double complex);
float            cabsf(float complex);
long double      cabsl(long double complex);
double complex   cacos(double complex);
float complex    cacosf(float complex);
double complex   cacosh(double complex);
float complex    cacoshf(float complex);
long double complex cacoshl(long double complex);
long double complex cacosl(long double complex);
double           carg(double complex);
float            cargf(float complex);
long double      cargl(long double complex);
double complex   casin(double complex);
float complex    casinf(float complex);
double complex   casinh(double complex);
float complex    casinhf(float complex);
long double complex casinhl(long double complex);
long double complex casinl(long double complex);
double complex   catan(double complex);
float complex    catanf(float complex);
double complex   catanh(double complex);
float complex    catanhf(float complex);
long double complex catanhl(long double complex);
long double complex catanl(long double complex);
```

```

7580     double complex      ccos(double complex);
7581     float complex       ccosf(float complex);
7582     double complex      ccosh(double complex);
7583     float complex       ccoshf(float complex);
7584     long double complex ccoshl(long double complex);
7585     long double complex ccosl(long double complex);
7586     double complex      cexp(double complex);
7587     float complex       cexpf(float complex);
7588     long double complex cexpl(long double complex);
7589     double              cimag(double complex);
7590     float               cimagf(float complex);
7591     long double         cimagl(long double complex);
7592     double complex      clog(double complex);
7593     float complex       clogf(float complex);
7594     long double complex clogl(long double complex);
7595     double complex      conj(double complex);
7596     float complex       conjf(float complex);
7597     long double complex conjl(long double complex);
7598     double complex      cpow(double complex, double complex);
7599     float complex       cpowf(float complex, float complex);
7600     long double complex cpowl(long double complex, long double complex);
7601     double complex      cproj(double complex);
7602     float complex       cprojf(float complex);
7603     long double complex cprojl(long double complex);
7604     double              creal(double complex);
7605     float               crealf(float complex);
7606     long double         creall(long double complex);
7607     double complex      csin(double complex);
7608     float complex       csinf(float complex);
7609     double complex      csinh(double complex);
7610     float complex       csinhf(float complex);
7611     long double complex csinhl(long double complex);
7612     long double complex csinl(long double complex);
7613     double complex      csqrt(double complex);
7614     float complex       csqrtf(float complex);
7615     long double complex csqrtl(long double complex);
7616     double complex      ctan(double complex);
7617     float complex       ctanf(float complex);
7618     double complex      ctanh(double complex);
7619     float complex       ctanhf(float complex);
7620     long double complex ctanhl(long double complex);
7621     long double complex ctanl(long double complex);

```

**APPLICATION USAGE**

Values are interpreted as radians, not degrees.

**RATIONALE**

The choice of *I* instead of *i* for the imaginary unit concedes to the widespread use of the identifier *i* for other purposes. The application can use a different identifier, say *j*, for the imaginary unit by following the inclusion of the <complex.h> header with:

```
#undef I
#define j _Imaginary_I
```

An *I* suffix to designate imaginary constants is not required, as multiplication by *I* provides a sufficiently convenient and more generally useful notation for imaginary terms. The corresponding real type for the imaginary unit is **float**, so that use of *I* for algorithmic or notational convenience will not result in widening types.

On systems with imaginary types, the application has the ability to control whether use of the macro *I* introduces an imaginary type, by explicitly defining *I* to be `_Imaginary_I` or `_Complex_I`. Disallowing imaginary types is useful for some applications intended to run on implementations without support for such types.

The macro `_Imaginary_I` provides a test for whether imaginary types are supported.

The `cis()` function ( $\cos(x) + I\sin(x)$ ) was considered but rejected because its implementation is easy and straightforward, even though some implementations could compute sine and cosine more efficiently in tandem.

**FUTURE DIRECTIONS**

The following function names and the same names suffixed with *f* or *l* are reserved for future use, and may be added to the declarations in the <complex.h> header.

```
cerf()      cexpm1()   clog2()
cerfc()     clog10()  clgamma()
cexp2()     clog1p()  ctgamma()
```

**SEE ALSO**

XSH `cabs()`, `cacos()`, `cacosh()`, `carg()`, `casin()`, `casinh()`, `catan()`, `catanh()`, `ccos()`, `ccosh()`, `cexp()`, `cimag()`, `clog()`, `conj()`, `cpow()`, `cproj()`, `creal()`, `csin()`, `csinh()`, `csqrt()`, `ctan()`, `ctanh()`

**CHANGE HISTORY**

First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

## NAME

cpio.h — cpio archive values

## SYNOPSIS

```
#include <cpio.h>
```

## DESCRIPTION

The <cpio.h> header shall define the symbolic constants needed by the *c\_mode* field of the *cpio* archive format, with the names and values given in the following table:

Name	Description	Value (Octal)
C_IRUSR	Read by owner.	0000400
C_IWUSR	Write by owner.	0000200
C_IXUSR	Execute by owner.	0000100
C_IRGRP	Read by group.	0000040
C_IWGRP	Write by group.	0000020
C_IXGRP	Execute by group.	0000010
C_IROTH	Read by others.	0000004
C_IWOTH	Write by others.	0000002
C_IXOTH	Execute by others.	0000001
C_ISUID	Set user ID.	0004000
C_ISGID	Set group ID.	0002000
C_ISVTX	On directories, restricted deletion flag.	0001000
C_ISDIR	Directory.	0040000
C_ISFIFO	FIFO.	0010000
C_ISREG	Regular file.	0100000
C_ISBLK	Block special.	0060000
C_ISCHR	Character special.	0020000
C_ISCTG	Reserved.	0110000
C_ISLNK	Symbolic link.	0120000
C_ISSOCK	Socket.	0140000

The <cpio.h> header shall define the following symbolic constant as a string:

```
MAGIC      "070707"
```

## APPLICATION USAGE

None.

## RATIONALE

None.

## FUTURE DIRECTIONS

None.

## SEE ALSO

XCU *pax*

## CHANGE HISTORY

First released in the Headers Interface, Issue 3 specification. Derived from the POSIX.1-1988 standard.

## Issue 6

The SEE ALSO is updated to refer to *pax*.

**Issue 7**

The **<cpio.h>** header is moved from the XSI option to the Base.

This reference page is clarified with respect to macros and symbolic constants.



7699 **NAME**

7700       ctype.h — character types

7701 **SYNOPSIS**

7702       #include &lt;ctype.h&gt;

7703 **DESCRIPTION**

7704 CX       Some of the functionality described on this reference page extends the ISO C standard.  
 7705       Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to  
 7706       enable the visibility of these symbols in this header.

7707       The <ctype.h> header shall define the **locale\_t** type as described in <locale.h>, representing a  
 7708       locale object.

7709       The following shall be declared as functions and may also be defined as macros. Function  
 7710       prototypes shall be provided for use with ISO C standard compilers.

```

7711       int    isalnum(int);
7712 CX       int   isalnum_l(int, locale_t);
7713       int    isalpha(int);
7714 CX       int   isalpha_l(int, locale_t);
7715 OB XSI   int   isascii(int);
7716       int    isblank(int);
7717 CX       int   isblank_l(int, locale_t);
7718       int    iscntrl(int);
7719 CX       int   iscntrl_l(int, locale_t);
7720       int    isdigit(int);
7721 CX       int   isdigit_l(int, locale_t);
7722       int    isgraph(int);
7723 CX       int   isgraph_l(int, locale_t);
7724       int    islower(int);
7725 CX       int   islower_l(int, locale_t);
7726       int    isprint(int);
7727 CX       int   isprint_l(int, locale_t);
7728       int    ispunct(int);
7729 CX       int   ispunct_l(int, locale_t);
7730       int    isspace(int);
7731 CX       int   isspace_l(int, locale_t);
7732       int    isupper(int);
7733 CX       int   isupper_l(int, locale_t);
7734       int    isxdigit(int);
7735 CX       int   isxdigit_l(int, locale_t);
7736 OB XSI   int   toascii(int);
7737       int    tolower(int);
7738 CX       int   tolower_l(int, locale_t);
7739       int    toupper(int);
7740 CX       int   toupper_l(int, locale_t);

```

7741       The <ctype.h> header shall define the following as macros:

```

7742 OB XSI   int   _toupper(int);
7743       int    _tolower(int);

```

7744 **APPLICATION USAGE**

7745 None.

7746 **RATIONALE**

7747 None.

7748 **FUTURE DIRECTIONS**

7749 None.

7750 **SEE ALSO**7751 **<locale.h>**

7752 XSH Section 2.2 (on page 472), *isalnum()*, *isalpha()*, *isascii()*, *isblank()*, *isctrl()*, *isdigit()*,  
7753 *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*, *isupper()*, *isxdigit()*, *mblen()*, *mbstowcs()*,  
7754 *mbtowc()*, *setlocale()*, *toascii()*, *tolower()*, *\_tolower()*, *toupper()*, *\_toupper()*, *wcstombs()*, *wctomb()*

7755 **CHANGE HISTORY**

7756 First released in Issue 1. Derived from Issue 1 of the SVID.

7757 **Issue 6**

7758 Extensions beyond the ISO C standard are marked.

7759 **Issue 7**

7760 SD5-XBD-ERN-6 is applied, updating the wording regarding the function declarations for  
7761 consistency.

7762 The \*\_l() functions are added from The Open Group Technical Standard, 2006, Extended API Set  
7763 Part 4.

**NAME**

dirent.h — format of directory entries

**SYNOPSIS**

```
#include <dirent.h>
```

**DESCRIPTION**

The internal format of directories is unspecified.

The <dirent.h> header shall define the following type:

**DIR** A type representing a directory stream. The **DIR** type may be an incomplete type.

It shall also define the structure **dirent** which shall include the following members:

XSI     `ino_t    d_ino`     File serial number.

`char    d_name[]`    Filename string of entry.

XSI     The <dirent.h> header shall define the `ino_t` type as described in <sys/types.h>.

The array `d_name` is of unspecified size, but shall contain a filename of at most {NAME\_MAX} bytes followed by a terminating null byte.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
int            alphasort(const struct dirent **, const struct dirent **);
```

```
int            closedir(DIR *);
```

```
int            dirfd(DIR *);
```

```
DIR            *fdopendir(int);
```

```
DIR            *opendir(const char *);
```

```
struct dirent *readdir(DIR *);
```

```
int            readdir_r(DIR *restrict, struct dirent *restrict,
              struct dirent **restrict);
```

```
void           rewinddir(DIR *);
```

```
int            scandir(const char *, struct dirent **,
```

```
              int (*)(const struct dirent *),
```

```
              int (*)(const struct dirent **,
```

```
              const struct dirent **));
```

XSI     `void           seekdir(DIR *, long);`

`long          telldir(DIR *);`

**APPLICATION USAGE**

None.

**RATIONALE**

Information similar to that in the <dirent.h> header is contained in a file <sys/dir.h> in 4.2 BSD and 4.3 BSD. The equivalent in these implementations of **struct dirent** from this volume of POSIX.1-2008 is **struct direct**. The filename was changed because the name <sys/dir.h> was also used in earlier implementations to refer to definitions related to the older access method; this produced name conflicts. The name of the structure was changed because this volume of POSIX.1-2008 does not completely define what is in the structure, so it could be different on some implementations from **struct direct**.

The name of an array of **char** of an unspecified size should not be used as an lvalue. Use of:

```
sizeof(d_name)
```

is incorrect; use:

7808           strlen(d\_name)

7809           instead.

7810           The array of **char** *d\_name* is not a fixed size. Implementations may need to declare **struct dirent**  
7811           with an array size for *d\_name* of 1, but the actual number of bytes provided matches (or only  
7812           slightly exceeds) the length of the filename string.

#### 7813 FUTURE DIRECTIONS

7814           None.

#### 7815 SEE ALSO

7816           <sys/types.h>

7817           XSH *alphasort()*, *closedir()*, *dirfd()*, *fdopendir()*, *readdir()*, *rewinddir()*, *seekdir()*, *telldir()*

#### 7818 CHANGE HISTORY

7819           First released in Issue 2.

#### 7820 Issue 5

7821           The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

#### 7822 Issue 6

7823           The Open Group Corrigendum U026/7 is applied, correcting the prototype for *readdir\_r()*.

7824           The **restrict** keyword is added to the prototype for *readdir\_r()*.

#### 7825 Issue 7

7826           The *alphasort()*, *dirfd()*, and *scandir()* functions are added from The Open Group Technical  
7827           Standard, 2006, Extended API Set Part 1.

7828           The *fdopendir()* function is added from The Open Group Technical Standard, 2006, Extended API  
7829           Set Part 2.

7830           Austin Group Interpretation 1003.1-2001 #110 is applied, clarifying the definition of the **DIR**  
7831           type.

7832           POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0039 [291], XBD/TC1-2008/0040 [291],  
7833           XBD/TC1-2008/0041 [291], and XBD/TC1-2008/0042 [206] are applied.

**NAME**

dlfcn.h — dynamic linking

**SYNOPSIS**

```
#include <dlfcn.h>
```

**DESCRIPTION**

The <dlfcn.h> header shall define at least the following symbolic constants for use in the construction of a *dlopen()* *mode* argument:

RTLD_LAZY	Relocations are performed at an implementation-defined time.
RTLD_NOW	Relocations are performed when the object is loaded.
RTLD_GLOBAL	All symbols are available for relocation processing of other modules.
RTLD_LOCAL	All symbols are not made available for relocation processing by other modules.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
int    dlclose(void *);
char  *dlerror(void);
void  *dlopen(const char *, int);
void  *dlsym(void *restrict, const char *restrict);
```

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

XSH *dlclose()*, *dlerror()*, *dlopen()*, *dlsym()*

**CHANGE HISTORY**

First released in Issue 5.

**Issue 6**

The **restrict** keyword is added to the prototype for *dlsym()*.

**Issue 7**

The <dlfcn.h> header is moved from the XSI option to the Base.

This reference page is clarified with respect to macros and symbolic constants.

7867 **NAME**

7868       errno.h — system error numbers

7869 **SYNOPSIS**

7870       #include &lt;errno.h&gt;

7871 **DESCRIPTION**

7872 CX       Some of the functionality described on this reference page extends the ISO C standard. Any  
 7873       conflict between the requirements described here and the ISO C standard is unintentional. This  
 7874       volume of POSIX.1-2008 defers to the ISO C standard.

7875       The ISO C standard only requires the symbols [EDOM], [EILSEQ], and [ERANGE] to be defined.

7876       The <errno.h> header shall provide a declaration or definition for *errno*. The symbol *errno* shall  
 7877       expand to a modifiable lvalue of type **int**. It is unspecified whether *errno* is a macro or an  
 7878       identifier declared with external linkage. If a macro definition is suppressed in order to access an  
 7879       actual object, or a program defines an identifier with the name *errno*, the behavior is undefined.

7880       The <errno.h> header shall define the following macros which shall expand to integer constant  
 7881       expressions with type **int**, distinct positive values (except as noted below), and which shall be  
 7882       suitable for use in **#if** preprocessing directives:

7883	[E2BIG]	Argument list too long.
7884	[EACCES]	Permission denied.
7885	[EADDRINUSE]	Address in use.
7886	[EADDRNOTAVAIL]	Address not available.
7887	[EAFNOSUPPORT]	Address family not supported.
7888	[EAGAIN]	Resource unavailable, try again (may be the same value as
7889		[EWOULDBLOCK]).
7890	[EALREADY]	Connection already in progress.
7891	[EBADF]	Bad file descriptor.
7892	[EBADMSG]	Bad message.
7893	[EBUSY]	Device or resource busy.
7894	[ECANCELED]	Operation canceled.
7895	[ECHILD]	No child processes.
7896	[ECONNABORTED]	Connection aborted.
7897	[ECONNREFUSED]	Connection refused.
7898	[ECONNRESET]	Connection reset.
7899	[EDEADLK]	Resource deadlock would occur.
7900	[EDESTADDRREQ]	Destination address required.
7901	[EDOM]	Mathematics argument out of domain of function.
7902	[EDQUOT]	Reserved.
7903	[EEXIST]	File exists.

7904		[EFAULT]	Bad address.
7905		[EFBIG]	File too large.
7906		[EHOSTUNREACH]	Host is unreachable.
7907		[EIDRM]	Identifier removed.
7908		[EILSEQ]	Illegal byte sequence.
7909		[EINPROGRESS]	Operation in progress.
7910		[EINTR]	Interrupted function.
7911		[EINVAL]	Invalid argument.
7912		[EIO]	I/O error.
7913		[EISCONN]	Socket is connected.
7914		[EISDIR]	Is a directory.
7915		[ELOOP]	Too many levels of symbolic links.
7916		[EMFILE]	File descriptor value too large.
7917		[EMLINK]	Too many links.
7918		[EMSGSIZE]	Message too large.
7919		[EMULTIHOP]	Reserved.
7920		[ENAMETOOLONG]	Filename too long.
7921		[ENETDOWN]	Network is down.
7922		[ENETRESET]	Connection aborted by network.
7923		[ENETUNREACH]	Network unreachable.
7924		[ENFILE]	Too many files open in system.
7925		[ENOBUFS]	No buffer space available.
7926	OB XSR	[ENODATA]	No message is available on the STREAM head read queue.
7927		[ENODEV]	No such device.
7928		[ENOENT]	No such file or directory.
7929		[ENOEXEC]	Executable file format error.
7930		[ENOLCK]	No locks available.
7931		[ENOLINK]	Reserved.
7932		[ENOMEM]	Not enough space.
7933		[ENOMSG]	No message of the desired type.
7934		[ENOPROTOOPT]	Protocol not available.
7935		[ENOSPC]	No space left on device.
7936	OB XSR	[ENOSR]	No STREAM resources.

7937	OB XSR	[ENOSTR]	Not a STREAM.
7938		[ENOSYS]	Functionality not supported.
7939		[ENOTCONN]	The socket is not connected.
7940		[ENOTDIR]	Not a directory or a symbolic link to a directory.
7941		[ENOTEMPTY]	Directory not empty.
7942		[ENOTRECOVERABLE]	
7943			State not recoverable.
7944		[ENOTSOCK]	Not a socket.
7945		[ENOTSUP]	Not supported (may be the same value as [EOPNOTSUPP]).
7946		[ENOTTY]	Inappropriate I/O control operation.
7947		[ENXIO]	No such device or address.
7948		[EOPNOTSUPP]	Operation not supported on socket (may be the same value as [ENOTSUP]).
7949			
7950		[EOVERFLOW]	Value too large to be stored in data type.
7951		[EOWNERDEAD]	Previous owner died.
7952		[EPERM]	Operation not permitted.
7953		[EPIPE]	Broken pipe.
7954		[EPROTO]	Protocol error.
7955		[EPROTONOSUPPORT]	
7956			Protocol not supported.
7957		[EPROTOTYPE]	Protocol wrong type for socket.
7958		[ERANGE]	Result too large.
7959		[EROFS]	Read-only file system.
7960		[ESPIPE]	Invalid seek.
7961		[ESRCH]	No such process.
7962		[ESTALE]	Reserved.
7963	OB XSR	[ETIME]	Stream <i>ioctl()</i> timeout.
7964		[ETIMEDOUT]	Connection timed out.
7965		[ETXTBSY]	Text file busy.
7966		[EWOULDBLOCK]	Operation would block (may be the same value as [EAGAIN]).
7967		[EXDEV]	Cross-device link.



**APPLICATION USAGE**

Additional error numbers may be defined on conforming systems; see the System Interfaces volume of POSIX.1-2008.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

XSH [Section 2.3](#) (on page 481)

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

Updated for alignment with the POSIX Realtime Extension.

**Issue 6**

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The majority of the error conditions previously marked as extensions are now mandatory, except for the STREAMS-related error conditions.

Values for *errno* are now required to be distinct positive values rather than non-zero values. This change is for alignment with the ISO/IEC 9899: 1999 standard.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #050 is applied, allowing [ENOTSUP] and [EOPNOTSUPP] to be the same values.

The [ENOTRECOVERABLE] and [EOWNERDEAD] errors are added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Functionality relating to the XSI STREAMS option is marked obsolescent.

Functionality relating to the Threads option is moved to the Base.

This reference page is clarified with respect to macros and symbolic constants.

POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0043 [324] is applied.

7997 **NAME**

7998       fcntl.h — file control options

7999 **SYNOPSIS**

8000       #include &lt;fcntl.h&gt;

8001 **DESCRIPTION**

8002       The <fcntl.h> header shall define the following symbolic constants for the *cmd* argument used  
 8003       by *fcntl()*. The values shall be unique and shall be suitable for use in **#if** preprocessing  
 8004       directives.

8005       F\_DUPFD       Duplicate file descriptor.

8006       F\_DUPFD\_CLOEXEC

8007       Duplicate file descriptor with the close-on-exec flag FD\_CLOEXEC set.

8008       F\_GETFD       Get file descriptor flags.

8009       F\_SETFD       Set file descriptor flags.

8010       F\_GETFL       Get file status flags and file access modes.

8011       F\_SETFL       Set file status flags.

8012       F\_GETLK       Get record locking information.

8013       F\_SETLK       Set record locking information.

8014       F\_SETLKW      Set record locking information; wait if blocked.

8015       F\_GETOWN      Get process or process group ID to receive SIGURG signals.

8016       F\_SETOWN      Set process or process group ID to receive SIGURG signals.

8017       The <fcntl.h> header shall define the following symbolic constant used for the *fcntl()* file  
 8018       descriptor flags, which shall be suitable for use in **#if** preprocessing directives.

8019       FD\_CLOEXEC    Close the file descriptor upon execution of an *exec* family function.

8020       The <fcntl.h> header shall also define the following symbolic constants for the *l\_type* argument  
 8021       used for record locking with *fcntl()*. The values shall be unique and shall be suitable for use in  
 8022       **#if** preprocessing directives.

8023       F\_RDLCK       Shared or read lock.

8024       F\_UNLCK       Unlock.

8025       F\_WRLCK       Exclusive or write lock.

8026       The <fcntl.h> header shall define the values used for *l\_whence*, SEEK\_SET, SEEK\_CUR, and  
 8027       SEEK\_END as described in <stdio.h>.

8028       The <fcntl.h> header shall define the following symbolic constants as file creation flags for use  
 8029       in the *oflag* value to *open()* and *openat()*. The values shall be bitwise-distinct and shall be  
 8030       suitable for use in **#if** preprocessing directives.

8031       O\_CLOEXEC      The FD\_CLOEXEC flag associated with the new descriptor shall be set to close  
 8032       the file descriptor upon execution of an *exec* family function.

8033       O\_CREAT        Create file if it does not exist.

8034       O\_DIRECTORY   Fail if file is a non-directory file.

8035		O_EXCL	Exclusive use flag.
8036		O_NOCTTY	Do not assign controlling terminal.
8037		O_NOFOLLOW	Do not follow symbolic links.
8038		O_TRUNC	Truncate flag.
8039		O_TTY_INIT	Set the <b>termios</b> structure terminal parameters to a state that provides conforming behavior; see <a href="#">Section 11.2</a> (on page 205).
8041		The O_TTY_INIT flag can have the value zero and in this case it need not be bitwise-distinct from the other flags.	
8043		The <fcntl.h> header shall define the following symbolic constants for use as file status flags for <i>open()</i> , <i>openat()</i> , and <i>fcntl()</i> . The values shall be suitable for use in <b>#if</b> preprocessing directives.	
8045		O_APPEND	Set append mode.
8046	SIO	O_DSYNC	Write according to synchronized I/O data integrity completion.
8047		O_NONBLOCK	Non-blocking mode.
8048	SIO	O_RSYNC	Synchronized read I/O operations.
8049		O_SYNC	Write according to synchronized I/O file integrity completion.
8050		The <fcntl.h> header shall define the following symbolic constant for use as the mask for file access modes. The value shall be suitable for use in <b>#if</b> preprocessing directives.	
8052		O_ACCMODE	Mask for file access modes.
8053		The <fcntl.h> header shall define the following symbolic constants for use as the file access modes for <i>open()</i> , <i>openat()</i> , and <i>fcntl()</i> . The values shall be unique, except that O_EXEC and O_SEARCH may have equal values. The values shall be suitable for use in <b>#if</b> preprocessing directives.	
8057		O_EXEC	Open for execute only (non-directory files). The result is unspecified if this flag is applied to a directory.
8059		O_RDONLY	Open for reading only.
8060		O_RDWR	Open for reading and writing.
8061		O_SEARCH	Open directory for search only. The result is unspecified if this flag is applied to a non-directory file.
8063		O_WRONLY	Open for writing only.
8064		The <fcntl.h> header shall define the symbolic constants for file modes for use as values of <b>mode_t</b> as described in <sys/stat.h>.	
8066		The <fcntl.h> header shall define the following symbolic constant as a special value used in place of a file descriptor for the <i>*at()</i> functions which take a directory file descriptor as a parameter:	
8069		AT_FDCWD	Use the current working directory to determine the target of relative file paths.
8070		The <fcntl.h> header shall define the following symbolic constant as a value for the <i>flag</i> used by <i>faccessat()</i> :	
8072		AT_EACCESS	Check access using effective user and group ID.
8073		The <fcntl.h> header shall define the following symbolic constant as a value for the <i>flag</i> used by	

8074 *fstatat()*, *fchmodat()*, *fchownat()*, and *utimensat()*:

8075 AT\_SYMLINK\_NOFOLLOW

8076 Do not follow symbolic links.

8077 The <fcntl.h> header shall define the following symbolic constant as a value for the flag used by *linkat()*:

8079 AT\_SYMLINK\_FOLLOW

8080 Follow symbolic link.

8081 The <fcntl.h> header shall define the following symbolic constant as a value for the flag used by *unlinkat()*:

8083 AT\_REMOVEDIR

8084 Remove directory instead of file.

8085 ADV The <fcntl.h> header shall define the following symbolic constants for the *advice* argument used by *posix\_fadvise()*:

8087 POSIX\_FADV\_DONTNEED

8088 The application expects that it will not access the specified data in the near future.

8089 POSIX\_FADV\_NOREUSE

8090 The application expects to access the specified data once and then not reuse it thereafter.

8091 POSIX\_FADV\_NORMAL

8092 The application has no advice to give on its behavior with respect to the specified data. It is the default characteristic if no advice is given for an open file.

8094 POSIX\_FADV\_RANDOM

8095 The application expects to access the specified data in a random order.

8096 POSIX\_FADV\_SEQUENTIAL

8097 The application expects to access the specified data sequentially from lower offsets to higher offsets.

8099 POSIX\_FADV\_WILLNEED

8100 The application expects to access the specified data in the near future.

8101 The <fcntl.h> header shall define the **flock** structure describing a file lock. It shall include the following members:

8103 short l\_type Type of lock; F\_RDLCK, F\_WRLCK, F\_UNLCK.

8104 short l\_whence Flag for starting offset.

8105 off\_t l\_start Relative offset in bytes.

8106 off\_t l\_len Size; if 0 then until EOF.

8107 pid\_t l\_pid Process ID of the process holding the lock; returned with F\_GETLK.

8108 The <fcntl.h> header shall define the **mode\_t**, **off\_t**, and **pid\_t** types as described in <sys/types.h>.

8110 The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

8112 int creat(const char \*, mode\_t);

8113 int fcntl(int, int, ...);

8114 int open(const char \*, int, ...);

8115 int openat(int, const char \*, int, ...);

```
8116 ADV int posix_fadvise(int, off_t, off_t, int);
8117 int posix_fallocate(int, off_t, off_t);
```

8118 Inclusion of the <fcntl.h> header may also make visible all symbols from <sys/stat.h> and  
8119 <unistd.h>.

## 8120 APPLICATION USAGE

8121 Although no existing implementation defines AT\_SYMLINK\_FOLLOW and  
8122 AT\_SYMLINK\_NOFOLLOW as the same numeric value, POSIX.1-2008 does not prohibit that as  
8123 the two constants are not used with the same interfaces.

## 8124 RATIONALE

8125 While many of the symbolic constants introduced in the <fcntl.h> header do not strictly need to  
8126 be used in #if preprocessor directives, widespread historic practice has defined them as macros  
8127 that are usable in such constructs, and examination of existing applications has shown that they  
8128 are occasionally used in such a way. Therefore it was decided to retain this requirement on an  
8129 implementation in POSIX.1-2008.

## 8130 FUTURE DIRECTIONS

8131 None.

## 8132 SEE ALSO

8133 <stdio.h>, <sys/stat.h>, <sys/types.h>, <unistd.h>

8134 XSH *creat()*, *exec*, *fcntl()*, *futimens()*, *open()*, *posix\_fadvise()*, *posix\_fallocate()*, *posix\_madvise()*

## 8135 CHANGE HISTORY

8136 First released in Issue 1. Derived from Issue 1 of the SVID.

### 8137 Issue 5

8138 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension.

### 8139 Issue 6

8140 The following changes are made for alignment with the ISO POSIX-1: 1996 standard:

- 8141 • O\_DSYNC and O\_RSYNC are marked as part of the Synchronized Input and Output  
8142 option.

8143 The following new requirements on POSIX implementations derive from alignment with the  
8144 Single UNIX Specification:

- 8145 • The definition of the **mode\_t**, **off\_t**, and **pid\_t** types is mandated.

8146 The F\_GETOWN and F\_SETOWN values are added for sockets.

8147 The *posix\_fadvise()*, *posix\_fallocate()*, and *posix\_madvise()* functions are added for alignment with  
8148 IEEE Std 1003.1d-1999.

8149 IEEE PASC Interpretation 1003.1 #102 is applied, moving the prototype for *posix\_madvise()* to  
8150 <sys/mman.h>.

8151 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/18 is applied, updating the prototypes for  
8152 *posix\_fadvise()* and *posix\_fallocate()* to be large file-aware, using **off\_t** instead of **size\_t**.

### 8153 Issue 7

8154 Austin Group Interpretation 1003.1-2001 #144 is applied, adding the O\_TTY\_INIT flag.

8155 Austin Group Interpretation 1003.1-2001 #171 is applied, adding support to set the  
8156 FD\_CLOEXEC flag atomically at *open()*, and adding the F\_DUPFD\_CLOEXEC flag.

8157 The *openat()* function is added from The Open Group Technical Standard, 2006, Extended API  
8158 Set Part 2.

8159 Additional flags are added to support *faccessat()*, *fchmodat()*, *fchownat()*, *fstatat()*, *linkat()*,  
8160 *open()*, *openat()*, and *unlinkat()*.

8161 This reference page is clarified with respect to macros and symbolic constants.

8162 Changes are made related to support for finegrained timestamps.

8163 Changes are made to allow a directory to be opened for searching.

8164 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0044 [274] and XBD/TC1-2008/0045  
8165 [78,432] are applied.

**NAME**

fenv.h — floating-point environment

**SYNOPSIS**

#include &lt;fenv.h&gt;

**DESCRIPTION**

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The <fenv.h> header shall define the following data types through **typedef**:

**fenv\_t** Represents the entire floating-point environment. The floating-point environment refers collectively to any floating-point status flags and control modes supported by the implementation.

**fexcept\_t** Represents the floating-point status flags collectively, including any status the implementation associates with the flags. A floating-point status flag is a system variable whose value is set (but never cleared) when a floating-point exception is raised, which occurs as a side-effect of exceptional floating-point arithmetic to provide auxiliary information. A floating-point control mode is a system variable whose value may be set by the user to affect the subsequent behavior of floating-point arithmetic.

The <fenv.h> header shall define each of the following macros if and only if the implementation supports the floating-point exception by means of the floating-point functions *feclearexcept()*, *fegetexceptflag()*, *feraiseexcept()*, *fesetexceptflag()*, and *fetestexcept()*. The defined macros shall expand to integer constant expressions with values that are bitwise-distinct.

```
FE_DIVBYZERO
FE_INEXACT
FE_INVALID
FE_OVERFLOW
FE_UNDERFLOW
```

If the implementation supports the IEC 60559 Floating-Point option, all five macros shall be defined. Additional implementation-defined floating-point exceptions with macros beginning with FE\_ and an uppercase letter may also be specified by the implementation.

The <fenv.h> header shall define the macro FE\_ALL\_EXCEPT as the bitwise-inclusive OR of all floating-point exception macros defined by the implementation, if any. If no such macros are defined, then the macro FE\_ALL\_EXCEPT shall be defined as zero.

The <fenv.h> header shall define each of the following macros if and only if the implementation supports getting and setting the represented rounding direction by means of the *fegetround()* and *fesetround()* functions. The defined macros shall expand to integer constant expressions whose values are distinct non-negative values.

```
FE_DOWNWARD
FE_TONEAREST
FE_TOWARDZERO
FE_UPWARD
```

If the implementation supports the IEC 60559 Floating-Point option, all four macros shall be defined. Additional implementation-defined rounding directions with macros beginning with FE\_ and an uppercase letter may also be specified by the implementation.



The <fenv.h> header shall define the following macro, which represents the default floating-point environment (that is, the one installed at program startup) and has type pointer to const-qualified **fenv\_t**. It can be used as an argument to the functions within the <fenv.h> header that manage the floating-point environment.

```
FE_DFL_ENV
```

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
int  feclearexcept(int);
int  fegetenv(fenv_t *);
int  fegetexceptflag(fexcept_t *, int);
int  fegetround(void);
int  feholdexcept(fenv_t *);
int  feraiseexcept(int);
int  fesetenv(const fenv_t *);
int  fesetexceptflag(const fexcept_t *, int);
int  fesetround(int);
int  fetestexcept(int);
int  feupdateenv(const fenv_t *);
```

The FENV\_ACCESS pragma provides a means to inform the implementation when an application might access the floating-point environment to test floating-point status flags or run under non-default floating-point control modes. The pragma shall occur either outside external declarations or preceding all explicit declarations and statements inside a compound statement. When outside external declarations, the pragma takes effect from its occurrence until another FENV\_ACCESS pragma is encountered, or until the end of the translation unit. When inside a compound statement, the pragma takes effect from its occurrence until another FENV\_ACCESS pragma is encountered (including within a nested compound statement), or until the end of the compound statement; at the end of a compound statement the state for the pragma is restored to its condition just before the compound statement. If this pragma is used in any other context, the behavior is undefined. If part of an application tests floating-point status flags, sets floating-point control modes, or runs under non-default mode settings, but was translated with the state for the FENV\_ACCESS pragma off, the behavior is undefined. The default state (on or off) for the pragma is implementation-defined. (When execution passes from a part of the application translated with FENV\_ACCESS off to a part translated with FENV\_ACCESS on, the state of the floating-point status flags is unspecified and the floating-point control modes have their default settings.)

## APPLICATION USAGE

This header is designed to support the floating-point exception status flags and directed-rounding control modes required by the IEC 60559:1989 standard, and other similar floating-point state information. Also it is designed to facilitate code portability among all systems.

Certain application programming conventions support the intended model of use for the floating-point environment:

- A function call does not alter its caller's floating-point control modes, clear its caller's floating-point status flags, nor depend on the state of its caller's floating-point status flags unless the function is so documented.
- A function call is assumed to require default floating-point control modes, unless its documentation promises otherwise.



- A function call is assumed to have the potential for raising floating-point exceptions, unless its documentation promises otherwise.

With these conventions, an application can safely assume default floating-point control modes (or be unaware of them). The responsibilities associated with accessing the floating-point environment fall on the application that does so explicitly.

Even though the rounding direction macros may expand to constants corresponding to the values of FLT\_ROUNDS, they are not required to do so.

For example:

```
#include <fenv.h>
void f(double x)
{
    #pragma STDC FENV_ACCESS ON
    void g(double);
    void h(double);
    /* ... */
    g(x + 1);
    h(x + 1);
    /* ... */
}
```

If the function `g()` might depend on status flags set as a side-effect of the first `x+1`, or if the second `x+1` might depend on control modes set as a side-effect of the call to function `g()`, then the application shall contain an appropriately placed invocation as follows:

```
#pragma STDC FENV_ACCESS ON
```

## RATIONALE

### The `fexcept_t` Type

`fexcept_t` does not have to be an integer type. Its values must be obtained by a call to `fegetexceptflag()`, and cannot be created by logical operations from the exception macros. An implementation might simply implement `fexcept_t` as an `int` and use the representations reflected by the exception macros, but is not required to; other representations might contain extra information about the exceptions. `fexcept_t` might be a `struct` with a member for each exception (that might hold the address of the first or last floating-point instruction that caused that exception). The ISO/IEC 9899:1999 standard makes no claims about the internals of an `fexcept_t`, and so the user cannot inspect it.

### Exception and Rounding Macros

Macros corresponding to unsupported modes and rounding directions are not defined by the implementation and must not be defined by the application. An application might use `#ifdef` to test for this.

## FUTURE DIRECTIONS

None.

## SEE ALSO

XSH [`feclearexcept\(\)`](#), [`fegetenv\(\)`](#), [`fegetexceptflag\(\)`](#), [`fegetround\(\)`](#), [`feholdexcept\(\)`](#), [`feraiseexcept\(\)`](#), [`fetestexcept\(\)`](#), [`feupdateenv\(\)`](#)

**CHANGE HISTORY**

First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

The return types for *feclearexcept()*, *fegetexceptflag()*, *feraiseexcept()*, *fesetexceptflag()*, *fegetenv()*, *fesetenv()*, and *feupdateenv()* are changed from **void** to **int** for alignment with the ISO/IEC 9899:1999 standard, Defect Report 202.

**Issue 7**

ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #37 (SD5-XBD-ERN-49) is applied.

ISO/IEC 9899:1999 standard, Technical Corrigendum 3 #36 is applied.

SD5-XBD-ERN-48 and SD5-XBD-ERN-69 are applied.

This reference page is clarified with respect to macros and symbolic constants.

**NAME**

float.h — floating types

**SYNOPSIS**

#include &lt;float.h&gt;

**DESCRIPTION**

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The characteristics of floating types are defined in terms of a model that describes a representation of floating-point numbers and values that provide information about an implementation's floating-point arithmetic.

The following parameters are used to define the model for each floating-point type:

*s* Sign ( $\pm 1$ ).

*b* Base or radix of exponent representation (an integer  $> 1$ ).

*e* Exponent (an integer between a minimum  $e_{\min}$  and a maximum  $e_{\max}$ ).

*p* Precision (the number of base-*b* digits in the significand).

*f<sub>k</sub>* Non-negative integers less than *b* (the significand digits).

A floating-point number *x* is defined by the following model:

$$x = sb^e \sum_{k=1}^p f_k b^{-k}, e_{\min} \leq e \leq e_{\max}$$

In addition to normalized floating-point numbers ( $f_1 > 0$  if  $x \neq 0$ ), floating types may be able to contain other kinds of floating-point numbers, such as subnormal floating-point numbers ( $x \neq 0$ ,  $e = e_{\min}$ ,  $f_1 = 0$ ) and unnormalized floating-point numbers ( $x \neq 0$ ,  $e > e_{\min}$ ,  $f_1 = 0$ ), and values that are not floating-point numbers, such as infinities and NaNs. A NaN is an encoding signifying Not-a-Number. A *quiet NaN* propagates through almost every arithmetic operation without raising a floating-point exception; a *signaling NaN* generally raises a floating-point exception when occurring as an arithmetic operand.

An implementation may give zero and non-numeric values, such as infinities and NaNs, a sign, or may leave them unsigned. Wherever such values are unsigned, any requirement in POSIX.1-2008 to retrieve the sign shall produce an unspecified sign and any requirement to set the sign shall be ignored.

The accuracy of the floating-point operations ('+', '-', '\*', '/') and of the functions in <math.h> and <complex.h> that return floating-point results is implementation-defined, as is the accuracy of the conversion between floating-point internal representations and string representations performed by the functions in <stdio.h>, <stdlib.h>, and <wchar.h>. The implementation may state that the accuracy is unknown.

All integer values in the <float.h> header, except FLT\_ROUNDS, shall be constant expressions suitable for use in #if preprocessing directives; all floating values shall be constant expressions. All except DECIMAL\_DIG, FLT\_EVAL\_METHOD, FLT\_RADIX, and FLT\_ROUNDS have separate names for all three floating-point types. The floating-point model representation is provided for all values except FLT\_EVAL\_METHOD and FLT\_ROUNDS.

The rounding mode for floating-point addition is characterized by the implementation-defined

8349 value of FLT\_ROUNDS:

- 8350 -1 Indeterminable.
- 8351 0 Toward zero.
- 8352 1 To nearest.
- 8353 2 Toward positive infinity.
- 8354 3 Toward negative infinity.

8355 All other values for FLT\_ROUNDS characterize implementation-defined rounding behavior.

8356 The values of operations with floating operands and values subject to the usual arithmetic  
 8357 conversions and of floating constants are evaluated to a format whose range and precision may  
 8358 be greater than required by the type. The use of evaluation formats is characterized by the  
 8359 implementation-defined value of FLT\_EVAL\_METHOD:

- 8360 -1 Indeterminable.
- 8361 0 Evaluate all operations and constants just to the range and precision of the type.
- 8362 1 Evaluate operations and constants of type **float** and **double** to the range and precision of  
 8363 the **double** type; evaluate **long double** operations and constants to the range and precision  
 8364 of the **long double** type.
- 8365 2 Evaluate all operations and constants to the range and precision of the **long double** type.

8366 All other negative values for FLT\_EVAL\_METHOD characterize implementation-defined  
 8367 behavior.

8368 The <float.h> header shall define the following values as constant expressions with  
 8369 implementation-defined values that are greater or equal in magnitude (absolute value) to those  
 8370 shown, with the same sign.

- 8371 • Radix of exponent representation,  $b$ .  
 8372 FLT\_RADIX 2
- 8373 • Number of base-FLT\_RADIX digits in the floating-point significand,  $p$ .  
 8374 FLT\_MANT\_DIG  
 8375 DBL\_MANT\_DIG  
 8376 LDBL\_MANT\_DIG
- 8377 • Number of decimal digits,  $n$ , such that any floating-point number in the widest supported  
 8378 floating type with  $p_{\max}$  radix  $b$  digits can be rounded to a floating-point number with  $n$   
 8379 decimal digits and back again without change to the value.

$$\begin{cases} p_{\max} \log_{10} b & \text{if } b \text{ is a power of } 10 \\ \left\lceil 1 + p_{\max} \log_{10} b \right\rceil & \text{otherwise} \end{cases}$$

8380 DECIMAL\_DIG 10

- 8381 • Number of decimal digits,  $q$ , such that any floating-point number with  $q$  decimal digits can  
8382 be rounded into a floating-point number with  $p$  radix  $b$  digits and back again without  
8383 change to the  $q$  decimal digits.

$$\begin{cases} p \log_{10} b & \text{if } b \text{ is a power of } 10 \\ \left\lceil (p-1) \log_{10} b \right\rceil & \text{otherwise} \end{cases}$$

8384 FLT\_DIG 6

8385 DBL\_DIG 10

8386 LDBL\_DIG 10

- 8387 • Minimum negative integer such that FLT\_RADIX raised to that power minus 1 is a  
8388 normalized floating-point number,  $e_{\min}$ .

8389 FLT\_MIN\_EXP

8390 DBL\_MIN\_EXP

8391 LDBL\_MIN\_EXP

- 8392 • Minimum negative integer such that 10 raised to that power is in the range of normalized  
8393 floating-point numbers.

$$\left\lceil \log_{10} b^{e_{\min}-1} \right\rceil$$

8394 FLT\_MIN\_10\_EXP -37

8395 DBL\_MIN\_10\_EXP -37

8396 LDBL\_MIN\_10\_EXP -37

- 8397 • Maximum integer such that FLT\_RADIX raised to that power minus 1 is a representable  
8398 finite floating-point number,  $e_{\max}$ .

8399 FLT\_MAX\_EXP

8400 DBL\_MAX\_EXP

8401 LDBL\_MAX\_EXP

8402 CX Additionally, FLT\_MAX\_EXP shall be at least as large as FLT\_MANT\_DIG,  
8403 DBL\_MAX\_EXP shall be at least as large as DBL\_MANT\_DIG, and LDBL\_MAX\_EXP shall  
8404 be at least as large as LDBL\_MANT\_DIG; which has the effect that FLT\_MAX, DBL\_MAX,  
8405 and LDBL\_MAX are integral.

- 8406 • Maximum integer such that 10 raised to that power is in the range of representable finite  
8407 floating-point numbers.

$$\left\lceil \log_{10}((1 - b^{-p}) b^{e_{\max}}) \right\rceil$$

8408 FLT\_MAX\_10\_EXP +37

8409 DBL\_MAX\_10\_EXP +37

8410 LDBL\_MAX\_10\_EXP +37

8411 The <float.h> header shall define the following values as constant expressions with  
8412 implementation-defined values that are greater than or equal to those shown:

- 8413
- Maximum representable finite floating-point number.

$$(1 - b^{-p}) b^{\epsilon_{\max}}$$

8414 FLT\_MAX 1E+37

8415 DBL\_MAX 1E+37

8416 LDBL\_MAX 1E+37

8417 The <float.h> header shall define the following values as constant expressions with  
8418 implementation-defined (positive) values that are less than or equal to those shown:

- 8419
- The difference between 1 and the least value greater than 1 that is representable in the  
8420 given floating-point type,  $b^{1-p}$ .

8421 FLT\_EPSILON 1E-5

8422 DBL\_EPSILON 1E-9

8423 LDBL\_EPSILON 1E-9

- 8424
- Minimum normalized positive floating-point number,  $b^{\epsilon_{\min}-1}$ .

8425 FLT\_MIN 1E-37

8426 DBL\_MIN 1E-37

8427 LDBL\_MIN 1E-37

## 8428 APPLICATION USAGE

8429 None.

## 8430 RATIONALE

8431 All known hardware floating-point formats satisfy the property that the exponent range is larger  
8432 than the number of mantissa digits. The ISO C standard permits a floating-point format where  
8433 this property is not true, such that the largest finite value would not be integral; however, it is  
8434 unlikely that there will ever be hardware support for such a floating-point format, and it  
8435 introduces boundary cases that portable programs should not have to be concerned with (for  
8436 example, a non-integral DBL\_MAX means that *ceil()* would have to worry about overflow).  
8437 Therefore, this standard imposes an additional requirement that the largest representable finite  
8438 value is integral.

## 8439 FUTURE DIRECTIONS

8440 None.

## 8441 SEE ALSO

8442 <complex.h>, <math.h>, <stdio.h>, <stdlib.h>, <wchar.h>

8443 **CHANGE HISTORY**

8444 First released in Issue 4. Derived from the ISO C standard.

8445 **Issue 6**8446 The description of the operations with floating-point values is updated for alignment with the  
8447 ISO/IEC 9899:1999 standard.8448 **Issue 7**8449 ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #4 (SD5-XBD-ERN-50) and #5  
8450 (SD5-XBD-ERN-51) are applied.8451 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0046 [346] and XBD/TC1-2008/0047  
8452 [346] are applied.

**NAME**

fmtmsg.h — message display structures

**SYNOPSIS**XSI `#include <fmtmsg.h>`**DESCRIPTION**

The &lt;fmtmsg.h&gt; header shall define the following symbolic constants:

MM_HARD	Source of the condition is hardware.
MM_SOFT	Source of the condition is software.
MM_FIRM	Source of the condition is firmware.
MM_APPL	Condition detected by application.
MM_UTIL	Condition detected by utility.
MM_OPSYS	Condition detected by operating system.
MM_RECOVER	Recoverable error.
MM_NRECOV	Non-recoverable error.
MM_HALT	Error causing application to halt.
MM_ERROR	Application has encountered a non-fatal fault.
MM_WARNING	Application has detected unusual non-error condition.
MM_INFO	Informative message.
MM_NOSEV	No severity level provided for the message.
MM_PRINT	Display message on standard error.
MM_CONSOLE	Display message on system console.

The table below indicates the null values and identifiers for *fmtmsg()* arguments. The <fmtmsg.h> header shall define the symbolic constants in the **Identifier** column, which shall have the type indicated in the **Type** column:

Argument	Type	Null-Value	Identifier
<i>label</i>	<b>char *</b>	(char*)0	MM_NULLLBL
<i>severity</i>	<b>int</b>	0	MM_NULLSEV
<i>class</i>	<b>long</b>	0L	MM_NULLMC
<i>text</i>	<b>char *</b>	(char*)0	MM_NULLTXT
<i>action</i>	<b>char *</b>	(char*)0	MM_NULLACT
<i>tag</i>	<b>char *</b>	(char*)0	MM_NULLTAG

The <fmtmsg.h> header shall also define the following symbolic constants for use as return values for *fmtmsg()*:

MM_OK	The function succeeded.
MM_NOTOK	The function failed completely.
MM_NOMSG	The function was unable to generate a message on standard error, but otherwise succeeded.



8490 MM\_NOCON The function was unable to generate a console message, but otherwise  
8491 succeeded.

8492 The following shall be declared as a function and may also be defined as a macro. A function  
8493 prototype shall be provided.

8494 int fmtmsg(long, const char \*, int,  
8495 const char \*, const char \*, const char \*);

#### 8496 APPLICATION USAGE

8497 None.

#### 8498 RATIONALE

8499 None.

#### 8500 FUTURE DIRECTIONS

8501 None.

#### 8502 SEE ALSO

8503 XSH *fmtmsg()*

#### 8504 CHANGE HISTORY

8505 First released in Issue 4, Version 2.

#### 8506 Issue 7

8507 This reference page is clarified with respect to macros and symbolic constants.

8508 **NAME**8509 `fnmatch.h` — filename-matching types8510 **SYNOPSIS**8511 `#include <fnmatch.h>`8512 **DESCRIPTION**

8513 The &lt;fnmatch.h&gt; header shall define the following symbolic constants:

8514 `FNM_NOMATCH` The string does not match the specified pattern.8515 `FNM_PATHNAME` <slash> in *string* only matches <slash> in *pattern*.8516 `FNM_PERIOD` Leading <period> in *string* must be exactly matched by <period> in  
8517 *pattern*.8518 `FNM_NOESCAPE` Disable backslash escaping.8519 The following shall be declared as a function and may also be defined as a macro. A function  
8520 prototype shall be provided.8521 `int fnmatch(const char *, const char *, int);`8522 **APPLICATION USAGE**

8523 None.

8524 **RATIONALE**

8525 None.

8526 **FUTURE DIRECTIONS**

8527 None.

8528 **SEE ALSO**8529 XSH *fnmatch()*8530 **CHANGE HISTORY**

8531 First released in Issue 4. Derived from the ISO POSIX-2 standard.

8532 **Issue 6**8533 The `FNM_NOSYS` constant is marked obsolescent.8534 **Issue 7**8535 The obsolescent `FNM_NOSYS` constant is removed.

8536 This reference page is clarified with respect to macros and symbolic constants.

**NAME**

ftw.h — file tree traversal

**SYNOPSIS**XSI `#include <ftw.h>`**DESCRIPTION**

The <ftw.h> header shall define the **FTW** structure, which shall include at least the following members:

```
int  base
int  level
```

The <ftw.h> header shall define the following symbolic constants for use as values of the third argument to the application-supplied function that is passed as the second argument to *ftw()* and *nftw()*:

```
FTW_F      Non-directory file.
FTW_D      Directory.
FTW_DNR    Directory without read permission.
FTW_DP     Directory with subdirectories visited.
FTW_NS     Unknown type; stat() failed.
FTW_SL     Symbolic link.
FTW_SLN    Symbolic link that names a nonexistent file.
```

The <ftw.h> header shall define the following symbolic constants for use as values of the fourth argument to *nftw()*:

```
FTW_PHYS    Physical walk, does not follow symbolic links. Otherwise, nftw() follows
links but does not walk down any path that crosses itself.
FTW_MOUNT   The walk does not cross a mount point.
FTW_DEPTH   All subdirectories are visited before the directory itself.
FTW_CHDIR   The walk changes to each directory before reading it.
```

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
OB int ftw(const char *, int (*)(const char *, const struct stat *,
int), int);
int nftw(const char *, int (*)(const char *, const struct stat *,
int, struct FTW *), int, int);
```

The <ftw.h> header shall define the **stat** structure and the symbolic names for *st\_mode* and the file type test macros as described in <sys/stat.h>.

Inclusion of the <ftw.h> header may also make visible all symbols from <sys/stat.h>.

8572 **APPLICATION USAGE**

8573 None.

8574 **RATIONALE**

8575 None.

8576 **FUTURE DIRECTIONS**

8577 None.

8578 **SEE ALSO**8579 [`<sys/stat.h>`](#)8580 XSH [`ftw\(\)`](#), [`nftw\(\)`](#)8581 **CHANGE HISTORY**

8582 First released in Issue 1. Derived from Issue 1 of the SVID.

8583 **Issue 5**

8584 A description of FTW\_DP is added.

8585 **Issue 7**8586 The `ftw()` function is marked obsolescent.

8587 This reference page is clarified with respect to macros and symbolic constants.

8588 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0048 [403] is applied.

**NAME**

glob.h — pathname pattern-matching types

**SYNOPSIS**

```
#include <glob.h>
```

**DESCRIPTION**

The <glob.h> header shall define the structures and symbolic constants used by the *glob()* function.

The <glob.h> header shall define the **glob\_t** structure type, which shall include at least the following members:

```
size_t    gl_pathc  Count of paths matched by pattern.
char      **gl_pathv Pointer to a list of matched pathnames.
size_t    gl_offs   Slots to reserve at the beginning of gl_pathv.
```

The <glob.h> header shall define the **size\_t** type as described in <sys/types.h>.

The <glob.h> header shall define the following symbolic constants as values for the *flags* argument:

GLOB_APPEND	Append generated pathnames to those previously obtained.
GLOB_DOOFFS	Specify how many null pointers to add to the beginning of <i>gl_pathv</i> .
GLOB_ERR	Cause <i>glob()</i> to return on error.
GLOB_MARK	Each pathname that is a directory that matches <i>pattern</i> has a <slash> appended.
GLOB_NOCHECK	If <i>pattern</i> does not match any pathname, then return a list consisting of only <i>pattern</i> .
GLOB_NOESCAPE	Disable backslash escaping.
GLOB_NOSORT	Do not sort the pathnames returned.

The <glob.h> header shall define the following symbolic constants as error return values:

GLOB_ABORTED	The scan was stopped because GLOB_ERR was set or (*errfunc)() returned non-zero.
GLOB_NOMATCH	The pattern does not match any existing pathname, and GLOB_NOCHECK was not set in <i>flags</i> .
GLOB_NOSPACE	An attempt to allocate memory failed.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
int  glob(const char *restrict, int, int (*)(const char *, int),
         glob_t *restrict);
void globfree(glob_t *);
```

8624 **APPLICATION USAGE**

8625       None.

8626 **RATIONALE**

8627       None.

8628 **FUTURE DIRECTIONS**

8629       None.

8630 **SEE ALSO**

8631       &lt;sys/types.h&gt;

8632       XSH *glob()*8633 **CHANGE HISTORY**

8634       First released in Issue 4. Derived from the ISO POSIX-2 standard.

8635 **Issue 6**8636       The **restrict** keyword is added to the prototype for *glob()*.

8637       The GLOB\_NOSYS constant is marked obsolescent.

8638       IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/8 is applied, correcting the *glob()*  
8639       prototype definition by removing the **restrict** qualifier from the function pointer argument.8640 **Issue 7**8641       SD5-XBD-ERN-56 is applied, adding a reference to <sys/types.h> for the **size\_t**

8642       The obsolescent GLOB\_NOSYS constant is removed.

8643       This reference page is clarified with respect to macros and symbolic constants.

**NAME**

grp.h — group structure

**SYNOPSIS**

```
#include <grp.h>
```

**DESCRIPTION**

The <grp.h> header shall declare the **group** structure, which shall include the following members:

```
char    *gr_name  The name of the group.
gid_t   gr_gid    Numerical group ID.
char    **gr_mem  Pointer to a null-terminated array of character
                  pointers to member names.
```

The <grp.h> header shall define the **gid\_t** and **size\_t** types as described in <sys/types.h>.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
XSI void      endgrent(void);
      struct group *getgrent(void);
      struct group *getgrgid(gid_t);
      int          getgrgid_r(gid_t, struct group *, char *,
                              size_t, struct group **);
      struct group *getgrnam(const char *);
      int          getgrnam_r(const char *, struct group *, char *,
                              size_t, struct group **);
XSI void      setgrent(void);
```

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

<sys/types.h>

XSH *endgrent()*, *getgrgid()*, *getgrnam()*

**CHANGE HISTORY**

First released in Issue 1.

**Issue 5**

The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

**Issue 6**

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The definition of **gid\_t** is mandated.
- The *getgrgid\_r()* and *getgrnam\_r()* functions are marked as part of the Thread-Safe Functions option.

**Issue 7**

SD5-XBD-ERN-56 is applied, adding a reference to **<sys/types.h>** for the **size\_t** type.  
POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0049 [24] is applied.



8689 **NAME**

8690 iconv.h — codeset conversion facility

8691 **SYNOPSIS**

8692 #include <iconv.h>

8693 **DESCRIPTION**

8694 The <iconv.h> header shall define the following types:

8695 **iconv\_t** Identifies the conversion from one codeset to another.

8696 **size\_t** As described in <sys/types.h>.

8697 The following shall be declared as functions and may also be defined as macros. Function  
8698 prototypes shall be provided.

8699 size\_t iconv(iconv\_t, char \*\*restrict, size\_t \*restrict,  
8700 char \*\*restrict, size\_t \*restrict);  
8701 int iconv\_close(iconv\_t);  
8702 iconv\_t iconv\_open(const char \*, const char \*);

8703 **APPLICATION USAGE**

8704 None.

8705 **RATIONALE**

8706 None.

8707 **FUTURE DIRECTIONS**

8708 None.

8709 **SEE ALSO**

8710 <sys/types.h>

8711 XSH *iconv()*, *iconv\_close()*, *iconv\_open()*

8712 **CHANGE HISTORY**

8713 First released in Issue 4.

8714 **Issue 6**

8715 The **restrict** keyword is added to the prototype for *iconv()*.

8716 **Issue 7**

8717 SD5-XBD-ERN-56 is applied, adding a reference to <sys/types.h> for the **size\_t** type.

8718 The <iconv.h> header is moved from the XSI option to the Base.

## 8719 NAME

8720 inttypes.h — fixed size integer types

## 8721 SYNOPSIS

8722 #include &lt;inttypes.h&gt;

## 8723 DESCRIPTION

8724 CX Some of the functionality described on this reference page extends the ISO C standard.  
 8725 Applications shall define the appropriate feature test macro (see XSH Section 2.2, on page 472) to  
 8726 enable the visibility of these symbols in this header.

8727 The &lt;inttypes.h&gt; header shall include the &lt;stdint.h&gt; header.

8728 The &lt;inttypes.h&gt; header shall define at least the following types:

8729 **imaxdiv\_t** Structure type that is the type of the value returned by the *imaxdiv()* function.8730 CX **wchar\_t** As described in <stddef.h>.

8731 The <inttypes.h> header shall define the following macros. Each expands to a character string  
 8732 literal containing a conversion specifier, possibly modified by a length modifier, suitable for use  
 8733 within the *format* argument of a formatted input/output function when converting the  
 8734 corresponding integer type. These macros have the general form of PRI (character string literals  
 8735 for the *fprintf()* and *fwprintf()* family of functions) or SCN (character string literals for the  
 8736 *fscanf()* and *fwscanf()* family of functions), followed by the conversion specifier, followed by a  
 8737 name corresponding to a similar type name in <stdint.h>. In these names, *N* represents the  
 8738 width of the type as described in <stdint.h>. For example, *PRIdFAST32* can be used in a format  
 8739 string to print the value of an integer of type **int\_fast32\_t**.

8740 The *fprintf()* macros for signed integers are:

8741	<b>PRIdN</b>	<b>PRIdLEASTN</b>	<b>PRIdFASTN</b>	<b>PRIdMAX</b>	<b>PRIdPTR</b>
8742	<b>PRiN</b>	<b>PRiLEASTN</b>	<b>PRiFASTN</b>	<b>PRiMAX</b>	<b>PRiPTR</b>

8743 The *fprintf()* macros for unsigned integers are:

8744	<b>PRIoN</b>	<b>PRIoLEASTN</b>	<b>PRIoFASTN</b>	<b>PRIoMAX</b>	<b>PRIoPTR</b>
8745	<b>PRiUN</b>	<b>PRiULEASTN</b>	<b>PRiUFASTN</b>	<b>PRiUMAX</b>	<b>PRiUPTR</b>
8746	<b>PRiXN</b>	<b>PRiXLEASTN</b>	<b>PRiXFASTN</b>	<b>PRiXMAX</b>	<b>PRiXPTR</b>
8747	<b>PRiXN</b>	<b>PRiXLEASTN</b>	<b>PRiXFASTN</b>	<b>PRiXMAX</b>	<b>PRiXPTR</b>

8748 The *fscanf()* macros for signed integers are:

8749	<b>SCNdN</b>	<b>SCNdLEASTN</b>	<b>SCNdFASTN</b>	<b>SCNdMAX</b>	<b>SCNdPTR</b>
8750	<b>SCNiN</b>	<b>SCNiLEASTN</b>	<b>SCNiFASTN</b>	<b>SCNiMAX</b>	<b>SCNiPTR</b>

8751 The *fscanf()* macros for unsigned integers are:

8752	<b>SCNoN</b>	<b>SCNoLEASTN</b>	<b>SCNoFASTN</b>	<b>SCNoMAX</b>	<b>SCNoPTR</b>
8753	<b>SCNuN</b>	<b>SCNuLEASTN</b>	<b>SCNuFASTN</b>	<b>SCNuMAX</b>	<b>SCNuPTR</b>
8754	<b>SCNxN</b>	<b>SCNxLEASTN</b>	<b>SCNxFASTN</b>	<b>SCNxMAX</b>	<b>SCNxPTR</b>

8755 For each type that the implementation provides in <stdint.h>, the corresponding *fprintf()* and  
 8756 *fwprintf()* macros shall be defined and the corresponding *fscanf()* and *fwscanf()* macros shall be  
 8757 defined unless the implementation does not have a suitable modifier for the type.

8758 The following shall be declared as functions and may also be defined as macros. Function  
 8759 prototypes shall be provided.

```

8760     intmax_t  imaxabs(intmax_t);
8761     imaxdiv_t imaxdiv(intmax_t, intmax_t);
8762     intmax_t  strtoumax(const char *restrict, char **restrict, int);
8763     uintmax_t strtoumax(const char *restrict, char **restrict, int);
8764     intmax_t  wcstoumax(const wchar_t *restrict, wchar_t **restrict, int);
8765     uintmax_t wcstoumax(const wchar_t *restrict, wchar_t **restrict, int);

```

**EXAMPLES**

```

8766     #include <inttypes.h>
8767     #include <wchar.h>
8768     int main(void)
8769     {
8770         uintmax_t i = UINTMAX_MAX; // This type always exists.
8771         wprintf(L"The largest integer value is %020"
8772             PRIxMAX "\n", i);
8773         return 0;
8774     }
8775

```

**APPLICATION USAGE**

8776 The purpose of **<inttypes.h>** is to provide a set of integer types whose definitions are consistent  
8777 across machines and independent of operating systems and other implementation  
8778 idiosyncrasies. It defines, through **typedef**, integer types of various sizes. Implementations are  
8779 free to **typedef** them as ISO C standard integer types or extensions that they support. Consistent  
8780 use of this header will greatly increase the portability of applications across platforms.

**RATIONALE**

8782 The ISO/IEC 9899:1990 standard specified that the language should support four signed and  
8783 unsigned integer data types—**char**, **short**, **int**, and **long**—but placed very little requirement on  
8784 their size other than that **int** and **short** be at least 16 bits and **long** be at least as long as **int** and  
8785 not smaller than 32 bits. For 16-bit systems, most implementations assigned 8, 16, 16, and 32 bits  
8786 to **char**, **short**, **int**, and **long**, respectively. For 32-bit systems, the common practice has been to  
8787 assign 8, 16, 32, and 32 bits to these types. This difference in **int** size can create some problems  
8788 for users who migrate from one system to another which assigns different sizes to integer types,  
8789 because the ISO C standard integer promotion rule can produce silent changes unexpectedly.  
8790 The need for defining an extended integer type increased with the introduction of 64-bit  
8791 systems.

**FUTURE DIRECTIONS**

8793 Macro names beginning with PRI or SCN followed by any lowercase letter or 'X' may be added  
8794 to the macros defined in the **<inttypes.h>** header.

**SEE ALSO**

8796 **<stddef.h>**

8798 XSH Section 2.2 (on page 472), *imaxabs()*, *imaxdiv()*, *strtoumax()*, *wcstoumax()*

**CHANGE HISTORY**

8800 First released in Issue 5.

**Issue 6**

8802 The Open Group Base Resolution bwg97-006 is applied.

8803 This reference page is updated to align with the ISO/IEC 9899:1999 standard.

8804 **Issue 7**  
8805

POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0050 [211] is applied.

8806 **NAME**

8807       iso646.h — alternative spellings

8808 **SYNOPSIS**

8809       #include &lt;iso646.h&gt;

8810 **DESCRIPTION**

8811 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 8812       conflict between the requirements described here and the ISO C standard is unintentional. This  
 8813       volume of POSIX.1-2008 defers to the ISO C standard.

8814       The <iso646.h> header shall define the following eleven macros (on the left) that expand to the  
 8815       corresponding tokens (on the right):

8816	and	&&
8817	and_eq	&=
8818	bitand	&
8819	bitor	
8820	compl	~
8821	not	!
8822	not_eq	!=
8823	or	
8824	or_eq	=
8825	xor	^
8826	xor_eq	^=

8827 **APPLICATION USAGE**

8828       None.

8829 **RATIONALE**

8830       None.

8831 **FUTURE DIRECTIONS**

8832       None.

8833 **SEE ALSO**

8834       None.

8835 **CHANGE HISTORY**

8836       First released in Issue 5. Derived from ISO/IEC 9899:1990/Amendment 1:1995 (E).

8837 **NAME**

8838 langinfo.h — language information constants

8839 **SYNOPSIS**

8840 #include &lt;langinfo.h&gt;

8841 **DESCRIPTION**8842 The <langinfo.h> header shall define the symbolic constants used to identify items of *langinfo*  
8843 data (see *nl\_langinfo()*).8844 The <langinfo.h> header shall define the **locale\_t** type as described in <locale.h>.8845 The <langinfo.h> header shall define the **nl\_item** type as described in <nl\_types.h>.8846 The <langinfo.h> header shall define the following symbolic constants with type **nl\_item**. The  
8847 entries under **Category** indicate in which *setlocale()* category each item is defined.

Constant	Category	Meaning
CODESET	LC_CTYPE	Codeset name.
D_T_FMT	LC_TIME	String for formatting date and time.
D_FMT	LC_TIME	Date format string.
T_FMT	LC_TIME	Time format string.
T_FMT_AMPM	LC_TIME	a.m. or p.m. time format string.
AM_STR	LC_TIME	Ante-meridiem affix.
PM_STR	LC_TIME	Post-meridiem affix.
DAY_1	LC_TIME	Name of the first day of the week (for example, Sunday).
DAY_2	LC_TIME	Name of the second day of the week (for example, Monday).
DAY_3	LC_TIME	Name of the third day of the week (for example, Tuesday).
DAY_4	LC_TIME	Name of the fourth day of the week
		(for example, Wednesday).
DAY_5	LC_TIME	Name of the fifth day of the week (for example, Thursday).
DAY_6	LC_TIME	Name of the sixth day of the week (for example, Friday).
DAY_7	LC_TIME	Name of the seventh day of the week
		(for example, Saturday).
ABDAY_1	LC_TIME	Abbreviated name of the first day of the week.
ABDAY_2	LC_TIME	Abbreviated name of the second day of the week.
ABDAY_3	LC_TIME	Abbreviated name of the third day of the week.
ABDAY_4	LC_TIME	Abbreviated name of the fourth day of the week.
ABDAY_5	LC_TIME	Abbreviated name of the fifth day of the week.
ABDAY_6	LC_TIME	Abbreviated name of the sixth day of the week.
ABDAY_7	LC_TIME	Abbreviated name of the seventh day of the week.
MON_1	LC_TIME	Name of the first month of the year.
MON_2	LC_TIME	Name of the second month.
MON_3	LC_TIME	Name of the third month.
MON_4	LC_TIME	Name of the fourth month.
MON_5	LC_TIME	Name of the fifth month.
MON_6	LC_TIME	Name of the sixth month.
MON_7	LC_TIME	Name of the seventh month.
MON_8	LC_TIME	Name of the eighth month.
MON_9	LC_TIME	Name of the ninth month.
MON_10	LC_TIME	Name of the tenth month.
MON_11	LC_TIME	Name of the eleventh month.
MON_12	LC_TIME	Name of the twelfth month.
ABMON_1	LC_TIME	Abbreviated name of the first month.

Constant	Category	Meaning
ABMON_2	LC_TIME	Abbreviated name of the second month.
ABMON_3	LC_TIME	Abbreviated name of the third month.
ABMON_4	LC_TIME	Abbreviated name of the fourth month.
ABMON_5	LC_TIME	Abbreviated name of the fifth month.
ABMON_6	LC_TIME	Abbreviated name of the sixth month.
ABMON_7	LC_TIME	Abbreviated name of the seventh month.
ABMON_8	LC_TIME	Abbreviated name of the eighth month.
ABMON_9	LC_TIME	Abbreviated name of the ninth month.
ABMON_10	LC_TIME	Abbreviated name of the tenth month.
ABMON_11	LC_TIME	Abbreviated name of the eleventh month.
ABMON_12	LC_TIME	Abbreviated name of the twelfth month.
ERA	LC_TIME	Era description segments.
ERA_D_FMT	LC_TIME	Era date format string.
ERA_D_T_FMT	LC_TIME	Era date and time format string.
ERA_T_FMT	LC_TIME	Era time format string.
ALT_DIGITS	LC_TIME	Alternative symbols for digits.
RADIXCHAR	LC_NUMERIC	Radix character.
THOUSEP	LC_NUMERIC	Separator for thousands.
YESEXPR	LC_MESSAGES	Affirmative response expression.
NOEXPR	LC_MESSAGES	Negative response expression.
CRNCYSTR	LC_MONETARY	Local currency symbol, preceded by '–' if the symbol should appear before the value, '+' if the symbol should appear after the value, or ' .' if the symbol should replace the radix character. If the local currency symbol is the empty string, implementations may return the empty string (" ").

If the locale's values for **p\_cs\_precedes** and **n\_cs\_precedes** do not match, the value of `nl_langinfo(CRNCYSTR)` and `nl_langinfo_l(CRNCYSTR, loc)` is unspecified.

The following shall be declared as a function and may also be defined as a macro. A function prototype shall be provided.

```
char *nl_langinfo(nl_item);
char *nl_langinfo_l(nl_item, locale_t);
```

Inclusion of the <langinfo.h> header may also make visible all symbols from <nl\_types.h>.

## APPLICATION USAGE

Wherever possible, users are advised to use functions compatible with those in the ISO C standard to access items of *langinfo* data. In particular, the `strftime()` function should be used to access date and time information defined in category `LC_TIME`. The `localeconv()` function should be used to access information corresponding to `RADIXCHAR`, `THOUSEP`, and `CRNCYSTR`.

## RATIONALE

None.

## FUTURE DIRECTIONS

None.

## SEE ALSO

Chapter 7 (on page 135), <locale.h>, <nl\_types.h>

XSH `nl_langinfo()`, `localeconv()`, `strfmon()`, `strftime()`

8931 **CHANGE HISTORY**

8932 First released in Issue 2.

8933 **Issue 5**

8934 The constants YESSTR and NOSTR are marked LEGACY.

8935 **Issue 6**

8936 The constants YESSTR and NOSTR are removed.

8937 IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/9 is applied, adding a sentence to  
8938 the “Meaning” column entry for the CRNCYSTR constant. This change is to accommodate  
8939 historic practice.

8940 **Issue 7**

8941 The &lt;langinfo.h&gt; header is moved from the XSI option to the Base.

8942 The `nl_langinfo_l()` function is added from The Open Group Technical Standard, 2006, Extended  
8943 API Set Part 4.

8944 This reference page is clarified with respect to macros and symbolic constants, and a declaration  
8945 for the `locale_t` type is added.

8946 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0051 [107] is applied.



8947 **NAME**

8948 libgen.h — definitions for pattern matching functions

8949 **SYNOPSIS**

8950 XSI `#include <libgen.h>`

8951 **DESCRIPTION**

8952 The following shall be declared as functions and may also be defined as macros. Function  
8953 prototypes shall be provided.

8954 `char *basename(char *);`

8955 `char *dirname(char *);`

8956 **APPLICATION USAGE**

8957 None.

8958 **RATIONALE**

8959 None.

8960 **FUTURE DIRECTIONS**

8961 None.

8962 **SEE ALSO**

8963 XSH *basename()*, *dirname()*

8964 **CHANGE HISTORY**

8965 First released in Issue 4, Version 2.

8966 **Issue 5**

8967 The function prototypes for *basename()* and *dirname()* are changed to indicate that the first  
8968 argument is of type **char \*** rather than **const char \***.

8969 **Issue 6**

8970 The **\_\_loc1** symbol and the *regcmp()* and *regex()* functions are removed.

**NAME**

limits.h — implementation-defined constants

**SYNOPSIS**

```
#include <limits.h>
```

**DESCRIPTION**

Some of the functionality described on this reference page extends the ISO C standard. Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to enable the visibility of these symbols in this header.

Many of the symbols listed here are not defined by the ISO/IEC 9899:1999 standard. Such symbols are not shown as CX shaded, except under the heading “Numerical Limits”.

The <limits.h> header shall define macros and symbolic constants for various limits. Different categories of limits are described below, representing various limits on resources that the implementation imposes on applications. All macros and symbolic constants defined in this header shall be suitable for use in #if preprocessing directives.

Implementations may choose any appropriate value for each limit, provided it is not more restrictive than the Minimum Acceptable Values listed below. Symbolic constant names beginning with \_POSIX may be found in <unistd.h>.

Applications should not assume any particular value for a limit. To achieve maximum portability, an application should not require more resource than the Minimum Acceptable Value quantity. However, an application wishing to avail itself of the full amount of a resource available on an implementation may make use of the value given in <limits.h> on that particular implementation, by using the macros and symbolic constants listed below. It should be noted, however, that many of the listed limits are not invariant, and at runtime, the value of the limit may differ from those given in this header, for the following reasons:

- The limit is pathname-dependent.
- The limit differs between the compile and runtime machines.

For these reasons, an application may use the *fpathconf()*, *pathconf()*, and *sysconf()* functions to determine the actual value of a limit at runtime.

The items in the list ending in \_MIN give the most negative values that the mathematical types are guaranteed to be capable of representing. Numbers of a more negative value may be supported on some implementations, as indicated by the <limits.h> header on the implementation, but applications requiring such numbers are not guaranteed to be portable to all implementations. For positive constants ending in \_MIN, this indicates the minimum acceptable value.

**Runtime Invariant Values (Possibly Indeterminate)**

A definition of one of the symbolic constants in the following list shall be omitted from <limits.h> on specific implementations where the corresponding value is equal to or greater than the stated minimum, but is unspecified.

This indetermination might depend on the amount of available memory space on a specific instance of a specific implementation. The actual value supported by a specific instance shall be provided by the *sysconf()* function.

{AIO\_LISTIO\_MAX}

Maximum number of I/O operations in a single list I/O call supported by the implementation.

Minimum Acceptable Value: {\_POSIX\_AIO\_LISTIO\_MAX}

9016 {AIO\_MAX}  
 9017 Maximum number of outstanding asynchronous I/O operations supported by the  
 9018 implementation.  
 9019 Minimum Acceptable Value: {\_POSIX\_AIO\_MAX}

9020 {AIO\_PRIO\_DELTA\_MAX}  
 9021 The maximum amount by which a process can decrease its asynchronous I/O priority level  
 9022 from its own scheduling priority.  
 9023 Minimum Acceptable Value: 0

9024 {ARG\_MAX}  
 9025 Maximum length of argument to the *exec* functions including environment data.  
 9026 Minimum Acceptable Value: {\_POSIX\_ARG\_MAX}

9027 {ATEXIT\_MAX}  
 9028 Maximum number of functions that may be registered with *atexit()*.  
 9029 Minimum Acceptable Value: 32

9030 {CHILD\_MAX}  
 9031 Maximum number of simultaneous processes per real user ID.  
 9032 Minimum Acceptable Value: {\_POSIX\_CHILD\_MAX}

9033 {DELAYTIMER\_MAX}  
 9034 Maximum number of timer expiration overruns.  
 9035 Minimum Acceptable Value: {\_POSIX\_DELAYTIMER\_MAX}

9036 {HOST\_NAME\_MAX}  
 9037 Maximum length of a host name (not including the terminating null) as returned from the  
 9038 *gethostname()* function.  
 9039 Minimum Acceptable Value: {\_POSIX\_HOST\_NAME\_MAX}

9040 XSI {IOV\_MAX}  
 9041 Maximum number of **iovec** structures that one process has available for use with *readv()* or  
 9042 *writev()*.  
 9043 Minimum Acceptable Value: {\_XOPEN\_IOV\_MAX}

9044 {LOGIN\_NAME\_MAX}  
 9045 Maximum length of a login name.  
 9046 Minimum Acceptable Value: {\_POSIX\_LOGIN\_NAME\_MAX}

9047 MSG {MQ\_OPEN\_MAX}  
 9048 The maximum number of open message queue descriptors a process may hold.  
 9049 Minimum Acceptable Value: {\_POSIX\_MQ\_OPEN\_MAX}

9050 MSG {MQ\_PRIO\_MAX}  
 9051 The maximum number of message priorities supported by the implementation.  
 9052 Minimum Acceptable Value: {\_POSIX\_MQ\_PRIO\_MAX}

9053 {OPEN\_MAX}  
 9054 A value one greater than the maximum value that the system may assign to a newly-created  
 9055 file descriptor.  
 9056 Minimum Acceptable Value: {\_POSIX\_OPEN\_MAX}

9057 {PAGESIZE}  
 9058 Size in bytes of a page.  
 9059 Minimum Acceptable Value: 1

9060 XSI {PAGE\_SIZE}  
 9061 Equivalent to {PAGESIZE}. If either {PAGESIZE} or {PAGE\_SIZE} is defined, the other is  
 9062 defined with the same value.

9063 {PTHREAD\_DESTRUCTOR\_ITERATIONS}  
 9064 Maximum number of attempts made to destroy a thread's thread-specific data values on  
 9065 thread exit.  
 9066 Minimum Acceptable Value: {\_POSIX\_THREAD\_DESTRUCTOR\_ITERATIONS}

9067 {PTHREAD\_KEYS\_MAX}  
 9068 Maximum number of data keys that can be created by a process.  
 9069 Minimum Acceptable Value: {\_POSIX\_THREAD\_KEYS\_MAX}

9070 {PTHREAD\_STACK\_MIN}  
 9071 Minimum size in bytes of thread stack storage.  
 9072 Minimum Acceptable Value: 0

9073 {PTHREAD\_THREADS\_MAX}  
 9074 Maximum number of threads that can be created per process.  
 9075 Minimum Acceptable Value: {\_POSIX\_THREAD\_THREADS\_MAX}

9076 {RTSIG\_MAX}  
 9077 Maximum number of realtime signals reserved for application use in this implementation.  
 9078 Minimum Acceptable Value: {\_POSIX\_RTSIG\_MAX}

9079 {SEM\_NSEMS\_MAX}  
 9080 Maximum number of semaphores that a process may have.  
 9081 Minimum Acceptable Value: {\_POSIX\_SEM\_NSEMS\_MAX}

9082 {SEM\_VALUE\_MAX}  
 9083 The maximum value a semaphore may have.  
 9084 Minimum Acceptable Value: {\_POSIX\_SEM\_VALUE\_MAX}

9085 {SIGQUEUE\_MAX}  
 9086 Maximum number of queued signals that a process may send and have pending at the  
 9087 receiver(s) at any time.  
 9088 Minimum Acceptable Value: {\_POSIX\_SIGQUEUE\_MAX}

9089 SS | TSP {SS\_REPL\_MAX}  
 9090 The maximum number of replenishment operations that may be simultaneously pending  
 9091 for a particular sporadic server scheduler.  
 9092 Minimum Acceptable Value: {\_POSIX\_SS\_REPL\_MAX}

9093 {STREAM\_MAX}  
 9094 Maximum number of streams that one process can have open at one time. If defined, it has  
 9095 the same value as {FOPEN\_MAX} (see <stdio.h>).  
 9096 Minimum Acceptable Value: {\_POSIX\_STREAM\_MAX}

9097 {SYMLOOP\_MAX}  
 9098 Maximum number of symbolic links that can be reliably traversed in the resolution of a  
 9099 pathname in the absence of a loop.  
 9100 Minimum Acceptable Value: {\_POSIX\_SYMLOOP\_MAX}

9101 {TIMER\_MAX}  
 9102 Maximum number of timers per process supported by the implementation.  
 9103 Minimum Acceptable Value: {\_POSIX\_TIMER\_MAX}

9104 OB TRC {TRACE\_EVENT\_NAME\_MAX}  
 9105 Maximum length of the trace event name (not including the terminating null).  
 9106 Minimum Acceptable Value: {\_POSIX\_TRACE\_EVENT\_NAME\_MAX}

9107 OB TRC {TRACE\_NAME\_MAX}  
 9108 Maximum length of the trace generation version string or of the trace stream name (not  
 9109 including the terminating null).  
 9110 Minimum Acceptable Value: {\_POSIX\_TRACE\_NAME\_MAX}

9111 OB TRC {TRACE\_SYS\_MAX}  
 9112 Maximum number of trace streams that may simultaneously exist in the system.  
 9113 Minimum Acceptable Value: {\_POSIX\_TRACE\_SYS\_MAX}

9114 OB TRC {TRACE\_USER\_EVENT\_MAX}  
 9115 Maximum number of user trace event type identifiers that may simultaneously exist in a  
 9116 traced process, including the predefined user trace event  
 9117 POSIX\_TRACE\_UNNAMED\_USER\_EVENT.  
 9118 Minimum Acceptable Value: {\_POSIX\_TRACE\_USER\_EVENT\_MAX}

9119 {TTY\_NAME\_MAX}  
 9120 Maximum length of terminal device name.  
 9121 Minimum Acceptable Value: {\_POSIX\_TTY\_NAME\_MAX}

9122 {TZNAME\_MAX}  
 9123 Maximum number of bytes supported for the name of a timezone (not of the TZ variable).  
 9124 Minimum Acceptable Value: {\_POSIX\_TZNAME\_MAX}

9125 **Note:** The length given by {TZNAME\_MAX} does not include the quoting characters mentioned in  
 9126 [Section 8.3](#) (on page 177).

### 9127 Pathname Variable Values

9128 The values in the following list may be constants within an implementation or may vary from  
 9129 one pathname to another. For example, file systems or directories may have different  
 9130 characteristics.

9131 A definition of one of the symbolic constants in the following list shall be omitted from the  
 9132 <limits.h> header on specific implementations where the corresponding value is equal to or  
 9133 greater than the stated minimum, but where the value can vary depending on the file to which it  
 9134 is applied. The actual value supported for a specific pathname shall be provided by the  
 9135 *pathconf()* function.

9136 {FILESIZEBITS}  
 9137 Minimum number of bits needed to represent, as a signed integer value, the maximum size  
 9138 of a regular file allowed in the specified directory.  
 9139 Minimum Acceptable Value: 32

9140 {LINK\_MAX}  
 9141 Maximum number of links to a single file.  
 9142 Minimum Acceptable Value: {\_POSIX\_LINK\_MAX}

9143 {MAX\_CANON}  
 9144 Maximum number of bytes in a terminal canonical input line.  
 9145 Minimum Acceptable Value: {\_POSIX\_MAX\_CANON}

9146 {MAX\_INPUT}  
 9147 Minimum number of bytes for which space is available in a terminal input queue; therefore,  
 9148 the maximum number of bytes a conforming application may require to be typed as input

9149 before reading them.  
 9150 Minimum Acceptable Value: {\_POSIX\_MAX\_INPUT}

9151 {NAME\_MAX}  
 9152 Maximum number of bytes in a filename (not including the terminating null of a filename  
 9153 string).  
 9154 Minimum Acceptable Value: {\_POSIX\_NAME\_MAX}

9155 XSI Minimum Acceptable Value: {\_XOPEN\_NAME\_MAX}

9156 {PATH\_MAX}  
 9157 Maximum number of bytes the implementation will store as a pathname in a user-supplied  
 9158 buffer of unspecified size, including the terminating null character. Minimum number the  
 9159 implementation will accept as the maximum number of bytes in a pathname.  
 9160 Minimum Acceptable Value: {\_POSIX\_PATH\_MAX}

9161 XSI Minimum Acceptable Value: {\_XOPEN\_PATH\_MAX}

9162 {PIPE\_BUF}  
 9163 Maximum number of bytes that is guaranteed to be atomic when writing to a pipe.  
 9164 Minimum Acceptable Value: {\_POSIX\_PIPE\_BUF}

9165 ADV {POSIX\_ALLOC\_SIZE\_MIN}  
 9166 Minimum number of bytes of storage actually allocated for any portion of a file.  
 9167 Minimum Acceptable Value: Not specified.

9168 ADV {POSIX\_REC\_INCR\_XFER\_SIZE}  
 9169 Recommended increment for file transfer sizes between the  
 9170 {POSIX\_REC\_MIN\_XFER\_SIZE} and {POSIX\_REC\_MAX\_XFER\_SIZE} values.  
 9171 Minimum Acceptable Value: Not specified.

9172 ADV {POSIX\_REC\_MAX\_XFER\_SIZE}  
 9173 Maximum recommended file transfer size.  
 9174 Minimum Acceptable Value: Not specified.

9175 ADV {POSIX\_REC\_MIN\_XFER\_SIZE}  
 9176 Minimum recommended file transfer size.  
 9177 Minimum Acceptable Value: Not specified.

9178 ADV {POSIX\_REC\_XFER\_ALIGN}  
 9179 Recommended file transfer buffer alignment.  
 9180 Minimum Acceptable Value: Not specified.

9181 {SYMLINK\_MAX}  
 9182 Maximum number of bytes in a symbolic link.  
 9183 Minimum Acceptable Value: {\_POSIX\_SYMLINK\_MAX}

### 9184 Runtime Increaseable Values

9185 The magnitude limitations in the following list shall be fixed by specific implementations. An  
 9186 application should assume that the value of the symbolic constant defined by <limits.h> in a  
 9187 specific implementation is the minimum that pertains whenever the application is run under  
 9188 that implementation. A specific instance of a specific implementation may increase the value  
 9189 relative to that supplied by <limits.h> for that implementation. The actual value supported by a  
 9190 specific instance shall be provided by the *sysconf*( ) function.

9191 {BC\_BASE\_MAX}  
 9192 Maximum *obase* values allowed by the *bc* utility.  
 9193 Minimum Acceptable Value: {\_POSIX2\_BC\_BASE\_MAX}

9194 {BC\_DIM\_MAX}  
 9195 Maximum number of elements permitted in an array by the *bc* utility.  
 9196 Minimum Acceptable Value: {\_POSIX2\_BC\_DIM\_MAX}

9197 {BC\_SCALE\_MAX}  
 9198 Maximum *scale* value allowed by the *bc* utility.  
 9199 Minimum Acceptable Value: {\_POSIX2\_BC\_SCALE\_MAX}

9200 {BC\_STRING\_MAX}  
 9201 Maximum length of a string constant accepted by the *bc* utility.  
 9202 Minimum Acceptable Value: {\_POSIX2\_BC\_STRING\_MAX}

9203 {CHARCLASS\_NAME\_MAX}  
 9204 Maximum number of bytes in a character class name.  
 9205 Minimum Acceptable Value: {\_POSIX2\_CHARCLASS\_NAME\_MAX}

9206 {COLL\_WEIGHTS\_MAX}  
 9207 Maximum number of weights that can be assigned to an entry of the *LC\_COLLATE* **order**  
 9208 keyword in the locale definition file; see [Chapter 7](#) (on page 135).  
 9209 Minimum Acceptable Value: {\_POSIX2\_COLL\_WEIGHTS\_MAX}

9210 {EXPR\_NEST\_MAX}  
 9211 Maximum number of expressions that can be nested within parentheses by the *expr* utility.  
 9212 Minimum Acceptable Value: {\_POSIX2\_EXPR\_NEST\_MAX}

9213 {LINE\_MAX}  
 9214 Unless otherwise noted, the maximum length, in bytes, of a utility's input line (either  
 9215 standard input or another file), when the utility is described as processing text files. The  
 9216 length includes room for the trailing <newline>.  
 9217 Minimum Acceptable Value: {\_POSIX2\_LINE\_MAX}

9218 {NGROUPS\_MAX}  
 9219 Maximum number of simultaneous supplementary group IDs per process.  
 9220 Minimum Acceptable Value: {\_POSIX\_NGROUPS\_MAX}

9221 {RE\_DUP\_MAX}  
 9222 Maximum number of repeated occurrences of a BRE or ERE interval expression; see [Section 9.3.6](#) (on page 186) and [Section 9.4.6](#) (on page 190).  
 9223 [Section 9.3.6](#) (on page 186) and [Section 9.4.6](#) (on page 190).  
 9224 Minimum Acceptable Value: {\_POSIX\_RE\_DUP\_MAX}

## Maximum Values

The <limits.h> header shall define the following symbolic constants with the values shown. These are the most restrictive values for certain features on an implementation. A conforming implementation shall provide values no larger than these values. A conforming application must not require a smaller value for correct operation.

9230 {\_POSIX\_CLOCKRES\_MIN}  
 9231 The resolution of the CLOCK\_REALTIME clock, in nanoseconds.  
 9232 Value: 20 000 000

9233 MON If the Monotonic Clock option is supported, the resolution of the CLOCK\_MONOTONIC  
 9234 clock, in nanoseconds, is represented by {\_POSIX\_CLOCKRES\_MIN}.



**Minimum Values**

The <limits.h> header shall define the following symbolic constants with the values shown. These are the most restrictive values for certain features on an implementation conforming to this volume of POSIX.1-2008. Related symbolic constants are defined elsewhere in this volume of POSIX.1-2008 which reflect the actual implementation and which need not be as restrictive. For each of these limits, a conforming implementation shall provide a value at least this large or shall have no limit. A strictly conforming application must not require a larger value for correct operation.

{\_POSIX\_AIO\_LISTIO\_MAX}

The number of I/O operations that can be specified in a list I/O call.

Value: 2

{\_POSIX\_AIO\_MAX}

The number of outstanding asynchronous I/O operations.

Value: 1

{\_POSIX\_ARG\_MAX}

Maximum length of argument to the *exec* functions including environment data.

Value: 4 096

{\_POSIX\_CHILD\_MAX}

Maximum number of simultaneous processes per real user ID.

Value: 25

{\_POSIX\_DELAYTIMER\_MAX}

The number of timer expiration overruns.

Value: 32

{\_POSIX\_HOST\_NAME\_MAX}

Maximum length of a host name (not including the terminating null) as returned from the *gethostname()* function.

Value: 255

{\_POSIX\_LINK\_MAX}

Maximum number of links to a single file.

Value: 8

{\_POSIX\_LOGIN\_NAME\_MAX}

The size of the storage required for a login name, in bytes (including the terminating null).

Value: 9

{\_POSIX\_MAX\_CANON}

Maximum number of bytes in a terminal canonical input queue.

Value: 255

{\_POSIX\_MAX\_INPUT}

Maximum number of bytes allowed in a terminal input queue.

Value: 255

{\_POSIX\_MQ\_OPEN\_MAX}

The number of message queues that can be open for a single process.

Value: 8

{\_POSIX\_MQ\_PRIO\_MAX}

The maximum number of message priorities supported by the implementation.

Value: 32



9280        { \_POSIX\_NAME\_MAX}  
 9281           Maximum number of bytes in a filename (not including the terminating null of a filename  
 9282           string).  
 9283           Value: 14

9284        { \_POSIX\_NGROUPS\_MAX}  
 9285           Maximum number of simultaneous supplementary group IDs per process.  
 9286           Value: 8

9287        { \_POSIX\_OPEN\_MAX}  
 9288           A value one greater than the maximum value that the system may assign to a newly-created  
 9289           file descriptor.  
 9290           Value: 20

9291        { \_POSIX\_PATH\_MAX}  
 9292           Minimum number the implementation will accept as the maximum number of bytes in a  
 9293           pathname.  
 9294           Value: 256

9295        { \_POSIX\_PIPE\_BUF}  
 9296           Maximum number of bytes that is guaranteed to be atomic when writing to a pipe.  
 9297           Value: 512

9298        { \_POSIX\_RE\_DUP\_MAX}  
 9299           Maximum number of repeated occurrences of a BRE or ERE interval expression; see [Section 9.3.6](#) (on page 186) and [Section 9.4.6](#) (on page 190).  
 9300           Value: 255

9301        { \_POSIX\_RTSIG\_MAX}  
 9302           The number of realtime signal numbers reserved for application use.  
 9303           Value: 8

9304        { \_POSIX\_SEM\_NSEMS\_MAX}  
 9305           The number of semaphores that a process may have.  
 9306           Value: 256

9307        { \_POSIX\_SEM\_VALUE\_MAX}  
 9308           The maximum value a semaphore may have.  
 9309           Value: 32 767

9310        { \_POSIX\_SIGQUEUE\_MAX}  
 9311           The number of queued signals that a process may send and have pending at the receiver(s)  
 9312           at any time.  
 9313           Value: 32

9314        { \_POSIX\_SSIZE\_MAX}  
 9315           The value that can be stored in an object of type **ssize\_t**.  
 9316           Value: 32 767

9317        { \_POSIX\_SS\_REPL\_MAX}  
 9318           The number of replenishment operations that may be simultaneously pending for a  
 9319           particular sporadic server scheduler.  
 9320           Value: 4

9321        { \_POSIX\_STREAM\_MAX}  
 9322           The number of streams that one process can have open at one time.  
 9323           Value: 8

9324

9325           { \_POSIX\_SYMLINK\_MAX}  
 9326           The number of bytes in a symbolic link.  
 9327           Value: 255

9328           { \_POSIX\_SYMLOOP\_MAX}  
 9329           The number of symbolic links that can be traversed in the resolution of a pathname in the  
 9330           absence of a loop.  
 9331           Value: 8

9332           { \_POSIX\_THREAD\_DESTRUCTOR\_ITERATIONS}  
 9333           The number of attempts made to destroy a thread's thread-specific data values on thread  
 9334           exit.  
 9335           Value: 4

9336           { \_POSIX\_THREAD\_KEYS\_MAX}  
 9337           The number of data keys per process.  
 9338           Value: 128

9339           { \_POSIX\_THREAD\_THREADS\_MAX}  
 9340           The number of threads per process.  
 9341           Value: 64

9342           { \_POSIX\_TIMER\_MAX}  
 9343           The per-process number of timers.  
 9344           Value: 32

9345 OB TRC { \_POSIX\_TRACE\_EVENT\_NAME\_MAX}  
 9346           The length in bytes of a trace event name (not including the terminating null).  
 9347           Value: 30

9348 OB TRC { \_POSIX\_TRACE\_NAME\_MAX}  
 9349           The length in bytes of a trace generation version string or a trace stream name (not  
 9350           including the terminating null).  
 9351           Value: 8

9352 OB TRC { \_POSIX\_TRACE\_SYS\_MAX}  
 9353           The number of trace streams that may simultaneously exist in the system.  
 9354           Value: 8

9355 OB TRC { \_POSIX\_TRACE\_USER\_EVENT\_MAX}  
 9356           The number of user trace event type identifiers that may simultaneously exist in a traced  
 9357           process, including the predefined user trace event  
 9358           \_POSIX\_TRACE\_UNNAMED\_USER\_EVENT.  
 9359           Value: 32

9360           { \_POSIX\_TTY\_NAME\_MAX}  
 9361           The size of the storage required for a terminal device name, in bytes (including the  
 9362           terminating null).  
 9363           Value: 9

9364           { \_POSIX\_TZNAME\_MAX}  
 9365           Maximum number of bytes supported for the name of a timezone (not of the TZ variable).  
 9366           Value: 6

9367           **Note:**   The length given by { \_POSIX\_TZNAME\_MAX} does not include the quoting characters  
 9368           mentioned in [Section 8.3](#) (on page 177).

9369		{_POSIX2_BC_BASE_MAX}
9370		Maximum <i>obase</i> values allowed by the <i>bc</i> utility.
9371		Value: 99
9372		{_POSIX2_BC_DIM_MAX}
9373		Maximum number of elements permitted in an array by the <i>bc</i> utility.
9374		Value: 2 048
9375		{_POSIX2_BC_SCALE_MAX}
9376		Maximum <i>scale</i> value allowed by the <i>bc</i> utility.
9377		Value: 99
9378		{_POSIX2_BC_STRING_MAX}
9379		Maximum length of a string constant accepted by the <i>bc</i> utility.
9380		Value: 1 000
9381		{_POSIX2_CHARCLASS_NAME_MAX}
9382		Maximum number of bytes in a character class name.
9383		Value: 14
9384		{_POSIX2_COLL_WEIGHTS_MAX}
9385		Maximum number of weights that can be assigned to an entry of the <i>LC_COLLATE</i> <b>order</b>
9386		keyword in the locale definition file; see <a href="#">Chapter 7</a> (on page 135).
9387		Value: 2
9388		{_POSIX2_EXPR_NEST_MAX}
9389		Maximum number of expressions that can be nested within parentheses by the <i>expr</i> utility.
9390		Value: 32
9391		{_POSIX2_LINE_MAX}
9392		Unless otherwise noted, the maximum length, in bytes, of a utility's input line (either
9393		standard input or another file), when the utility is described as processing text files. The
9394		length includes room for the trailing <newline>.
9395		Value: 2 048
9396		{_POSIX2_RE_DUP_MAX}
9397		Maximum number of repeated occurrences of a BRE or ERE interval expression; see <a href="#">Section 9.3.6</a> (on page 186) and <a href="#">Section 9.4.6</a> (on page 190).
9398		Value: 255
9399		
9400	XSI	{_XOPEN_IOV_MAX}
9401		Maximum number of <b>iovec</b> structures that one process has available for use with <i>readv()</i> or
9402		<i>writev()</i> .
9403		Value: 16
9404	XSI	{_XOPEN_NAME_MAX}
9405		Maximum number of bytes in a filename (not including the terminating null of a filename
9406		string).
9407		Value: 255
9408	XSI	{_XOPEN_PATH_MAX}
9409		Minimum number the implementation will accept as the maximum number of bytes in a
9410		pathname.
9411		Value: 1 024

**Numerical Limits**

The <limits.h> header shall define the following macros and, except for {CHAR\_BIT}, {LONG\_BIT}, {MB\_LEN\_MAX}, and {WORD\_BIT}, they shall be replaced by expressions that have the same type as would an expression that is an object of the corresponding type converted according to the integer promotions.

If the value of an object of type **char** is treated as a signed integer when used in an expression, the value of {CHAR\_MIN} is the same as that of {SCHAR\_MIN} and the value of {CHAR\_MAX} is the same as that of {SCHAR\_MAX}. Otherwise, the value of {CHAR\_MIN} is 0 and the value of {CHAR\_MAX} is the same as that of {UCHAR\_MAX}.

{CHAR\_BIT}

Number of bits in a type **char**.

Value: 8

{CHAR\_MAX}

Maximum value for an object of type **char**.

Value: {UCHAR\_MAX} or {SCHAR\_MAX}

{CHAR\_MIN}

Minimum value for an object of type **char**.

Value: {SCHAR\_MIN} or 0

{INT\_MAX}

Maximum value for an object of type **int**.

Minimum Acceptable Value: 2 147 483 647

{INT\_MIN}

Minimum value for an object of type **int**.

Maximum Acceptable Value: -2 147 483 647

{LLONG\_MAX}

Maximum value for an object of type **long long**.

Minimum Acceptable Value: +9 223 372 036 854 775 807

{LLONG\_MIN}

Minimum value for an object of type **long long**.

Maximum Acceptable Value: -9 223 372 036 854 775 807

{LONG\_BIT}

Number of bits in an object of type **long**.

Minimum Acceptable Value: 32

{LONG\_MAX}

Maximum value for an object of type **long**.

Minimum Acceptable Value: +2 147 483 647

{LONG\_MIN}

Minimum value for an object of type **long**.

Maximum Acceptable Value: -2 147 483 647

{MB\_LEN\_MAX}

Maximum number of bytes in a character, for any supported locale.

Minimum Acceptable Value: 1

{SCHAR\_MAX}

Maximum value for an object of type **signed char**.

9456	CX	Value: +127
9457		{SCHAR_MIN}
9458		Minimum value for an object of type <b>signed char</b> .
9459	CX	Value: -128
9460		{SHRT_MAX}
9461		Maximum value for an object of type <b>short</b> .
9462		Minimum Acceptable Value: +32 767
9463		{SHRT_MIN}
9464		Minimum value for an object of type <b>short</b> .
9465		Maximum Acceptable Value: -32 767
9466	CX	{SSIZE_MAX}
9467		Maximum value for an object of type <b>ssize_t</b> .
9468		Minimum Acceptable Value: {_POSIX_SSIZE_MAX}
9469		{UCHAR_MAX}
9470		Maximum value for an object of type <b>unsigned char</b> .
9471	CX	Value: 255
9472		{UINT_MAX}
9473		Maximum value for an object of type <b>unsigned</b> .
9474	CX	Minimum Acceptable Value: 4 294 967 295
9475		{ULLONG_MAX}
9476		Maximum value for an object of type <b>unsigned long long</b> .
9477		Minimum Acceptable Value: 18 446 744 073 709 551 615
9478		{ULONG_MAX}
9479		Maximum value for an object of type <b>unsigned long</b> .
9480		Minimum Acceptable Value: 4 294 967 295
9481		{USHRT_MAX}
9482		Maximum value for an object of type <b>unsigned short</b> .
9483		Minimum Acceptable Value: 65 535
9484	CX	{WORD_BIT}
9485		Number of bits in an object of type <b>int</b> .
9486		Minimum Acceptable Value: 32
9487		<b>Other Invariant Values</b>
9488		The <limits.h> header shall define the following symbolic constants:
9489		{NL_ARGMAX}
9490		Maximum value of <i>n</i> in conversion specifications using the "%n\$" sequence in calls to the <i>printf()</i> and <i>scanf()</i> families of functions.
9491		Minimum Acceptable Value: 9
9492		
9493	XSI	{NL_LANGMAX}
9494		Maximum number of bytes in a <i>LANG</i> name.
9495		Minimum Acceptable Value: 14
9496		{NL_MSGMAX}
9497		Maximum message number.
9498		Minimum Acceptable Value: 32 767

9499 {NL\_SETMAX}  
9500 Maximum set number.  
9501 Minimum Acceptable Value: 255

9502 {NL\_TEXTMAX}  
9503 Maximum number of bytes in a message string.  
9504 Minimum Acceptable Value: {\_POSIX2\_LINE\_MAX}

9505 XSI {NZERO}  
9506 Default process priority.  
9507 Minimum Acceptable Value: 20

## APPLICATION USAGE

None.

## RATIONALE

A request was made to reduce the value of {\_POSIX\_LINK\_MAX} from the value of 8 specified for it in the POSIX.1-1990 standard to 2. The standard developers decided to deny this request for several reasons:

- They wanted to avoid making any changes to the standard that could break conforming applications, and the requested change could have that effect.
- The use of multiple hard links to a file cannot always be replaced with use of symbolic links. Symbolic links are semantically different from hard links in that they associate a pathname with another pathname rather than a pathname with a file. This has implications for access control, file permanence, and transparency.
- The original standard developers had considered the issue of allowing for implementations that did not in general support hard links, and decided that this would reduce consensus on the standard.

Systems that support historical versions of the development option of the ISO POSIX-2 standard retain the name {\_POSIX2\_RE\_DUP\_MAX} as an alias for {\_POSIX\_RE\_DUP\_MAX}.

{PATH\_MAX}  
IEEE PASC Interpretation 1003.1 #15 addressed the inconsistency in the standard with the definition of pathname and the description of {PATH\_MAX}, allowing application developers to allocate either {PATH\_MAX} or {PATH\_MAX}+1 bytes. The inconsistency has been removed by correction to the {PATH\_MAX} definition to include the null character. With this change, applications that previously allocated {PATH\_MAX} bytes will continue to succeed.

{SYMLINK\_MAX}  
This symbol refers to space for data that is stored in the file system, as opposed to {PATH\_MAX} which is the length of a name that can be passed to a function. In some existing implementations, the pathnames pointed to by symbolic links are stored in the *inodes* of the links, so it is important that {SYMLINK\_MAX} not be constrained to be as large as {PATH\_MAX}.

## FUTURE DIRECTIONS

None.

## SEE ALSO

Chapter 7 (on page 135), <stdio.h>, <unistd.h>  
XSH Section 2.2 (on page 472), *fpathconf()*, *sysconf()*

## CHANGE HISTORY

First released in Issue 1.

### Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

{FILESIZEBITS} is added for the Large File Summit extensions.

The minimum acceptable values for {INT\_MAX}, {INT\_MIN}, and {UINT\_MAX} are changed to make 32-bit values the minimum requirement.

The entry is restructured to improve readability.

### Issue 6

The Open Group Corrigendum U033/4 is applied. The wording is made clear for {CHAR\_MIN}, {INT\_MIN}, {LONG\_MIN}, {SCHAR\_MIN}, and {SHRT\_MIN} that these are maximum acceptable values.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The minimum value for {CHILD\_MAX} is 25. This is a FIPS requirement.
- The minimum value for {OPEN\_MAX} is 20. This is a FIPS requirement.
- The minimum value for {NGROUPS\_MAX} is 8. This is also a FIPS requirement.

Symbolic constants are added for {\_POSIX\_SYMLINK\_MAX}, {\_POSIX\_SYMLOOP\_MAX}, {\_POSIX\_RE\_DUP\_MAX}, {RE\_DUP\_MAX}, {SYMLOOP\_MAX}, and {SYMLINK\_MAX}.

The following values are added for alignment with IEEE Std 1003.1d-1999:

```
{_POSIX_SS_REPL_MAX}
{SS_REPL_MAX}
{POSIX_ALLOC_SIZE_MIN}
{POSIX_REC_INCR_XFER_SIZE}
{POSIX_REC_MAX_XFER_SIZE}
{POSIX_REC_MIN_XFER_SIZE}
{POSIX_REC_XFER_ALIGN}
```

Reference to CLOCK\_MONOTONIC is added in the description of {\_POSIX\_CLOCKRES\_MIN} for alignment with IEEE Std 1003.1j-2000.

The constants {LLONG\_MIN}, {LLONG\_MAX}, and {ULLONG\_MAX} are added for alignment with the ISO/IEC 9899:1999 standard.

The following values are added for alignment with IEEE Std 1003.1q-2000:

```
{_POSIX_TRACE_EVENT_NAME_MAX}
{_POSIX_TRACE_NAME_MAX}
{_POSIX_TRACE_SYS_MAX}
{_POSIX_TRACE_USER_EVENT_MAX}
{TRACE_EVENT_NAME_MAX}
{TRACE_NAME_MAX}
{TRACE_SYS_MAX}
{TRACE_USER_EVENT_MAX}
```

The new limits {\_XOPEN\_NAME\_MAX} and {\_XOPEN\_PATH\_MAX} are added as minimum



values for {PATH\_MAX} and {NAME\_MAX} limits on XSI-conformant systems.

The LEGACY symbols {PASS\_MAX} and {TMP\_MAX} are removed.

The values for the limits {CHAR\_BIT}, {SCHAR\_MAX}, and {UCHAR\_MAX} are now required to be 8, +127, and 255, respectively.

The value for the limit {CHAR\_MAX} is now {UCHAR\_MAX} or {SCHAR\_MAX}.

The value for the limit {CHAR\_MIN} is now {SCHAR\_MIN} or zero.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/10 is applied, correcting the value of {\_POSIX\_CHILD\_MAX} from 6 to 25. This is for FIPS 151-2 alignment.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/19 is applied, updating the values for {INT\_MAX}, {UINT\_MAX}, and {INT\_MIN} to be CX extensions over the ISO C standard, and correcting {WORD\_BIT} from 16 to 32.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/20 is applied, removing {CHARCLASS\_NAME\_MAX} from the “Other Invariant Values” section (it also occurs under “Runtime Increaseable Values”).

#### Issue 7

Austin Group Interpretations 1003.1-2001 #143 and #160 are applied.

Austin Group Interpretation 1003.1-2001 #173 is applied, updating the descriptions of {TRACE\_EVENT\_NAME\_MAX} and {TRACE\_NAME\_MAX} to not include the terminating null.

SD5-XBD-ERN-36 is applied, changing the description of {RE\_DUP\_MAX}.

SD5-XBD-ERN-90 is applied.

{NL\_NMAX} is removed; it should have been removed in Issue 6.

The Trace option values are marked obsolescent.

The {ATEXIT\_MAX}, {LONG\_BIT}, {NL\_MSGMAX}, {NL\_SETMAX}, {NL\_TEXTMAX}, and {WORD\_BIT} values are moved from the XSI option to the Base.

The AIO\_\* and \_POSIX\_AIO\_\* values are moved from the Asynchronous Input and Output option to the Base.

The {\_POSIX\_RTSIG\_MAX}, {\_POSIX\_SIGQUEUE\_MAX}, {RTSIG\_MAX}, and {SIGQUEUE\_MAX} values are moved from the Realtime Signals Extension option to the Base.

Functionality relating to the Threads and Timers options is moved to the Base.

This reference page is clarified with respect to macros and symbolic constants.

POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0052 [108], XBD/TC1-2008/0053 [291], XBD/TC1-2008/0054 [182,427], XBD/TC1-2008/0055 [291], XBD/TC1-2008/0056 [371], XBD/TC1-2008/0057 [291], XBD/TC1-2008/0058 [108], and XBD/TC1-2008/0059 [291] are applied.



**NAME**

locale.h — category macros

**SYNOPSIS**

#include &lt;locale.h&gt;

**DESCRIPTION**

Some of the functionality described on this reference page extends the ISO C standard. Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to enable the visibility of these symbols in this header.

The <locale.h> header shall define the **lconv** structure, which shall include at least the following members. (See the definitions of *LC\_MONETARY* in [Section 7.3.3](#) (on page 155) and [Section 7.3.4](#) (on page 158).)

```
char    *currency_symbol
char    *decimal_point
char    frac_digits
char    *grouping
char    *int_curr_symbol
char    int_frac_digits
char    int_n_cs_precedes
char    int_n_sep_by_space
char    int_n_sign_posn
char    int_p_cs_precedes
char    int_p_sep_by_space
char    int_p_sign_posn
char    *mon_decimal_point
char    *mon_grouping
char    *mon_thousands_sep
char    *negative_sign
char    n_cs_precedes
char    n_sep_by_space
char    n_sign_posn
char    *positive_sign
char    p_cs_precedes
char    p_sep_by_space
char    p_sign_posn
char    *thousands_sep
```

The <locale.h> header shall define NULL (as described in <stddef.h>) and at least the following as macros:

```
LC_ALL
LC_COLLATE
LC_CTYPE
LC_MESSAGES
LC_MONETARY
LC_NUMERIC
LC_TIME
```

which shall expand to integer constant expressions with distinct values for use as the first argument to the *setlocale()* function.

Additional macro definitions, beginning with the characters *LC\_* and an uppercase letter, may also be specified by the implementation.

9668 CX The <locale.h> header shall contain at least the following macros representing bitmasks for use  
 9669 with the *newlocale()* function for each supported locale category:

9670 LC\_COLLATE\_MASK  
 9671 LC\_CTYPE\_MASK  
 9672 LC\_MESSAGES\_MASK  
 9673 LC\_MONETARY\_MASK  
 9674 LC\_NUMERIC\_MASK  
 9675 LC\_TIME\_MASK

9676 In addition, a macro to set the bits for all categories set shall be defined:

9677 LC\_ALL\_MASK

9678 The <locale.h> header shall define LC\_GLOBAL\_LOCALE, a special locale object descriptor  
 9679 used by the *duplocale()* and *uselocale()* functions.

9680 The <locale.h> header shall define the **locale\_t** type, representing a locale object.

9681 The following shall be declared as functions and may also be defined as macros. Function  
 9682 prototypes shall be provided for use with ISO C standard compilers.

9683 CX locale\_t duplocale(locale\_t);  
 9684 void freelocale(locale\_t);  
 9685 struct lconv \*localeconv(void);

9686 CX locale\_t newlocale(int, const char \*, locale\_t);  
 9687 char \*setlocale(int, const char \*);

9688 CX locale\_t uselocale (locale\_t);

## 9689 APPLICATION USAGE

9690 None.

## 9691 RATIONALE

9692 It is suggested that each category macro name for use in *setlocale()* have a corresponding macro  
 9693 name ending in *\_MASK* for use in *newlocale()*.

## 9694 FUTURE DIRECTIONS

9695 None.

## 9696 SEE ALSO

9697 Chapter 8 (on page 173), <stddef.h>

9698 XSH *duplocale()*, *freelocale()*, *localeconv()*, *newlocale()*, *setlocale()*, *uselocale()*

## 9699 CHANGE HISTORY

9700 First released in Issue 3.

9701 Included for alignment with the ISO C standard.

## 9702 Issue 6

9703 The **lconv** structure is expanded with new members (**int\_n\_cs\_precedes**, **int\_n\_sep\_by\_space**,  
 9704 **int\_n\_sign\_posn**, **int\_p\_cs\_precedes**, **int\_p\_sep\_by\_space**, and **int\_p\_sign\_posn**) for alignment  
 9705 with the ISO/IEC 9899:1999 standard.

9706 Extensions beyond the ISO C standard are marked.

9707 **Issue 7**

9708 The *duplocale()*, *freelocale()*, *newlocale()*, and *uselocale()* functions are added from The Open  
9709 Group Technical Standard, 2006, Extended API Set Part 4.

9710 This reference page is clarified with respect to macros and symbolic constants.

9711 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0060 [301,427] is applied.

## 9712 NAME

9713 math.h — mathematical declarations

## 9714 SYNOPSIS

9715 #include &lt;math.h&gt;

## 9716 DESCRIPTION

9717 CX Some of the functionality described on this reference page extends the ISO C standard.  
 9718 Applications shall define the appropriate feature test macro (see XSH Section 2.2, on page 472) to  
 9719 enable the visibility of these symbols in this header.

9720 The &lt;math.h&gt; header shall define at least the following types:

9721 **float\_t** A real-floating type at least as wide as **float**.9722 **double\_t** A real-floating type at least as wide as **double**, and at least as wide as **float\_t**.

9723 If FLT\_EVAL\_METHOD equals 0, **float\_t** and **double\_t** shall be **float** and **double**, respectively; if  
 9724 FLT\_EVAL\_METHOD equals 1, they shall both be **double**; if FLT\_EVAL\_METHOD equals 2,  
 9725 they shall both be **long double**; for other values of FLT\_EVAL\_METHOD, they are otherwise  
 9726 implementation-defined.

9727 The <math.h> header shall define the following macros, where real-floating indicates that the  
 9728 argument shall be an expression of real-floating type:

```
9729 int fpclassify(real-floating x);
9730 int isfinite(real-floating x);
9731 int isgreater(real-floating x, real-floating y);
9732 int isgreaterequal(real-floating x, real-floating y);
9733 int isinf(real-floating x);
9734 int isless(real-floating x, real-floating y);
9735 int islessequal(real-floating x, real-floating y);
9736 int islessgreater(real-floating x, real-floating y);
9737 int isnan(real-floating x);
9738 int isnormal(real-floating x);
9739 int isunordered(real-floating x, real-floating y);
9740 int signbit(real-floating x);
```

9741 The <math.h> header shall define the following symbolic constants. The values shall have type  
 9742 **double** and shall be accurate to at least the precision of the **double** type.

9743	XSI	<b>M_E</b>	Value of $e$
9744	XSI	<b>M_LOG2E</b>	Value of $\log_2 e$
9745	XSI	<b>M_LOG10E</b>	Value of $\log_{10} e$
9746	XSI	<b>M_LN2</b>	Value of $\log_e 2$
9747	XSI	<b>M_LN10</b>	Value of $\log_e 10$
9748	XSI	<b>M_PI</b>	Value of $\pi$
9749	XSI	<b>M_PI_2</b>	Value of $\pi/2$
9750	XSI	<b>M_PI_4</b>	Value of $\pi/4$
9751	XSI	<b>M_1_PI</b>	Value of $1/\pi$
9752	XSI	<b>M_2_PI</b>	Value of $2/\pi$

9753	XSI	<b>M_2_SQRTPI</b>	Value of $2/\sqrt{\pi}$
9754	XSI	<b>M_SQRT2</b>	Value of $\sqrt{2}$
9755	XSI	<b>M_SQRT1_2</b>	Value of $1/\sqrt{2}$
9756		The <math.h> header shall define the following symbolic constant:	
9757	OB XSI	<b>MAXFLOAT</b>	Same value as FLT_MAX in <float.h>.
9758		The <math.h> header shall define the following macros:	
9759		<b>HUGE_VAL</b>	A positive <b>double</b> constant expression, not necessarily representable as a
9760			<b>float</b> . Used as an error value returned by the mathematics library.
9761			HUGE_VAL evaluates to +infinity on systems supporting IEEE Std 754-1985.
9762		<b>HUGE_VALF</b>	A positive <b>float</b> constant expression. Used as an error value returned by the
9763			mathematics library. HUGE_VALF evaluates to +infinity on systems
9764			supporting IEEE Std 754-1985.
9765		<b>HUGE_VALL</b>	A positive <b>long double</b> constant expression. Used as an error value returned
9766			by the mathematics library. HUGE_VALL evaluates to +infinity on systems
9767			supporting IEEE Std 754-1985.
9768		<b>INFINITY</b>	A constant expression of type <b>float</b> representing positive or unsigned infinity,
9769			if available; else a positive constant of type <b>float</b> that overflows at translation
9770			time.
9771		<b>NAN</b>	A constant expression of type <b>float</b> representing a quiet NaN. This macro is
9772			only defined if the implementation supports quiet NaNs for the <b>float</b> type.
9773		The following macros shall be defined for number classification. They represent the mutually-	
9774		exclusive kinds of floating-point values. They expand to integer constant expressions with	
9775		distinct values. Additional implementation-defined floating-point classifications, with macro	
9776		definitions beginning with FP_ and an uppercase letter, may also be specified by the	
9777		implementation.	
9778		<b>FP_INFINITE</b>	
9779		<b>FP_NAN</b>	
9780		<b>FP_NORMAL</b>	
9781		<b>FP_SUBNORMAL</b>	
9782		<b>FP_ZERO</b>	
9783		The following optional macros indicate whether the <i>fma()</i> family of functions are fast compared	
9784		with direct code:	
9785		<b>FP_FAST_FMA</b>	
9786		<b>FP_FAST_FMAF</b>	
9787		<b>FP_FAST_FMAL</b>	
9788		If defined, the FP_FAST_FMA macro shall expand to the integer constant 1 and shall indicate	
9789		that the <i>fma()</i> function generally executes about as fast as, or faster than, a multiply and an add	
9790		of <b>double</b> operands. If undefined, the speed of execution is unspecified. The other macros have	
9791		the equivalent meaning for the <b>float</b> and <b>long double</b> versions.	
9792		The following macros shall expand to integer constant expressions whose values are returned by	
9793		<i>ilogb(x)</i> if <i>x</i> is zero or NaN, respectively. The value of FP_ILOGB0 shall be either {INT_MIN} or	
9794		-{INT_MAX}. The value of FP_ILOGBNAN shall be either {INT_MAX} or {INT_MIN}.	

9795           FP\_ILOGB0  
 9796           FP\_ILOGBNAN

9797           The following macros shall expand to the integer constants 1 and 2, respectively;

9798           MATH\_ERRNO  
 9799           MATH\_ERREXCEPT

9800           The following macro shall expand to an expression that has type **int** and the value  
 9801           MATH\_ERRNO, MATH\_ERREXCEPT, or the bitwise-inclusive OR of both:

9802           math\_errhandling

9803           The value of math\_errhandling is constant for the duration of the program. It is unspecified  
 9804           whether math\_errhandling is a macro or an identifier with external linkage. If a macro definition  
 9805           is suppressed or a program defines an identifier with the name math\_errhandling, the behavior  
 9806           is undefined. If the expression (math\_errhandling & MATH\_ERREXCEPT) can be non-zero, the  
 9807           implementation shall define the macros FE\_DIVBYZERO, FE\_INVALID, and FE\_OVERFLOW in  
 9808           <fenv.h>.

9809           The following shall be declared as functions and may also be defined as macros. Function  
 9810           prototypes shall be provided.

9811           double       acos(double);  
 9812           float        acosf(float);  
 9813           double       acosh(double);  
 9814           float        acoshf(float);  
 9815           long double  acoshl(long double);  
 9816           long double  acosl(long double);  
 9817           double       asin(double);  
 9818           float        asinf(float);  
 9819           double       asinh(double);  
 9820           float        asinhf(float);  
 9821           long double  asinh1(long double);  
 9822           long double  asin1(long double);  
 9823           double       atan(double);  
 9824           double       atan2(double, double);  
 9825           float        atan2f(float, float);  
 9826           long double  atan2l(long double, long double);  
 9827           float        atanf(float);  
 9828           double       atanh(double);  
 9829           float        atanhf(float);  
 9830           long double  atanh1(long double);  
 9831           long double  atan1(long double);  
 9832           double       cbrt(double);  
 9833           float        cbrtf(float);  
 9834           long double  cbrt1(long double);  
 9835           double       ceil(double);  
 9836           float        ceilf(float);  
 9837           long double  ceill(long double);  
 9838           double       copysign(double, double);  
 9839           float        copysignf(float, float);  
 9840           long double  copysignl(long double, long double);  
 9841           double       cos(double);

```

9842     float      cosf(float);
9843     double     cosh(double);
9844     float      coshf(float);
9845     long double coshl(long double);
9846     long double cosl(long double);
9847     double     erf(double);
9848     double     erfc(double);
9849     float      erfcf(float);
9850     long double erfcl(long double);
9851     float      erff(float);
9852     long double erfl(long double);
9853     double     exp(double);
9854     double     exp2(double);
9855     float      exp2f(float);
9856     long double exp2l(long double);
9857     float      expf(float);
9858     long double expl(long double);
9859     double     expm1(double);
9860     float      expm1f(float);
9861     long double expm1l(long double);
9862     double     fabs(double);
9863     float      fabsf(float);
9864     long double fabsl(long double);
9865     double     fdim(double, double);
9866     float      fdimf(float, float);
9867     long double fdiml(long double, long double);
9868     double     floor(double);
9869     float      floorf(float);
9870     long double floorl(long double);
9871     double     fma(double, double, double);
9872     float      fmaf(float, float, float);
9873     long double fmal(long double, long double, long double);
9874     double     fmax(double, double);
9875     float      fmaxf(float, float);
9876     long double fmaxl(long double, long double);
9877     double     fmin(double, double);
9878     float      fminf(float, float);
9879     long double fminl(long double, long double);
9880     double     fmod(double, double);
9881     float      fmodf(float, float);
9882     long double fmodl(long double, long double);
9883     double     frexp(double, int *);
9884     float      frexpf(float, int *);
9885     long double frexpl(long double, int *);
9886     double     hypot(double, double);
9887     float      hypotf(float, float);
9888     long double hypotl(long double, long double);
9889     int        ilogb(double);
9890     int        ilogbf(float);
9891     int        ilogbl(long double);
9892     double     j0(double);
9893     double     j1(double);

```

```

9894 double      jn(int, double);
9895 double      ldexp(double, int);
9896 float       ldexpf(float, int);
9897 long double ldexpl(long double, int);
9898 double      lgamma(double);
9899 float       lgammaf(float);
9900 long double lgammal(long double);
9901 long long    llrint(double);
9902 long long    llrintf(float);
9903 long long    llrintl(long double);
9904 long long    llround(double);
9905 long long    llroundf(float);
9906 long long    llroundl(long double);
9907 double      log(double);
9908 double      log10(double);
9909 float       log10f(float);
9910 long double log10l(long double);
9911 double      log1p(double);
9912 float       log1pf(float);
9913 long double log1pl(long double);
9914 double      log2(double);
9915 float       log2f(float);
9916 long double log2l(long double);
9917 double      logb(double);
9918 float       logbf(float);
9919 long double logbl(long double);
9920 float       logf(float);
9921 long double logl(long double);
9922 long        lrint(double);
9923 long        lrintf(float);
9924 long        lrintl(long double);
9925 long        lround(double);
9926 long        lroundf(float);
9927 long        lroundl(long double);
9928 double      modf(double, double *);
9929 float       modff(float, float *);
9930 long double modfl(long double, long double *);
9931 double      nan(const char *);
9932 float       nanf(const char *);
9933 long double nanl(const char *);
9934 double      nearbyint(double);
9935 float       nearbyintf(float);
9936 long double nearbyintl(long double);
9937 double      nextafter(double, double);
9938 float       nextafterf(float, float);
9939 long double nextafterl(long double, long double);
9940 double      nexttoward(double, long double);
9941 float       nexttowardf(float, long double);
9942 long double nexttowardl(long double, long double);
9943 double      pow(double, double);
9944 float       powf(float, float);
9945 long double powl(long double, long double);

```



```

9946     double      remainder(double, double);
9947     float        remainderf(float, float);
9948     long double  remainderl(long double, long double);
9949     double       remquo(double, double, int *);
9950     float        remquof(float, float, int *);
9951     long double  remquol(long double, long double, int *);
9952     double       rint(double);
9953     float        rintf(float);
9954     long double  rintl(long double);
9955     double       round(double);
9956     float        roundf(float);
9957     long double  roundl(long double);
9958     double       scalbln(double, long);
9959     float        scalblnf(float, long);
9960     long double  scalblnl(long double, long);
9961     double       scalbn(double, int);
9962     float        scalbnf(float, int);
9963     long double  scalbnl(long double, int);
9964     double       sin(double);
9965     float        sinf(float);
9966     double       sinh(double);
9967     float        sinhlf(float);
9968     long double  sinhl(long double);
9969     long double  sinl(long double);
9970     double       sqrt(double);
9971     float        sqrtf(float);
9972     long double  sqrtl(long double);
9973     double       tan(double);
9974     float        tanf(float);
9975     double       tanh(double);
9976     float        tanhlf(float);
9977     long double  tanhl(long double);
9978     long double  tanl(long double);
9979     double       tgamma(double);
9980     float        tgammaf(float);
9981     long double  tgamma1(long double);
9982     double       trunc(double);
9983     float        truncf(float);
9984     long double  trunc1(long double);
9985     XSI double    y0(double);
9986     double       y1(double);
9987     double       yn(int, double);

```

9988 The following external variable shall be defined:

```

9989 XSI extern int signgam;

```

9990 The behavior of each of the functions defined in <math.h> is specified in the System Interfaces  
9991 volume of POSIX.1-2008 for all representable values of its input arguments, except where stated  
9992 otherwise. Each function shall execute as if it were a single operation without generating any  
9993 externally visible exceptional conditions.

**APPLICATION USAGE**

The FP\_CONTRACT pragma can be used to allow (if the state is on) or disallow (if the state is off) the implementation to contract expressions. Each pragma can occur either outside external declarations or preceding all explicit declarations and statements inside a compound statement. When outside external declarations, the pragma takes effect from its occurrence until another FP\_CONTRACT pragma is encountered, or until the end of the translation unit. When inside a compound statement, the pragma takes effect from its occurrence until another FP\_CONTRACT pragma is encountered (including within a nested compound statement), or until the end of the compound statement; at the end of a compound statement the state for the pragma is restored to its condition just before the compound statement. If this pragma is used in any other context, the behavior is undefined. The default state (on or off) for the pragma is implementation-defined.

Applications should use FLT\_MAX as described in the <float.h> header instead of the obsolescent MAXFLOAT.

Note that if FLT\_EVAL\_METHOD is neither 0 nor 1, then some constants might not compare + equal as expected; for example, (double)M\_PI == M\_PI can fail.

**RATIONALE**

Before the ISO/IEC 9899:1999 standard, the math library was defined only for the floating type **double**. All the names formed by appending 'f' or 'l' to a name in <math.h> were reserved to allow for the definition of **float** and **long double** libraries; and the ISO/IEC 9899:1999 standard provides for all three versions of math functions.

The functions *ecvt()*, *fcvt()*, and *gcvt()* have been dropped from the ISO C standard since their capability is available through *sprintf()*.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

<float.h>, <stddef.h>, <sys/types.h>

XSH Section 2.2 (on page 472), *acos()*, *acosh()*, *asin()*, *asinh()*, *atan()*, *atan2()*, *atanh()*, *cbrt()*, *ceil()*, *copysign()*, *cos()*, *cosh()*, *erf()*, *erfc()*, *exp()*, *exp2()*, *expm1()*, *fabs()*, *fdim()*, *floor()*, *fma()*, *fmax()*, *fmin()*, *fmod()*, *fpclassify()*, *frexp()*, *hypot()*, *ilogb()*, *isfinite()*, *isgreater()*, *isgreaterequal()*, *isinf()*, *isless()*, *islessequal()*, *islessgreater()*, *isnan()*, *isnormal()*, *isunordered()*, *j0()*, *ldexp()*, *lgamma()*, *llrint()*, *llround()*, *log()*, *log10()*, *log1p()*, *log2()*, *logb()*, *lrint()*, *lround()*, *modf()*, *nan()*, *nearbyint()*, *nextafter()*, *pow()*, *remainder()*, *remquo()*, *rint()*, *round()*, *scalbln()*, *signbit()*, *sin()*, *sinh()*, *sqrt()*, *tan()*, *tanh()*, *tgamma()*, *trunc()*, *y0()*

**CHANGE HISTORY**

First released in Issue 1.

**Issue 6**

This reference page is updated to align with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/21 is applied, making it clear that the meaning of the FP\_FAST\_FMA macro is unspecified if the macro is undefined.

**Issue 7**

ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #47 (SD5-XBD-ERN-52) is applied, clarifying the wording of the FP\_FAST\_FMA macro.

The MAXFLOAT constant is marked obsolescent.

This reference page is clarified with respect to macros and symbolic constants.

10038 **NAME**

10039       monetary.h — monetary types

10040 **SYNOPSIS**

10041       #include &lt;monetary.h&gt;

10042 **DESCRIPTION**10043       The <monetary.h> header shall define the **locale\_t** type as described in <locale.h>.10044       The <monetary.h> header shall define the **size\_t** type as described in <stddef.h>.10045       The <monetary.h> header shall define the **ssize\_t** type as described in <sys/types.h>.10046       The following shall be declared as functions and may also be defined as macros. Function  
10047       prototypes shall be provided for use with ISO C standard compilers.10048       **ssize\_t**   strfmon(char \*restrict, size\_t, const char \*restrict, ...);10049       **ssize\_t**   strfmon\_l(char \*restrict, size\_t, locale\_t,

10050                   const char \*restrict, ...);

10051 **APPLICATION USAGE**

10052       None.

10053 **RATIONALE**

10054       None.

10055 **FUTURE DIRECTIONS**

10056       None.

10057 **SEE ALSO**

10058       &lt;locale.h&gt;, &lt;stddef.h&gt;, &lt;sys/types.h&gt;

10059       XSH *strfmon()*10060 **CHANGE HISTORY**

10061       First released in Issue 4.

10062 **Issue 6**10063       The **restrict** keyword is added to the prototype for *strfmon()*.10064 **Issue 7**

10065       The &lt;monetary.h&gt; header is moved from the XSI option to the Base.

10066       The *strfmon\_l()* function is added from The Open Group Technical Standard, 2006, Extended API  
10067       Set Part 4.10068       A declaration for the **locale\_t** type is added.

10069 **NAME**10070       mqueue.h — message queues (**REALTIME**)10071 **SYNOPSIS**

10072 MSG    #include &lt;mqueue.h&gt;

10073 **DESCRIPTION**10074       The <mqueue.h> header shall define the **mqd\_t** type, which is used for message queue  
10075       descriptors. This is not an array type.10076       The <mqueue.h> header shall define the **pthread\_attr\_t**, **size\_t**, and **ssize\_t** types as described  
10077       in <sys/types.h>.10078       The <mqueue.h> header shall define the **struct timespec** structure as described in <time.h>.10079       The tag **sigevent** shall be declared as naming an incomplete structure type, the contents of which  
10080       are described in the <signal.h> header.10081       The <mqueue.h> header shall define the **mq\_attr** structure, which is used in getting and setting  
10082       the attributes of a message queue. Attributes are initially set when the message queue is created.  
10083       An **mq\_attr** structure shall have at least the following fields:

10084	long	mq_flags	Message queue flags.
10085	long	mq_maxmsg	Maximum number of messages.
10086	long	mq_msgsize	Maximum message size.
10087	long	mq_curmsgs	Number of messages currently queued.

10088       The following shall be declared as functions and may also be defined as macros. Function  
10089       prototypes shall be provided.

```

10090     int      mq_close(mqd_t);
10091     int      mq_getattr(mqd_t, struct mq_attr *);
10092     int      mq_notify(mqd_t, const struct sigevent *);
10093     mqd_t    mq_open(const char *, int, ...);
10094     ssize_t  mq_receive(mqd_t, char *, size_t, unsigned *);
10095     int      mq_send(mqd_t, const char *, size_t, unsigned);
10096     int      mq_setattr(mqd_t, const struct mq_attr *restrict,
10097                        struct mq_attr *restrict);
10098     ssize_t  mq_timedreceive(mqd_t, char *restrict, size_t,
10099                        unsigned *restrict, const struct timespec *restrict);
10100     int      mq_timedsend(mqd_t, const char *, size_t, unsigned,
10101                        const struct timespec *);
10102     int      mq_unlink(const char *);

```

10103       Inclusion of the <mqueue.h> header may make visible symbols defined in the headers  
10104       <fcntl.h>, <signal.h>, and <time.h>.10105 **APPLICATION USAGE**

10106       None.

10107 **RATIONALE**

10108       None.

10109 **FUTURE DIRECTIONS**

10110       None.

10111 **SEE ALSO**

10112 [<fcntl.h>](#), [<signal.h>](#), [<sys/types.h>](#), [<time.h>](#)

10113 XSH [mq\\_close\(\)](#), [mq\\_getattr\(\)](#), [mq\\_notify\(\)](#), [mq\\_open\(\)](#), [mq\\_receive\(\)](#), [mq\\_send\(\)](#), [mq\\_setattr\(\)](#),  
10114 [mq\\_unlink\(\)](#)

10115 **CHANGE HISTORY**

10116 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

10117 **Issue 6**

10118 The **<mqueue.h>** header is marked as part of the Message Passing option.

10119 The [mq\\_timedreceive\(\)](#) and [mq\\_timedsend\(\)](#) functions are added for alignment with IEEE Std  
10120 1003.1d-1999.

10121 The **restrict** keyword is added to the prototypes for [mq\\_setattr\(\)](#) and [mq\\_timedreceive\(\)](#).

10122 **Issue 7**

10123 Type and structure declarations are added.

10124 **NAME**

10125 ndbm.h — definitions for ndbm database operations

10126 **SYNOPSIS**10127 XSI `#include <ndbm.h>`10128 **DESCRIPTION**10129 The <ndbm.h> header shall define the **datum** type as a structure, which shall include at least the  
10130 following members:10131 `void *dptr` A pointer to the application's data.  
10132 `size_t dsize` The size of the object pointed to by *dptr*.10133 The <ndbm.h> header shall define the **size\_t** type as described in <stddef.h>.10134 The <ndbm.h> header shall define the **DBM** type.10135 The <ndbm.h> header shall define the following symbolic constants as possible values for the  
10136 *store\_mode* argument to *dbm\_store()*:10137 **DBM\_INSERT** Insertion of new entries only.10138 **DBM\_REPLACE** Allow replacing existing entries.10139 The following shall be declared as functions and may also be defined as macros. Function  
10140 prototypes shall be provided.10141 `int dbm_clearerr(DBM *);`  
10142 `void dbm_close(DBM *);`  
10143 `int dbm_delete(DBM *, datum);`  
10144 `int dbm_error(DBM *);`  
10145 `datum dbm_fetch(DBM *, datum);`  
10146 `datum dbm_firstkey(DBM *);`  
10147 `datum dbm_nextkey(DBM *);`  
10148 `DBM *dbm_open(const char *, int, mode_t);`  
10149 `int dbm_store(DBM *, datum, datum, int);`10150 The <ndbm.h> header shall define the **mode\_t** type through **typedef**, as described in  
10151 <sys/types.h>.10152 **APPLICATION USAGE**

10153 None.

10154 **RATIONALE**

10155 None.

10156 **FUTURE DIRECTIONS**

10157 None.

10158 **SEE ALSO**

10159 &lt;stddef.h&gt;, &lt;sys/types.h&gt;

10160 XSH *dbm\_clearerr()*10161 **CHANGE HISTORY**

10162 First released in Issue 4, Version 2.

10163 **Issue 5**

10164 References to the definitions of **size\_t** and **mode\_t** are added to the DESCRIPTION.

10165 **Issue 7**

10166 This reference page is clarified with respect to macros and symbolic constants.

10167 **NAME**

10168 net/if.h — sockets local interfaces

10169 **SYNOPSIS**

10170 #include &lt;net/if.h&gt;

10171 **DESCRIPTION**10172 The <net/if.h> header shall define the **if\_nameindex** structure, which shall include at least the  
10173 following members:10174 unsigned if\_index Numeric index of the interface.  
10175 char \*if\_name Null-terminated name of the interface.10176 The <net/if.h> header shall define the following symbolic constant for the length of a buffer  
10177 containing an interface name (including the terminating NULL character):

10178 IF\_NAMESIZE Interface name length.

10179 The following shall be declared as functions and may also be defined as macros. Function  
10180 prototypes shall be provided.10181 void if\_freenameindex(struct if\_nameindex \*);  
10182 char \*if\_indextoname(unsigned, char \*);  
10183 struct if\_nameindex \*if\_nameindex(void);  
10184 unsigned if\_nametoindex(const char \*);10185 **APPLICATION USAGE**

10186 None.

10187 **RATIONALE**

10188 None.

10189 **FUTURE DIRECTIONS**

10190 None.

10191 **SEE ALSO**10192 XSH [if\\_freenameindex\(\)](#), [if\\_indextoname\(\)](#), [if\\_nameindex\(\)](#), [if\\_nametoindex\(\)](#)10193 **CHANGE HISTORY**

10194 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

10195 **Issue 7**

10196 This reference page is clarified with respect to macros and symbolic constants.



10197 **NAME**

10198 netdb.h — definitions for network database operations

10199 **SYNOPSIS**

10200 #include <netdb.h>

10201 **DESCRIPTION**

10202 The <netdb.h> header may define the **in\_port\_t** type and the **in\_addr\_t** type as described in  
10203 <netinet/in.h>.

10204 The <netdb.h> header shall define the **hostent** structure, which shall include at least the  
10205 following members:

10206	char	*h_name	Official name of the host.
10207	char	**h_aliases	A pointer to an array of pointers to 10208 alternative host names, terminated by a 10209 null pointer.
10210	int	h_addrtype	Address type.
10211	int	h_length	The length, in bytes, of the address.
10212	char	**h_addr_list	A pointer to an array of pointers to network 10213 addresses (in network byte order) for the host, 10214 terminated by a null pointer.

10215 The <netdb.h> header shall define the **netent** structure, which shall include at least the  
10216 following members:

10217	char	*n_name	Official, fully-qualified (including the 10218 domain) name of the host.
10219	char	**n_aliases	A pointer to an array of pointers to 10220 alternative network names, terminated by a 10221 null pointer.
10222	int	n_addrtype	The address type of the network.
10223	uint32_t	n_net	The network number, in host byte order.

10224 The <netdb.h> header shall define the **uint32\_t** type as described in <inttypes.h>.

10225 The <netdb.h> header shall define the **protoent** structure, which shall include at least the  
10226 following members:

10227	char	*p_name	Official name of the protocol.
10228	char	**p_aliases	A pointer to an array of pointers to 10229 alternative protocol names, terminated by 10230 a null pointer.
10231	int	p_proto	The protocol number.

10232 The <netdb.h> header shall define the **servent** structure, which shall include at least the  
10233 following members:

10234	char	*s_name	Official name of the service.
10235	char	**s_aliases	A pointer to an array of pointers to 10236 alternative service names, terminated by 10237 a null pointer.
10238	int	s_port	A value which, when converted to uint16_t, 10239 yields the port number in network byte order 10240 at which the service resides.
10241	char	*s_proto	The name of the protocol to use when 10242 contacting the service.

10243 The <netdb.h> header shall define the IPPORT\_RESERVED symbolic constant with the value of  
10244 the highest reserved Internet port number.

#### 10245 Address Information Structure

10246 The <netdb.h> header shall define the **addrinfo** structure, which shall include at least the  
10247 following members:

10248	int	ai_flags	Input flags.
10249	int	ai_family	Address family of socket.
10250	int	ai_socktype	Socket type.
10251	int	ai_protocol	Protocol of socket.
10252	socklen_t	ai_addrlen	Length of socket address.
10253	struct sockaddr	*ai_addr	Socket address of socket.
10254	char	*ai_canonname	Canonical name of service location.
10255	struct addrinfo	*ai_next	Pointer to next in list.

10256 The <netdb.h> header shall define the following symbolic constants that evaluate to bitwise-  
10257 distinct integer constants for use in the *flags* field of the **addrinfo** structure:

10258	AI_PASSIVE	Socket address is intended for <i>bind()</i> .
10259	AI_CANONNAME	Request for canonical name.
10260	AI_NUMERICHOST	Return numeric host address as name.
10261	AI_NUMERICSERV	Inhibit service name resolution.
10262	AI_V4MAPPED	If no IPv6 addresses are found, query for IPv4 addresses and return them 10263 to the caller as IPv4-mapped IPv6 addresses.
10264	AI_ALL	Query for both IPv4 and IPv6 addresses.
10265	AI_ADDRCONFIG	Query for IPv4 addresses only when an IPv4 address is configured; query 10266 for IPv6 addresses only when an IPv6 address is configured.

10267 The <netdb.h> header shall define the following symbolic constants that evaluate to bitwise-  
10268 distinct integer constants for use in the *flags* argument to *getnameinfo()*:

10269	NI_NOFQDN	Only the nodename portion of the FQDN is returned for local hosts.
10270	NI_NUMERICHOST	The numeric form of the node's address is returned instead of its name.
10271	NI_NAMEREQD	Return an error if the node's name cannot be located in the database.
10272	NI_NUMERICSERV	The numeric form of the service address is returned instead of its name.
10273	NI_NUMERICSCOPE	
10274		For IPv6 addresses, the numeric form of the scope identifier is returned 10275 instead of its name.
10276	NI_DGRAM	Indicates that the service is a datagram service (SOCK_DGRAM).

## Address Information Errors

The <netdb.h> header shall define the following symbolic constants for use as error values for *getaddrinfo()* and *getnameinfo()*. The values shall be suitable for use in *#if* preprocessing directives.

EAI_AGAIN	The name could not be resolved at this time. Future attempts may succeed.
EAI_BADFLAGS	The flags had an invalid value.
EAI_FAIL	A non-recoverable error occurred.
EAI_FAMILY	The address family was not recognized or the address length was invalid for the specified family.
EAI_MEMORY	There was a memory allocation failure.
EAI_NONAME	The name does not resolve for the supplied parameters.
	NI_NAMEREQD is set and the host's name cannot be located, or both <i>nodename</i> and <i>servname</i> were null.
EAI_SERVICE	The service passed was not recognized for the specified socket type.
EAI_SOCKTYPE	The intended socket type was not recognized.
EAI_SYSTEM	A system error occurred. The error code can be found in <i>errno</i> .
EAI_OVERFLOW	An argument buffer overflowed.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

void	<i>endhostent</i> (void);
void	<i>endnetent</i> (void);
void	<i>endprotoent</i> (void);
void	<i>endservent</i> (void);
void	<i>freeaddrinfo</i> (struct <i>addrinfo</i> *);
const char	* <i>gai_strerror</i> (int);
int	<i>getaddrinfo</i> (const char *restrict, const char *restrict, const struct <i>addrinfo</i> *restrict, struct <i>addrinfo</i> **restrict);
struct <i>hostent</i>	* <i>gethostent</i> (void);
int	<i>getnameinfo</i> (const struct <i>sockaddr</i> *restrict, <i>socklen_t</i> , char *restrict, <i>socklen_t</i> , char *restrict, <i>socklen_t</i> , int);
struct <i>netent</i>	* <i>getnetbyaddr</i> ( <i>uint32_t</i> , int);
struct <i>netent</i>	* <i>getnetbyname</i> (const char *);
struct <i>netent</i>	* <i>getnetent</i> (void);
struct <i>protoent</i>	* <i>getprotobyname</i> (const char *);
struct <i>protoent</i>	* <i>getprotobynumber</i> (int);
struct <i>protoent</i>	* <i>getprotoent</i> (void);
struct <i>servent</i>	* <i>getservbyname</i> (const char *, const char *);
struct <i>servent</i>	* <i>getservbyport</i> (int, const char *);
struct <i>servent</i>	* <i>getservent</i> (void);
void	<i>sethostent</i> (int);
void	<i>setnetent</i> (int);
void	<i>setprotoent</i> (int);

10322           void                       setservent(int);

10323           The <netdb.h> header shall define the **socklen\_t** type through **typedef**, as described in  
10324           <sys/socket.h>.

10325           Inclusion of the <netdb.h> header may also make visible all symbols from <netinet/in.h>,  
10326           <sys/socket.h>, and <inttypes.h>.

#### 10327 APPLICATION USAGE

10328           None.

#### 10329 RATIONALE

10330           None.

#### 10331 FUTURE DIRECTIONS

10332           None.

#### 10333 SEE ALSO

10334           <inttypes.h>, <netinet/in.h>, <sys/socket.h>

10335           XSH *bind()*, *endhostent()*, *endnetent()*, *endprotoent()*, *endservent()*, *freeaddrinfo()*, *gai\_strerror()*,  
10336           *getnameinfo()*

#### 10337 CHANGE HISTORY

10338           First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

10339           The Open Group Base Resolution bwg2001-009 is applied, which changes the return type for  
10340           *gai\_strerror()* from **char \*** to **const char \***. This is for coordination with the IPnG Working Group.

10341           IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/11 is applied, adding a description of the  
10342           NI\_NUMERICSCOPE macro and correcting the *getnameinfo()* function prototype. These changes  
10343           are for alignment with IPv6.

#### 10344 Issue 7

10345           SD5-XBD-ERN-14 is applied, changing the description of the *s\_port* member of the **servent**  
10346           structure.

10347           The obsolescent *h\_errno* external integer, and the obsolescent *gethostbyaddr()* and *gethostbyname()*  
10348           functions are removed, along with the HOST\_NOT\_FOUND, NO\_DATA, NO\_RECOVERY, and  
10349           TRY\_AGAIN macros.

10350           This reference page is clarified with respect to macros and symbolic constants.

10351 **NAME**10352        `netinet/in.h` — Internet address family10353 **SYNOPSIS**10354        `#include <netinet/in.h>`10355 **DESCRIPTION**10356        The `<netinet/in.h>` header shall define the following types:10357        **in\_port\_t**    Equivalent to the type **uint16\_t** as described in `<inttypes.h>`.10358        **in\_addr\_t**    Equivalent to the type **uint32\_t** as described in `<inttypes.h>`.10359        The `<netinet_in.h>` header shall define the **sa\_family\_t** type as described in `<sys/socket.h>`.10360        The `<netinet_in.h>` header shall define the **uint8\_t** and **uint32\_t** types as described in  
10361 `<inttypes.h>`. Inclusion of the `<netinet/in.h>` header may also make visible all symbols from  
10362 `<inttypes.h>` and `<sys/socket.h>`.10363        The `<netinet/in.h>` header shall define the **in\_addr** structure, which shall include at least the  
10364 following member:10365        `in_addr_t    s_addr`10366        The `<netinet/in.h>` header shall define the **sockaddr\_in** structure, which shall include at least  
10367 the following members:10368        `sa_family_t        sin_family        AF_INET.`  
10369        `in_port_t           sin_port        Port number.`  
10370        `struct in_addr       sin_addr        IP address.`10371        The `sin_port` and `sin_addr` members shall be in network byte order.10372        The **sockaddr\_in** structure is used to store addresses for the Internet address family. Pointers to  
10373 this type shall be cast by applications to **struct sockaddr \*** for use with socket functions.10374 **IPv6**        The `<netinet/in.h>` header shall define the **in6\_addr** structure, which shall include at least the  
10375 following member:10376        `uint8_t    s6_addr[16]`

10377        This array is used to contain a 128-bit IPv6 address, stored in network byte order.

10378        The `<netinet/in.h>` header shall define the **sockaddr\_in6** structure, which shall include at least  
10379 the following members:10380        `sa_family_t        sin6_family        AF_INET6.`  
10381        `in_port_t           sin6_port        Port number.`  
10382        `uint32_t           sin6_flowinfo    IPv6 traffic class and flow information.`  
10383        `struct in6_addr       sin6_addr        IPv6 address.`  
10384        `uint32_t           sin6_scope_id    Set of interfaces for a scope.`10385        The `sin6_port` and `sin6_addr` members shall be in network byte order.10386        Prior to calling a function in this standard which reads values from a **sockaddr\_in6** structure (for  
10387 example, `bind()` or `connect()`), the application shall ensure that all members of the structure,  
10388 including any additional non-standard members, if any, are initialized. If the **sockaddr\_in6**  
10389 structure has a non-standard member, and that member has a value other than the value that  
10390 would result from default initialization, the behavior of any function in this standard that reads  
10391 values from the **sockaddr\_in6** structure is implementation-defined. All functions in this  
10392 standard that return data in a **sockaddr\_in6** structure (for example, `getaddrinfo()` or `accept()`)  
10393 shall initialize the structure in a way that meets the above requirements, and shall ensure that

each non-standard member, if any, has a value that produces the same behavior as default initialization would in all functions in this standard which read values from a **sockaddr\_in6** structure.

The *sin6\_scope\_id* field is a 32-bit integer that identifies a set of interfaces as appropriate for the scope of the address carried in the *sin6\_addr* field. For a link scope *sin6\_addr*, the application shall ensure that *sin6\_scope\_id* is a link index. For a site scope *sin6\_addr*, the application shall ensure that *sin6\_scope\_id* is a site index. The mapping of *sin6\_scope\_id* to an interface or set of interfaces is implementation-defined.

The <netinet/in.h> header shall declare the following external variable:

```
const struct in6_addr in6addr_any
```

This variable is initialized by the system to contain the wildcard IPv6 address. The <netinet/in.h> header also defines the IN6ADDR\_ANY\_INIT macro. This macro must be constant at compile time and can be used to initialize a variable of type **struct in6\_addr** to the IPv6 wildcard address.

The <netinet/in.h> header shall declare the following external variable:

```
const struct in6_addr in6addr_loopback
```

This variable is initialized by the system to contain the loopback IPv6 address. The <netinet/in.h> header also defines the IN6ADDR\_LOOPBACK\_INIT macro. This macro must be constant at compile time and can be used to initialize a variable of type **struct in6\_addr** to the IPv6 loopback address.

The <netinet/in.h> header shall define the **ipv6\_mreq** structure, which shall include at least the following members:

```
struct in6_addr  ipv6mr_multiaddr  IPv6 multicast address.
unsigned        ipv6mr_interface  Interface index.
```

The <netinet/in.h> header shall define the following symbolic constants for use as values of the *level* argument of *getsockopt()* and *setsockopt()*:

IPPROTO_IP	Internet protocol.
IPPROTO_IPV6	Internet Protocol Version 6.

IPPROTO_ICMP	Control message protocol.
--------------	---------------------------

IPPROTO_RAW	Raw IP Packets Protocol.
-------------	--------------------------

IPPROTO_TCP	Transmission control protocol.
-------------	--------------------------------

IPPROTO_UDP	User datagram protocol.
-------------	-------------------------

The <netinet/in.h> header shall define the following symbolic constant for use as a local address in the structure passed to *bind()*:

INADDR_ANY	IPv4 wildcard address.
------------	------------------------

The <netinet/in.h> header shall define the following symbolic constant for use as a destination address in the structures passed to *connect()*, *sendmsg()*, and *sendto()*:

INADDR_BROADCAST	IPv4 broadcast address.
------------------	-------------------------

The <netinet/in.h> header shall define the following symbolic constant, with the value specified, to help applications declare buffers of the proper size to store IPv4 addresses in string

10434		form:
10435		INET_ADDRSTRLEN      16. Length of the string form for IP.
10436		The <i>htonl()</i> , <i>htons()</i> , <i>ntohl()</i> , and <i>ntohs()</i> functions shall be available as described in
10437		<arpa/inet.h>. Inclusion of the <netinet/in.h> header may also make visible all symbols from
10438		<arpa/inet.h>.
10439	IP6	The <netinet/in.h> header shall define the following symbolic constant, with the value
10440		specified, to help applications declare buffers of the proper size to store IPv6 addresses in string
10441		form:
10442		INET6_ADDRSTRLEN      46. Length of the string form for IPv6.
10443	IP6	The <netinet/in.h> header shall define the following symbolic constants, with distinct integer
10444		values, for use in the <i>option_name</i> argument in the <i>getsockopt()</i> or <i>setsockopt()</i> functions at
10445		protocol level IPPROTO_IPV6:
10446		IPV6_JOIN_GROUP      Join a multicast group.
10447		IPV6_LEAVE_GROUP      Quit a multicast group.
10448		IPV6_MULTICAST_HOPS
10449		Multicast hop limit.
10450		IPV6_MULTICAST_IF      Interface to use for outgoing multicast packets.
10451		IPV6_MULTICAST_LOOP
10452		Multicast packets are delivered back to the local application.
10453		IPV6_UNICAST_HOPS      Unicast hop limit.
10454		IPV6_V6ONLY            Restrict AF_INET6 socket to IPv6 communications only.
10455		The <netinet/in.h> header shall define the following macros that test for special IPv6 addresses.
10456		Each macro is of type <b>int</b> and takes a single argument of type <b>const struct in6_addr *</b> :
10457		IN6_IS_ADDR_UNSPECIFIED
10458		Unspecified address.
10459		IN6_IS_ADDR_LOOPBACK
10460		Loopback address.
10461		IN6_IS_ADDR_MULTICAST
10462		Multicast address.
10463		IN6_IS_ADDR_LINKLOCAL
10464		Unicast link-local address.
10465		IN6_IS_ADDR_SITELOCAL
10466		Unicast site-local address.
10467		IN6_IS_ADDR_V4MAPPED
10468		IPv4 mapped address.
10469		IN6_IS_ADDR_V4COMPAT
10470		IPv4-compatible address.
10471		IN6_IS_ADDR_MC_NODELOCAL
10472		Multicast node-local address.



```

10473 IN6_IS_ADDR_MC_LINKLOCAL
10474     Multicast link-local address.
10475 IN6_IS_ADDR_MC_SITELOCAL
10476     Multicast site-local address.
10477 IN6_IS_ADDR_MC_ORGLOCAL
10478     Multicast organization-local address.
10479 IN6_IS_ADDR_MC_GLOBAL
10480     Multicast global address.

```

## APPLICATION USAGE

Although applications are required to initialize all members (including any non-standard ones) of a **sockaddr\_in6** structure, the same is not required for the **sockaddr\_in** structure, since historically many applications only initialized the standard members. Despite this, applications are encouraged to initialize **sockaddr\_in** structures in a manner similar to the required initialization of **sockaddr\_in6** structures.

Although it is common practice to initialize a **sockaddr\_in6** structure using:

```

struct sockaddr_in6 sa;
memset(&sa, 0, sizeof sa);

```

this method is not portable according to this standard, because the structure can contain pointer or floating-point members that are not required to have an all-bits-zero representation after default initialization. Portable methods make use of default initialization; for example:

```

struct sockaddr_in6 sa = { 0 };

```

or:

```

static struct sockaddr_in6 sa_init;
struct sockaddr_in6 sa = sa_init;

```

A future version of this standard may require that a pointer object with an all-bits-zero representation is a null pointer, and that **sockaddr\_in6** does not have any floating-point members if a floating-point object with an all-bits-zero representation does not have the value 0.0.

## RATIONALE

The INADDR\_ANY and INADDR\_BROADCAST values are byte-order-neutral and thus their byte order is not specified. Many implementations have additional constants as extensions, such as INADDR\_LOOPBACK, that are not byte-order-neutral. Traditionally, these constants are in host byte order, requiring the use of *htonl()* when using them in a **sockaddr\_in** structure.

## FUTURE DIRECTIONS

None.

## SEE ALSO

Section 4.10 (on page 110), [<arpa/inet.h>](#), [<inttypes.h>](#), [<sys/socket.h>](#)

XSH *connect()*, *getsockopt()*, *htonl()*, *sendmsg()*, *sendto()*, *setsockopt()*

## CHANGE HISTORY

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

The *sin\_zero* member was removed from the **sockaddr\_in** structure as per The Open Group Base Resolution bwg2001-004.



- 10515 IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/12 is applied, adding **const** qualifiers to  
 10516 the *in6addr\_any* and *in6addr\_loopback* external variables.
- 10517 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/22 is applied, making it clear which  
 10518 structure members are in network byte order.
- 10519 **Issue 7**
- 10520 This reference page is clarified with respect to macros and symbolic constants.
- 10521 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0061 [355] is applied.

10522 **NAME**

10523       netinet/tcp.h — definitions for the Internet Transmission Control Protocol (TCP)

10524 **SYNOPSIS**

10525       #include &lt;netinet/tcp.h&gt;

10526 **DESCRIPTION**10527       The <netinet/tcp.h> header shall define the following symbolic constant for use as a socket  
10528       option at the IPPROTO\_TCP level:

10529       TCP\_NODELAY   Avoid coalescing of small segments.

10530       The implementation need not allow the value of the option to be set via *setsockopt()* or retrieved  
10531       via *getsockopt()*.10532 **APPLICATION USAGE**

10533       None.

10534 **RATIONALE**

10535       None.

10536 **FUTURE DIRECTIONS**

10537       None.

10538 **SEE ALSO**

10539       &lt;sys/socket.h&gt;

10540       XSH *getsockopt()*, *setsockopt()*10541 **CHANGE HISTORY**

10542       First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

10543 **Issue 7**

10544       This reference page is clarified with respect to macros and symbolic constants.

10545 **NAME**

10546 nl\_types.h — data types

10547 **SYNOPSIS**

10548 #include &lt;nl\_types.h&gt;

10549 **DESCRIPTION**

10550 The &lt;nl\_types.h&gt; header shall define at least the following types:

10551 **nl\_catd** Used by the message catalog functions *catopen()*, *catgets()*, and *catclose()*  
 10552 to identify a catalog descriptor.

10553 **nl\_item** Used by *nl\_langinfo()* to identify items of *langinfo* data. Values of objects  
 10554 of type **nl\_item** are defined in <langinfo.h>.

10555 The &lt;nl\_types.h&gt; header shall define at least the following symbolic constants:

10556 **NL\_SETD** Used by *gencat* when no *\$set* directive is specified in a message text source  
 10557 file. This constant can be passed as the value of *set\_id* on subsequent calls  
 10558 to *catgets()* (that is, to retrieve messages from the default message set).  
 10559 The value of **NL\_SETD** is implementation-defined.

10560 **NL\_CAT\_LOCALE** Value that must be passed as the *oflag* argument to *catopen()* to ensure that  
 10561 message catalog selection depends on the *LC\_MESSAGES* locale category,  
 10562 rather than directly on the *LANG* environment variable.

10563 The following shall be declared as functions and may also be defined as macros. Function  
 10564 prototypes shall be provided.

```
10565 int      catclose(nl_catd);
10566 char     *catgets(nl_catd, int, int, const char *);
10567 nl_catd  catopen(const char *, int);
```

10568 **APPLICATION USAGE**

10569 None.

10570 **RATIONALE**

10571 None.

10572 **FUTURE DIRECTIONS**

10573 None.

10574 **SEE ALSO**

10575 &lt;langinfo.h&gt;

10576 XSH *catclose()*, *catgets()*, *catopen()*, *nl\_langinfo()*10577 XCU *gencat*10578 **CHANGE HISTORY**

10579 First released in Issue 2.

10580 **Issue 7**

10581 The &lt;nl\_types.h&gt; header is moved from the XSI option to the Base.

10582 This reference page is clarified with respect to macros and symbolic constants.

10583 **NAME**

10584 poll.h — definitions for the poll() function

10585 **SYNOPSIS**

10586 #include &lt;poll.h&gt;

10587 **DESCRIPTION**10588 The <poll.h> header shall define the **pollfd** structure, which shall include at least the following  
10589 members:

10590 int fd The following descriptor being polled.

10591 short events The input event flags (see below).

10592 short revents The output event flags (see below).

10593 The <poll.h> header shall define the following type through **typedef**:10594 **nfds\_t** An unsigned integer type used for the number of file descriptors.10595 The implementation shall support one or more programming environments in which the width  
10596 of **nfds\_t** is no greater than the width of type **long**. The names of these programming  
10597 environments can be obtained using the *confstr()* function or the *getconf* utility.10598 The <poll.h> header shall define the following symbolic constants, zero or more of which may  
10599 be OR'ed together to form the *events* or *revents* members in the **pollfd** structure:

10600 POLLIN Data other than high-priority data may be read without blocking.

10601 POLLRDNORM Normal data may be read without blocking.

10602 POLLRDBAND Priority data may be read without blocking.

10603 POLLPRI High priority data may be read without blocking.

10604 POLLOUT Normal data may be written without blocking.

10605 POLLWRNORM Equivalent to POLLOUT.

10606 POLLWRBAND Priority data may be written.

10607 POLLERR An error has occurred (*revents* only).10608 POLLHUP Device has been disconnected (*revents* only).10609 POLLNVAL Invalid *fd* member (*revents* only).10610 The significance and semantics of normal, priority, and high-priority data are file and device-  
10611 specific.10612 The following shall be declared as a function and may also be defined as a macro. A function  
10613 prototype shall be provided.

10614 int poll(struct pollfd [], nfds\_t, int);

10615 **APPLICATION USAGE**

10616 None.

10617 **RATIONALE**

10618 None.

10619 **FUTURE DIRECTIONS**

10620 None.

10621 **SEE ALSO**

10622 XSH *confstr()*, *poll()*

10623 XCU *getconf*

10624 **CHANGE HISTORY**

10625 First released in Issue 4, Version 2.

10626 **Issue 6**

10627 The description of the symbolic constants is updated to match the *poll()* function.

10628 Text related to STREAMS has been moved to the *poll()* reference page.

10629 A note is added to the DESCRIPTION regarding the significance and semantics of normal,  
10630 priority, and high-priority data.

10631 **Issue 7**

10632 The <poll.h> header is moved from the XSI option to the Base.

10633 **NAME**

10634 pthread.h — threads

10635 **SYNOPSIS**

10636 #include &lt;pthread.h&gt;

10637 **DESCRIPTION**

10638 The &lt;pthread.h&gt; header shall define the following symbolic constants:

10639 PTHREAD\_BARRIER\_SERIAL\_THREAD  
 10640 PTHREAD\_CANCEL\_ASYNCHRONOUS  
 10641 PTHREAD\_CANCEL\_ENABLE  
 10642 PTHREAD\_CANCEL\_DEFERRED  
 10643 PTHREAD\_CANCEL\_DISABLE  
 10644 PTHREAD\_CANCELED  
 10645 PTHREAD\_CREATE\_DETACHED  
 10646 PTHREAD\_CREATE\_JOINABLE  
 10647 TPS PTHREAD\_EXPLICIT\_SCHED  
 10648 PTHREAD\_INHERIT\_SCHED  
 10649 PTHREAD\_MUTEX\_DEFAULT  
 10650 PTHREAD\_MUTEX\_ERRORCHECK  
 10651 PTHREAD\_MUTEX\_NORMAL  
 10652 PTHREAD\_MUTEX\_RECURSIVE  
 10653 PTHREAD\_MUTEX\_ROBUST  
 10654 PTHREAD\_MUTEX\_STALLED  
 10655 PTHREAD\_ONCE\_INIT  
 10656 RPI | TPI PTHREAD\_PRIO\_INHERIT  
 10657 MC1 PTHREAD\_PRIO\_NONE  
 10658 RPP | TPP PTHREAD\_PRIO\_PROTECT  
 10659 PTHREAD\_PROCESS\_SHARED  
 10660 PTHREAD\_PROCESS\_PRIVATE  
 10661 TPS PTHREAD\_SCOPE\_PROCESS  
 10662 PTHREAD\_SCOPE\_SYSTEM

10663 The <pthread.h> header shall define the following compile-time constant expressions valid as  
 10664 initializers for the following types:

Name	Initializer for Type
PTHREAD_COND_INITIALIZER	pthread_cond_t
PTHREAD_MUTEX_INITIALIZER	pthread_mutex_t
PTHREAD_RWLOCK_INITIALIZER	pthread_rwlock_t

10669 The <pthread.h> header shall define the pthread\_attr\_t, pthread\_barrier\_t,  
 10670 pthread\_barrierattr\_t, pthread\_cond\_t, pthread\_condattr\_t, pthread\_key\_t, pthread\_mutex\_t,  
 10671 pthread\_mutexattr\_t, pthread\_once\_t, pthread\_rwlock\_t, pthread\_rwlockattr\_t,  
 10672 pthread\_spinlock\_t, and pthread\_t types as described in <sys/types.h>.

10673 The following shall be declared as functions and may also be defined as macros. Function  
 10674 prototypes shall be provided.

```
10675 int pthread_atfork(void (*)(void), void (*)(void),
10676                  void (*)(void));
10677 int pthread_attr_destroy(pthread_attr_t *);
10678 int pthread_attr_getdetachstate(const pthread_attr_t *, int *);
10679 int pthread_attr_getguardsize(const pthread_attr_t *restrict,
```

```

10680         size_t *restrict);
10681 TPS    int    pthread_attr_getinheritsched(const pthread_attr_t *restrict,
10682         int *restrict);
10683         int    pthread_attr_getschedparam(const pthread_attr_t *restrict,
10684         struct sched_param *restrict);
10685 TPS    int    pthread_attr_getschedpolicy(const pthread_attr_t *restrict,
10686         int *restrict);
10687         int    pthread_attr_getscope(const pthread_attr_t *restrict,
10688         int *restrict);
10689 TSA TSS int    pthread_attr_getstack(const pthread_attr_t *restrict,
10690         void **restrict, size_t *restrict);
10691 TSS    int    pthread_attr_getstacksize(const pthread_attr_t *restrict,
10692         size_t *restrict);
10693         int    pthread_attr_init(pthread_attr_t *);
10694         int    pthread_attr_setdetachstate(pthread_attr_t *, int);
10695         int    pthread_attr_setguardsize(pthread_attr_t *, size_t);
10696 TPS    int    pthread_attr_setinheritsched(pthread_attr_t *, int);
10697         int    pthread_attr_setschedparam(pthread_attr_t *restrict,
10698         const struct sched_param *restrict);
10699 TPS    int    pthread_attr_setschedpolicy(pthread_attr_t *, int);
10700         int    pthread_attr_setscope(pthread_attr_t *, int);
10701 TSA TSS int    pthread_attr_setstack(pthread_attr_t *, void *, size_t);
10702 TSS    int    pthread_attr_setstacksize(pthread_attr_t *, size_t);
10703         int    pthread_barrier_destroy(pthread_barrier_t *);
10704         int    pthread_barrier_init(pthread_barrier_t *restrict,
10705         const pthread_barrierattr_t *restrict, unsigned);
10706         int    pthread_barrier_wait(pthread_barrier_t *);
10707         int    pthread_barrierattr_destroy(pthread_barrierattr_t *);
10708 TSH    int    pthread_barrierattr_getpshared(
10709         const pthread_barrierattr_t *restrict, int *restrict);
10710         int    pthread_barrierattr_init(pthread_barrierattr_t *);
10711 TSH    int    pthread_barrierattr_setpshared(pthread_barrierattr_t *, int);
10712         int    pthread_cancel(pthread_t);
10713         int    pthread_cond_broadcast(pthread_cond_t *);
10714         int    pthread_cond_destroy(pthread_cond_t *);
10715         int    pthread_cond_init(pthread_cond_t *restrict,
10716         const pthread_condattr_t *restrict);
10717         int    pthread_cond_signal(pthread_cond_t *);
10718         int    pthread_cond_timedwait(pthread_cond_t *restrict,
10719         pthread_mutex_t *restrict, const struct timespec *restrict);
10720         int    pthread_cond_wait(pthread_cond_t *restrict,
10721         pthread_mutex_t *restrict);
10722         int    pthread_condattr_destroy(pthread_condattr_t *);
10723         int    pthread_condattr_getclock(const pthread_condattr_t *restrict,
10724         clockid_t *restrict);
10725 TSH    int    pthread_condattr_getpshared(const pthread_condattr_t *restrict,
10726         int *restrict);
10727         int    pthread_condattr_init(pthread_condattr_t *);
10728         int    pthread_condattr_setclock(pthread_condattr_t *, clockid_t);
10729 TSH    int    pthread_condattr_setpshared(pthread_condattr_t *, int);
10730         int    pthread_create(pthread_t *restrict, const pthread_attr_t *restrict,
10731         void *(*)(void*), void *restrict);

```

```

10732     int    pthread_detach(pthread_t);
10733     int    pthread_equal(pthread_t, pthread_t);
10734     void    pthread_exit(void *);
10735     OB XSI  int    pthread_getconcurrency(void);
10736     TCT     int    pthread_getcpuclockid(pthread_t, clockid_t *);
10737     TPS     int    pthread_getschedparam(pthread_t, int *restrict,
10738         struct sched_param *restrict);
10739     void *pthread_getspecific(pthread_key_t);
10740     int    pthread_join(pthread_t, void **);
10741     int    pthread_key_create(pthread_key_t *, void (*)(void*));
10742     int    pthread_key_delete(pthread_key_t);
10743     int    pthread_mutex_consistent(pthread_mutex_t *);
10744     int    pthread_mutex_destroy(pthread_mutex_t *);
10745     RPP|TPP int    pthread_mutex_getprioceiling(const pthread_mutex_t *restrict,
10746         int *restrict);
10747     int    pthread_mutex_init(pthread_mutex_t *restrict,
10748         const pthread_mutexattr_t *restrict);
10749     int    pthread_mutex_lock(pthread_mutex_t *);
10750     RPP|TPP int    pthread_mutex_setprioceiling(pthread_mutex_t *restrict, int,
10751         int *restrict);
10752     int    pthread_mutex_timedlock(pthread_mutex_t *restrict,
10753         const struct timespec *restrict);
10754     int    pthread_mutex_trylock(pthread_mutex_t *);
10755     int    pthread_mutex_unlock(pthread_mutex_t *);
10756     int    pthread_mutexattr_destroy(pthread_mutexattr_t *);
10757     RPP|TPP int    pthread_mutexattr_getprioceiling(
10758         const pthread_mutexattr_t *restrict, int *restrict);
10759     MC1     int    pthread_mutexattr_getprotocol(const pthread_mutexattr_t *restrict,
10760         int *restrict);
10761     TSH     int    pthread_mutexattr_getpshared(const pthread_mutexattr_t *restrict,
10762         int *restrict);
10763     int    pthread_mutexattr_getrobust(const pthread_mutexattr_t *restrict,
10764         int *restrict);
10765     int    pthread_mutexattr_gettype(const pthread_mutexattr_t *restrict,
10766         int *restrict);
10767     int    pthread_mutexattr_init(pthread_mutexattr_t *);
10768     RPP|TPP int    pthread_mutexattr_setprioceiling(pthread_mutexattr_t *, int);
10769     MC1     int    pthread_mutexattr_setprotocol(pthread_mutexattr_t *, int);
10770     TSH     int    pthread_mutexattr_setpshared(pthread_mutexattr_t *, int);
10771     int    pthread_mutexattr_setrobust(pthread_mutexattr_t *, int);
10772     int    pthread_mutexattr_settype(pthread_mutexattr_t *, int);
10773     int    pthread_once(pthread_once_t *, void (*)(void));
10774     int    pthread_rwlock_destroy(pthread_rwlock_t *);
10775     int    pthread_rwlock_init(pthread_rwlock_t *restrict,
10776         const pthread_rwlockattr_t *restrict);
10777     int    pthread_rwlock_rdlock(pthread_rwlock_t *);
10778     int    pthread_rwlock_timedrdlock(pthread_rwlock_t *restrict,
10779         const struct timespec *restrict);
10780     int    pthread_rwlock_timedwrlock(pthread_rwlock_t *restrict,
10781         const struct timespec *restrict);
10782     int    pthread_rwlock_tryrdlock(pthread_rwlock_t *);
10783     int    pthread_rwlock_trywrlock(pthread_rwlock_t *);

```



```

10784 int pthread_rwlock_unlock(pthread_rwlock_t *);
10785 int pthread_rwlock_wrlock(pthread_rwlock_t *);
10786 int pthread_rwlockattr_destroy(pthread_rwlockattr_t *);
10787 TSH int pthread_rwlockattr_getpshared(
10788     const pthread_rwlockattr_t *restrict, int *restrict);
10789 int pthread_rwlockattr_init(pthread_rwlockattr_t *);
10790 TSH int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *, int);
10791 pthread_t
10792     pthread_self(void);
10793 int pthread_setcancelstate(int, int *);
10794 int pthread_setcanceltype(int, int *);
10795 OB XSI int pthread_setconcurrency(int);
10796 TPS int pthread_setschedparam(pthread_t, int,
10797     const struct sched_param *);
10798 int pthread_setschedprio(pthread_t, int);
10799 int pthread_setspecific(pthread_key_t, const void *);
10800 int pthread_spin_destroy(pthread_spinlock_t *);
10801 int pthread_spin_init(pthread_spinlock_t *, int);
10802 int pthread_spin_lock(pthread_spinlock_t *);
10803 int pthread_spin_trylock(pthread_spinlock_t *);
10804 int pthread_spin_unlock(pthread_spinlock_t *);
10805 void pthread_testcancel(void);

```

The following may be declared as functions, or defined as macros, or both. If functions are declared, function prototypes shall be provided. +

```

10808     pthread_cleanup_pop()
10809     pthread_cleanup_push()

```

Inclusion of the <pthread.h> header shall make symbols defined in the headers <sched.h> and <time.h> visible.

## APPLICATION USAGE

None.

## RATIONALE

None.

## FUTURE DIRECTIONS

None.

## SEE ALSO

<sched.h>, <sys/types.h>, <time.h>

```

10820 XSH pthread_atfork(), pthread_attr_destroy(), pthread_attr_getdetachstate(),
10821     pthread_attr_getguardsize(), pthread_attr_getinheritsched(), pthread_attr_getschedparam(),
10822     pthread_attr_getschedpolicy(), pthread_attr_getscope(), pthread_attr_getstack(),
10823     pthread_attr_getstacksize(), pthread_barrier_destroy(), pthread_barrier_wait(),
10824     pthread_barrierattr_destroy(), pthread_barrierattr_getpshared(), pthread_cancel(),
10825     pthread_cleanup_pop(), pthread_cond_broadcast(), pthread_cond_destroy(), pthread_cond_timedwait(),
10826     pthread_condattr_destroy(), pthread_condattr_getclock(), pthread_condattr_getpshared(),
10827     pthread_create(), pthread_detach(), pthread_equal(), pthread_exit(), pthread_getconcurrency(),
10828     pthread_getcpuclockid(), pthread_getschedparam(), pthread_getspecific(), pthread_join(),
10829     pthread_key_create(), pthread_key_delete(), pthread_mutex_consistent(), pthread_mutex_destroy(),
10830     pthread_mutex_getprioceiling(), pthread_mutex_lock(), pthread_mutex_timedlock(),

```

*pthread\_mutexattr\_destroy(), pthread\_mutexattr\_getprioceiling(), pthread\_mutexattr\_getprotocol(), pthread\_mutexattr\_getpshared(), pthread\_mutexattr\_getrobust(), pthread\_mutexattr\_gettype(), pthread\_once(), pthread\_rwlock\_destroy(), pthread\_rwlock\_rdlock(), pthread\_rwlock\_timedrdlock(), pthread\_rwlock\_timedwrlock(), pthread\_rwlock\_trywrlock(), pthread\_rwlock\_unlock(), pthread\_rwlockattr\_destroy(), pthread\_rwlockattr\_getpshared(), pthread\_self(), pthread\_setcancelstate(), pthread\_setschedprio(), pthread\_spin\_destroy(), pthread\_spin\_lock(), pthread\_spin\_unlock()*

## CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

### Issue 6

The RTT margin markers are broken out into their POSIX options.

The Open Group Corrigendum U021/9 is applied, correcting the prototype for the *pthread\_cond\_wait()* function.

The Open Group Corrigendum U026/2 is applied, correcting the prototype for the *pthread\_setschedparam()* function so that its second argument is of type **int**.

The *pthread\_getcpuclockid()* and *pthread\_mutex\_timedlock()* functions are added for alignment with IEEE Std 1003.1d-1999.

The following functions are added for alignment with IEEE Std 1003.1j-2000:

*pthread\_barrier\_destroy(), pthread\_barrier\_init(), pthread\_barrier\_wait(), pthread\_barrierattr\_destroy(), pthread\_barrierattr\_getpshared(), pthread\_barrierattr\_init(), pthread\_barrierattr\_setpshared(), pthread\_condattr\_getclock(), pthread\_condattr\_setclock(), pthread\_rwlock\_timedrdlock(), pthread\_rwlock\_timedwrlock(), pthread\_spin\_destroy(), pthread\_spin\_init(), pthread\_spin\_lock(), pthread\_spin\_trylock(), and pthread\_spin\_unlock()*.

PTHREAD\_RWLOCK\_INITIALIZER is removed for alignment with IEEE Std 1003.1j-2000.

Functions previously marked as part of the Read-Write Locks option are now moved to the Threads option.

The **restrict** keyword is added to the prototypes for *pthread\_attr\_getguardsize(), pthread\_attr\_getinheritsched(), pthread\_attr\_getschedparam(), pthread\_attr\_getschedpolicy(), pthread\_attr\_getscope(), pthread\_attr\_getstackaddr(), pthread\_attr\_getstacksize(), pthread\_attr\_setschedparam(), pthread\_barrier\_init(), pthread\_barrierattr\_getpshared(), pthread\_cond\_init(), pthread\_cond\_signal(), pthread\_cond\_timedwait(), pthread\_cond\_wait(), pthread\_condattr\_getclock(), pthread\_condattr\_getpshared(), pthread\_create(), pthread\_getschedparam(), pthread\_mutex\_getprioceiling(), pthread\_mutex\_init(), pthread\_mutex\_setprioceiling(), pthread\_mutexattr\_getprioceiling(), pthread\_mutexattr\_getprotocol(), pthread\_mutexattr\_getpshared(), pthread\_mutexattr\_gettype(), pthread\_rwlock\_init(), pthread\_rwlock\_timedrdlock(), pthread\_rwlock\_timedwrlock(), pthread\_rwlockattr\_getpshared(), and pthread\_sigmask()*.

IEEE PASC Interpretation 1003.1 #86 is applied, allowing the symbols from <sched.h> and <time.h> to be made visible when <pthread.h> is included. Previously this was an XSI option.

IEEE PASC Interpretation 1003.1c #42 is applied, removing the requirement for prototypes for the *pthread\_kill()* and *pthread\_sigmask()* functions. These are required to be in the <signal.h> header. They are allowed here through the name space rules.

IEEE PASC Interpretation 1003.1 #96 is applied, adding the *pthread\_setschedprio()* function.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/13 is applied, correcting shading errors that were in contradiction with the System Interfaces volume of POSIX.1-2008.

**Issue 7**

SD5-XBD-ERN-55 is applied, adding the **restrict** keyword to the *pthread\_mutex\_timedlock()* function prototype.

SD5-XBD-ERN-62 is applied.

Austin Group Interpretation 1003.1-2001 #048 is applied, reinstating the PTHREAD\_RWLOCK\_INITIALIZER symbol.

The <pthread.h> header is moved from the Threads option to the Base.

The following extended mutex types are moved from the XSI option to the Base:

```
PTHREAD_MUTEX_NORMAL
PTHREAD_MUTEX_ERRORCHECK
PTHREAD_MUTEX_RECURSIVE
PTHREAD_MUTEX_DEFAULT
```

The PTHREAD\_MUTEX\_ROBUST and PTHREAD\_MUTEX\_STALLED symbols and the *pthread\_mutex\_consistent()*, *pthread\_mutexattr\_getrobust()*, and *pthread\_mutexattr\_setrobust()* functions are added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Functionality relating to the Thread Priority Protection and Thread Priority Inheritance options is changed to be Non-Robust Mutex or Robust Mutex Priority Protection and Non-Robust Mutex or Robust Mutex Priority Inheritance, respectively.

This reference page is clarified with respect to macros and symbolic constants.

10895 **NAME**

10896       pwd.h — password structure

10897 **SYNOPSIS**

10898       #include &lt;pwd.h&gt;

10899 **DESCRIPTION**10900       The <pwd.h> header shall define the **struct passwd**, structure, which shall include at least the  
10901       following members:

10902	char	*pw_name	User's login name.
10903	uid_t	pw_uid	Numerical user ID.
10904	gid_t	pw_gid	Numerical group ID.
10905	char	*pw_dir	Initial working directory.
10906	char	*pw_shell	Program to use as shell.

10907       The <pwd.h> header shall define the **gid\_t**, **uid\_t**, and **size\_t** types as described in  
10908       <sys/types.h>.10909       The following shall be declared as functions and may also be defined as macros. Function  
10910       prototypes shall be provided.

10911	XSI	void	endpwent(void);
10912		struct passwd *	getpwent(void);
10913		struct passwd *	getpwnam(const char *);
10914		int	getpwnam_r(const char *, struct passwd *, char *,
10915			size_t, struct passwd **);
10916		struct passwd *	getpwuid(uid_t);
10917		int	getpwuid_r(uid_t, struct passwd *, char *,
10918			size_t, struct passwd **);
10919	XSI	void	setpwent(void);

10920 **APPLICATION USAGE**

10921       None.

10922 **RATIONALE**

10923       None.

10924 **FUTURE DIRECTIONS**

10925       None.

10926 **SEE ALSO**

10927       &lt;sys/types.h&gt;

10928       XSH *endpwent()*, *getpwnam()*, *getpwuid()*10929 **CHANGE HISTORY**

10930       First released in Issue 1.

10931 **Issue 5**

10932       The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

10933 **Issue 6**10934       The following new requirements on POSIX implementations derive from alignment with the  
10935       Single UNIX Specification:

- 10936
- The **gid\_t** and **uid\_t** types are mandated.

- 10937
- 10938
- The `getpwnam_r()` and `getpwuid_r()` functions are marked as part of the Thread-Safe Functions option.
- 10939 **Issue 7**
- 10940 SD5-XBD-ERN-56 is applied, adding a reference to `<sys/types.h>` for the `size_t` type.

10941 **NAME**

10942       regex.h — regular expression matching types

10943 **SYNOPSIS**

10944       #include &lt;regex.h&gt;

10945 **DESCRIPTION**10946       The <regex.h> header shall define the structures and symbolic constants used by the *regcomp()*,  
10947       *regexexec()*, *regerror()*, and *regfree()* functions.10948       The <regex.h> header shall define the **regex\_t** structure type, which shall include at least the  
10949       following member:

10950       size\_t       re\_nsub       Number of parenthesized subexpressions.

10951       The <regex.h> header shall define the **size\_t** type as described in <sys/types.h>.10952       The <regex.h> header shall define the **regoff\_t** type as a signed integer type that can hold the  
10953       largest value that can be stored in either a **ptrdiff\_t** type or a **ssize\_t** type.10954       The <regex.h> header shall define the **regmatch\_t** structure type, which shall include at least the  
10955       following members:

10956	regoff_t	rm_so	Byte offset from start of string
10957			to start of substring.
10958	regoff_t	rm_eo	Byte offset from start of string of the
10959			first character after the end of substring.

10960       The <regex.h> header shall define the following symbolic constants for the *cflags* parameter to  
10961       the *regcomp()* function:

10962	REG_EXTENDED	Use Extended Regular Expressions.
10963	REG_ICASE	Ignore case in match.
10964	REG_NOSUB	Report only success or fail in <i>regexexec()</i> .
10965	REG_NEWLINE	Change the handling of <newline>.

10966       The <regex.h> header shall define the following symbolic constants for the *eflags* parameter to  
10967       the *regexexec()* function:

10968	REG_NOTBOL	The <circumflex> character ('^'), when taken as a special character, does
10969		not match the beginning of <i>string</i> .
10970	REG_NOTEOL	The <dollar-sign> ('\$'), when taken as a special character, does not
10971		match the end of <i>string</i> .

10972       The &lt;regex.h&gt; header shall define the following symbolic constants as error return values:

10973	REG_NOMATCH	<i>regexexec()</i> failed to match.
10974	REG_BADPAT	Invalid regular expression.
10975	REG_ECOLLATE	Invalid collating element referenced.
10976	REG_ETYPE	Invalid character class type referenced.
10977	REG_EESCAPE	Trailing <backslash> character in pattern.
10978	REG_ESUBREG	Number in \digit invalid or in error.

10979	REG_EBRACK	" [ ] " imbalance.
10980	REG_EPAREN	" \ ( \ ) " or " ( ) " imbalance.
10981	REG_EBRACE	" \ { \ } " imbalance.
10982	REG_BADBR	Content of " \ { \ } " invalid: not a number, number too large, more than
10983		two numbers, first larger than second.
10984	REG_ERANGE	Invalid endpoint in range expression.
10985	REG_ESPACE	Out of memory.
10986	REG_BADRPT	' ? ', ' * ', or ' + ' not preceded by valid regular expression.
10987	The following shall be declared as functions and may also be defined as macros. Function	
10988	prototypes shall be provided.	
10989	<code>int regcomp(regex_t *restrict, const char *restrict, int);</code>	
10990	<code>size_t regerror(int, const regex_t *restrict, char *restrict, size_t);</code>	
10991	<code>int regexec(const regex_t *restrict, const char *restrict, size_t,</code>	
10992	<code>regmatch_t [restrict], int);</code>	
10993	<code>void regfree(regex_t *);</code>	
10994	The implementation may define additional macros or constants using names beginning with	
10995	REG_.	
10996	<b>APPLICATION USAGE</b>	
10997	None.	
10998	<b>RATIONALE</b>	
10999	None.	
11000	<b>FUTURE DIRECTIONS</b>	
11001	None.	
11002	<b>SEE ALSO</b>	
11003	<a href="#">&lt;sys/types.h&gt;</a>	
11004	XSH <a href="#">regcomp()</a>	
11005	<b>CHANGE HISTORY</b>	
11006	First released in Issue 4.	
11007	Originally derived from the ISO POSIX-2 standard.	
11008	<b>Issue 6</b>	
11009	The REG_ENOSYS constant is marked obsolescent.	
11010	The <b>restrict</b> keyword is added to the prototypes for <i>regcomp()</i> , <i>regerror()</i> , and <i>regexec()</i> .	
11011	A statement is added that the <b>size_t</b> type is defined as described in <a href="#">&lt;sys/types.h&gt;</a> .	
11012	<b>Issue 7</b>	
11013	SD5-XBD-ERN-60 is applied.	
11014	The obsolescent REG_ENOSYS constant is removed.	
11015	This reference page is clarified with respect to macros and symbolic constants.	

11016 **NAME**

11017 sched.h — execution scheduling

11018 **SYNOPSIS**

11019 #include &lt;sched.h&gt;

11020 **DESCRIPTION**11021 PS The <sched.h> header shall define the **pid\_t** type as described in <sys/types.h>.11022 SS|TSP The <sched.h> header shall define the **time\_t** type as described in <sys/types.h>.11023 The <sched.h> header shall define the **timespec** structure as described in <time.h>.

11024 The <sched.h> header shall define the **sched\_param** structure, which shall include the  
 11025 scheduling parameters required for implementation of each supported scheduling policy. This  
 11026 structure shall include at least the following member:

11027 int sched\_priority Process or thread execution scheduling priority.

11028 SS|TSP The **sched\_param** structure defined in <sched.h> shall include the following members in  
 11029 addition to those specified above:

11030 int sched\_ss\_low\_priority Low scheduling priority for  
 11031 sporadic server.

11032 struct timespec sched\_ss\_repl\_period Replenishment period for  
 11033 sporadic server.

11034 struct timespec sched\_ss\_init\_budget Initial budget for sporadic server.

11035 int sched\_ss\_max\_repl Maximum pending replenishments for  
 11036 sporadic server.

11037 Each process or thread is controlled by an associated scheduling policy and priority. Associated  
 11038 with each policy is a priority range. Each policy definition specifies the minimum priority range  
 11039 for that policy. The priority ranges for each policy may overlap the priority ranges of other  
 11040 policies.

11041 Four scheduling policies are defined; others may be defined by the implementation. The four  
 11042 standard policies are indicated by the values of the following symbolic constants:

11043 PS|TPS **SCHED\_FIFO** First in-first out (FIFO) scheduling policy.

11044 PS|TPS **SCHED\_RR** Round robin scheduling policy.

11045 SS|TSP **SCHED\_SPORADIC** Sporadic server scheduling policy.

11046 PS|TPS **SCHED\_OTHER** Another scheduling policy.

11047 The values of these constants are distinct.

11048 The following shall be declared as functions and may also be defined as macros. Function  
 11049 prototypes shall be provided.

11050 PS|TPS int sched\_get\_priority\_max(int);

11051 int sched\_get\_priority\_min(int);

11052 PS int sched\_getparam(pid\_t, struct sched\_param \*);

11053 int sched\_getscheduler(pid\_t);

11054 PS|TPS int sched\_rr\_get\_interval(pid\_t, struct timespec \*);

11055 PS int sched\_setparam(pid\_t, const struct sched\_param \*);

11056 int sched\_setscheduler(pid\_t, int, const struct sched\_param \*);

11057 int sched\_yield(void);



11058 Inclusion of the <sched.h> header may make visible all symbols from the <time.h> header.

11059 **APPLICATION USAGE**

11060 None.

11061 **RATIONALE**

11062 None.

11063 **FUTURE DIRECTIONS**

11064 None.

11065 **SEE ALSO**

11066 <sys/types.h>, <time.h>

11067 XSH *sched\_get\_priority\_max()*, *sched\_getparam()*, *sched\_getscheduler()*, *sched\_rr\_get\_interval()*,  
11068 *sched\_setparam()*, *sched\_setscheduler()*, *sched\_yield()*

11069 **CHANGE HISTORY**

11070 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

11071 **Issue 6**

11072 The <sched.h> header is marked as part of the Process Scheduling option.

11073 Sporadic server members are added to the **sched\_param** structure, and the SCHED\_SPORADIC  
11074 scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

11075 IEEE PASC Interpretation 1003.1 #108 is applied, correcting the **sched\_param** structure whose  
11076 members *sched\_ss\_repl\_period* and *sched\_ss\_init\_budget* should be type **struct timespec** and not  
11077 **timespec**.

11078 Symbols from <time.h> may be made visible when <sched.h> is included.

11079 IEEE Std 1003.1-2001/Cor 1-2002, items XSH/TC1/D6/52 and XSH/TC1/D6/53 are applied,  
11080 aligning the function prototype shading and margin codes with the System Interfaces volume of  
11081 IEEE Std 1003.1-2001.

11082 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/23 is applied, updating the  
11083 DESCRIPTION to differentiate between thread and process execution.

11084 **Issue 7**

11085 SD5-XBD-ERN-13 is applied.

11086 Austin Group Interpretation 1003.1-2001 #064 is applied, correcting the options markings.

11087 The <sched.h> headers is moved from the Threads option to the Base.

11088 Declarations for the **pid\_t** and **time\_t** types and the **timespec** structure are added.

11089 **NAME**

11090 search.h — search tables

11091 **SYNOPSIS**11092 XSI `#include <search.h>`11093 **DESCRIPTION**

11094 The <search.h> header shall define the **ENTRY** type for structure **entry** which shall include the  
 11095 following members:

```
11096 char    *key
11097 void     *data
```

11098 and shall define **ACTION** and **VISIT** as enumeration data types through type definitions as  
 11099 follows:

```
11100 enum { FIND, ENTER } ACTION;
11101 enum { preorder, postorder, endorder, leaf } VISIT;
```

11102 The <search.h> header shall define the **size\_t** type as described in <sys/types.h>.

11103 The following shall be declared as functions and may also be defined as macros. Function  
 11104 prototypes shall be provided.

```
11105 int      hcreate(size_t);
11106 void     hdestroy(void);
11107 ENTRY *hsearch(ENTRY, ACTION);
11108 void     insque(void *, void *);
11109 void     *lfind(const void *, const void *, size_t *,
11110               size_t, int (*)(const void *, const void *));
11111 void     *lsearch(const void *, void *, size_t *,
11112               size_t, int (*)(const void *, const void *));
11113 void     remque(void *);
11114 void     *tdelete(const void *restrict, void **restrict,
11115               int (*)(const void *, const void *));
11116 void     *tfind(const void *, void *const *,
11117               int (*)(const void *, const void *));
11118 void     *tsearch(const void *, void **,
11119               int (*)(const void *, const void *));
11120 void     twalk(const void *,
11121               void (*)(const void *, VISIT, int ));
```

11122 **APPLICATION USAGE**

11123 None.

11124 **RATIONALE**

11125 None.

11126 **FUTURE DIRECTIONS**

11127 None.

11128 **SEE ALSO**11129 [<sys/types.h>](#)11130 XSH [hcreate\(\)](#), [insque\(\)](#), [lsearch\(\)](#), [tdelete\(\)](#)

11131 **CHANGE HISTORY**

11132 First released in Issue 1. Derived from Issue 1 of the SVID.

11133 **Issue 6**

11134 The Open Group Corrigendum U021/6 is applied, updating the prototypes for *tdelete()* and  
11135 *tsearch()*.

11136 The **restrict** keyword is added to the prototype for *tdelete()*.

11137 **NAME**

11138 semaphore.h — semaphores

11139 **SYNOPSIS**

11140 #include &lt;semaphore.h&gt;

11141 **DESCRIPTION**

11142 The <semaphore.h> header shall define the **sem\_t** type, used in performing semaphore  
 11143 operations. The semaphore may be implemented using a file descriptor, in which case  
 11144 applications are able to open up at least a total of {OPEN\_MAX} files and semaphores.

11145 The <semaphore.h> header shall define the symbolic constant SEM\_FAILED which shall have  
 11146 type **sem\_t** \*.

11147 The following shall be declared as functions and may also be defined as macros. Function  
 11148 prototypes shall be provided.

```

11149 int    sem_close(sem_t *);
11150 int    sem_destroy(sem_t *);
11151 int    sem_getvalue(sem_t *restrict, int *restrict);
11152 int    sem_init(sem_t *, int, unsigned);
11153 sem_t *sem_open(const char *, int, ...);
11154 int    sem_post(sem_t *);
11155 int    sem_timedwait(sem_t *restrict, const struct timespec *restrict);
11156 int    sem_trywait(sem_t *);
11157 int    sem_unlink(const char *);
11158 int    sem_wait(sem_t *);

```

11159 Inclusion of the <semaphore.h> header may make visible symbols defined in the <fcntl.h> and  
 11160 <time.h> headers.

11161 **APPLICATION USAGE**

11162 None.

11163 **RATIONALE**

11164 None.

11165 **FUTURE DIRECTIONS**

11166 None.

11167 **SEE ALSO**

11168 &lt;fcntl.h&gt;, &lt;sys/types.h&gt;, &lt;time.h&gt;

11169 XSH *sem\_close()*, *sem\_destroy()*, *sem\_getvalue()*, *sem\_init()*, *sem\_open()*, *sem\_post()*,  
 11170 *sem\_timedwait()*, *sem\_trywait()*, *sem\_unlink()*

11171 **CHANGE HISTORY**

11172 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

11173 **Issue 6**

11174 The &lt;semaphore.h&gt; header is marked as part of the Semaphores option.

11175 The Open Group Corrigendum U021/3 is applied, adding a description of SEM\_FAILED.

11176 The *sem\_timedwait()* function is added for alignment with IEEE Std 1003.1d-1999.11177 The **restrict** keyword is added to the prototypes for *sem\_getvalue()* and *sem\_timedwait()*.

11178 **Issue 7**

11179 SD5-XBD-ERN-57 is applied, allowing the header to make visible symbols from the **<time.h>**  
11180 header.

11181 The **<semaphore.h>** header is moved from the Semaphores option to the Base.

11182 This reference page is clarified with respect to macros and symbolic constants.

11183 **NAME**

11184 setjmp.h — stack environment declarations

11185 **SYNOPSIS**

11186 #include &lt;setjmp.h&gt;

11187 **DESCRIPTION**

11188 CX Some of the functionality described on this reference page extends the ISO C standard.  
 11189 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to  
 11190 enable the visibility of these symbols in this header.

11191 CX The <setjmp.h> header shall define the array types **jmp\_buf** and **sigjmp\_buf**.

11192 The following shall be declared as functions and may also be defined as macros. Function  
 11193 prototypes shall be provided.

11194 OB XSI void \_longjmp(jmp\_buf, int);

11195 void longjmp(jmp\_buf, int);

11196 CX void siglongjmp(sigjmp\_buf, int);

11197 The following may be declared as functions, or defined as macros, or both. If functions are  
 11198 declared, function prototypes shall be provided.

11199 OB XSI int \_setjmp(jmp\_buf);

11200 int setjmp(jmp\_buf);

11201 CX int sigsetjmp(sigjmp\_buf, int);

11202 **APPLICATION USAGE**

11203 None.

11204 **RATIONALE**

11205 None.

11206 **FUTURE DIRECTIONS**

11207 None.

11208 **SEE ALSO**11209 XSH [Section 2.2](#) (on page 472), [\\_longjmp\(\)](#), [longjmp\(\)](#), [setjmp\(\)](#), [siglongjmp\(\)](#), [sigsetjmp\(\)](#)11210 **CHANGE HISTORY**

11211 First released in Issue 1.

11212 **Issue 6**

11213 Extensions beyond the ISO C standard are marked.

11214 **Issue 7**

11215 SD5-XBD-ERN-6 is applied.

11216 **NAME**

11217       signal.h — signals

11218 **SYNOPSIS**

11219       #include &lt;signal.h&gt;

11220 **DESCRIPTION**

11221 CX       Some of the functionality described on this reference page extends the ISO C standard.  
 11222       Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to  
 11223       enable the visibility of these symbols in this header.

11224       The <signal.h> header shall define the following macros, which shall expand to constant  
 11225       expressions with distinct values that have a type compatible with the second argument to, and  
 11226       the return value of, the *signal()* function, and whose values shall compare unequal to the  
 11227       address of any declarable function.

11228       SIG\_DFL               Request for default signal handling.

11229       SIG\_ERR               Return value from *signal()* in case of error.

11230 OB XSI   SIG\_HOLD       Request that signal be held.

11231       SIG\_IGN               Request that signal be ignored.

11232 CX       The <signal.h> header shall define the **pthread\_t**, **size\_t**, and **uid\_t** types as described in  
 11233       <sys/types.h>.

11234       The <signal.h> header shall define the **timespec** structure as described in <time.h>.

11235       The <signal.h> header shall define the following data types:

11236       **sig\_atomic\_t**       Possibly volatile-qualified integer type of an object that can be accessed as  
 11237       an atomic entity, even in the presence of asynchronous interrupts.

11238 CX       **sigset\_t**       Integer or structure type of an object used to represent sets of signals.

11239 CX       **pid\_t**           As described in <sys/types.h>.

11240 CX       The <signal.h> header shall define the **pthread\_attr\_t** type as described in <sys/types.h>.

11241       The <signal.h> header shall define the **sigevent** structure, which shall include at least the  
 11242       following members:

11243	int	sigev_notify	Notification type.
11244	int	sigev_signo	Signal number.
11245	union sigval	sigev_value	Signal value.
11246	void	(*sigev_notify_function)(union sigval)	Notification function.
11247			
11248	pthread_attr_t *	sigev_notify_attributes	Notification attributes.

11249       The <signal.h> header shall define the following symbolic constants for the values of  
 11250       *sigev\_notify*:

11251       SIGEV\_NONE       No asynchronous notification is delivered when the event of interest  
 11252       occurs.

11253       SIGEV\_SIGNAL      A queued signal, with an application-defined value, is generated when  
 11254       the event of interest occurs.

11255       SIGEV\_THREAD      A notification function is called to perform notification.

11256 The **signal** union shall be defined as:

11257     int     sival\_int     Integer signal value.

11258     void   \*sival\_ptr     Pointer signal value.

11259 The <**signal.h**> header shall declare the SIGRTMIN and SIGRTMAX macros, which shall expand

11260 to positive integer expressions with type **int**, but which need not be constant expressions. These

11261 macros specify a range of signal numbers that are reserved for application use and for which the

11262 realtime signal behavior specified in this volume of POSIX.1-2008 is supported. The signal

11263 numbers in this range do not overlap any of the signals specified in the following table.

11264 The range SIGRTMIN through SIGRTMAX inclusive shall include at least {RTSIG\_MAX} signal

11265 numbers.

11266 It is implementation-defined whether realtime signal behavior is supported for other signals.

11267 The <**signal.h**> header shall define the following macros that are used to refer to the signals that

11268 occur in the system. Signals defined here begin with the letters SIG followed by an uppercase

11269 letter. The macros shall expand to positive integer constant expressions with type **int** and

11270 distinct values. The value 0 is reserved for use as the null signal (see *kill()*). Additional

11271 implementation-defined signals may occur in the system.

11272 The ISO C standard only requires the signal names SIGABRT, SIGFPE, SIGILL, SIGINT, |

11273 SIGSEGV, and SIGTERM to be defined. An implementation need not generate any of these six |

11274 CX signals, except as a result of explicit use of interfaces that generate signals, such as *raise()*, *kill()*, |

11275 the General Terminal Interface (see [Section 11.1.9](#), on page 204), and the *kill* utility, unless |

11276 otherwise stated (see, for example, XSH [Section 2.8.3.3](#), on page 505).

11277 The following signals shall be supported on all implementations (default actions are explained

11278 below the table):



	Signal	Default Action	Description
11279			
11280	SIGABRT	A	Process abort signal.
11281	SIGALRM	T	Alarm clock.
11282	SIGBUS	A	Access to an undefined portion of a memory object.
11283	SIGCHLD	I	Child process terminated, stopped,
11284	XSI		or continued.
11285	SIGCONT	C	Continue executing, if stopped.
11286	SIGFPE	A	Erroneous arithmetic operation.
11287	SIGHUP	T	Hangup.
11288	SIGILL	A	Illegal instruction.
11289	SIGINT	T	Terminal interrupt signal.
11290	SIGKILL	T	Kill (cannot be caught or ignored).
11291	SIGPIPE	T	Write on a pipe with no one to read it.
11292	SIGQUIT	A	Terminal quit signal.
11293	SIGSEGV	A	Invalid memory reference.
11294	SIGSTOP	S	Stop executing (cannot be caught or ignored).
11295	SIGTERM	T	Termination signal.
11296	SIGTSTP	S	Terminal stop signal.
11297	SIGTTIN	S	Background process attempting read.
11298	SIGTTOU	S	Background process attempting write.
11299	SIGUSR1	T	User-defined signal 1.
11300	SIGUSR2	T	User-defined signal 2.
11301	OB XSR	T	Pollable event.
11302	OB XSI	T	Profiling timer expired.
11303	XSI	A	Bad system call.
11304		A	Trace/breakpoint trap.
11305	SIGURG	I	High bandwidth data is available at a socket.
11306	XSI	T	Virtual timer expired.
11307		A	CPU time limit exceeded.
11308		A	File size limit exceeded.

11309 The default actions are as follows:

- 11310 T Abnormal termination of the process.
- 11311 XSI A Abnormal termination of the process with additional actions.
- 11312 I Ignore the signal.
- 11313 S Stop the process.
- 11314 C Continue the process, if it is stopped; otherwise, ignore the signal.

11315 The effects on the process in each case are described in XSH [Section 2.4.3](#) (on page 490).

11316 CX The <signal.h> header shall declare the **sigaction** structure, which shall include at least the

11317 following members:

```

11318 void    (*sa_handler)(int)  Pointer to a signal-catching function
11319                                or one of the SIG_IGN or SIG_DFL.
11320 sigset_t sa_mask            Set of signals to be blocked during execution
11321                                of the signal handling function.
11322 int      sa_flags            Special flags.
11323 void    (*sa_sigaction)(int, siginfo_t *, void *)
11324                                Pointer to a signal-catching function.
```

11325 CX The storage occupied by *sa\_handler* and *sa\_sigaction* may overlap, and a conforming application  
11326 shall not use both simultaneously.

11327 The <signal.h> header shall define the following macros which shall expand to integer constant  
11328 expressions that need not be usable in **#if** preprocessing directives:

11329 CX **SIG\_BLOCK** The resulting set is the union of the current set and the signal set pointed  
11330 to by the argument *set*.

11331 CX **SIG\_UNBLOCK** The resulting set is the intersection of the current set and the complement  
11332 of the signal set pointed to by the argument *set*.

11333 CX **SIG\_SETMASK** The resulting set is the signal set pointed to by the argument *set*.

11334 The <signal.h> header shall also define the following symbolic constants:

11335 CX **SA\_NOCLDSTOP** Do not generate SIGCHLD when children stop  
11336 XSI or stopped children continue.

11337 XSI **SA\_ONSTACK** Causes signal delivery to occur on an alternate stack.

11338 CX **SA\_RESETHAND** Causes signal dispositions to be set to SIG\_DFL on entry to signal  
11339 handlers.

11340 CX **SA\_RESTART** Causes certain functions to become restartable.

11341 CX **SA\_SIGINFO** Causes extra information to be passed to signal handlers at the time of  
11342 receipt of a signal.

11343 XSI **SA\_NOCLDWAIT** Causes implementations not to create zombie processes or status  
11344 information on child termination. See *sigaction()*.

11345 CX **SA\_NODEFER** Causes signal not to be automatically blocked on entry to signal handler.

11346 XSI **SS\_ONSTACK** Process is executing on an alternate signal stack.

11347 XSI **SS\_DISABLE** Alternate signal stack is disabled.

11348 XSI **MINSIGSTKSZ** Minimum stack size for a signal handler.

11349 XSI **SIGSTKSZ** Default size in bytes for the alternate signal stack.

11350 CX The <signal.h> header shall define the **mcontext\_t** type through **typedef**.

11351 CX The <signal.h> header shall define the **ucontext\_t** type as a structure that shall include at least  
11352 the following members:

11353	<b>ucontext_t</b>	<b>*uc_link</b>	Pointer to the context that is resumed
11354			when this context returns.
11355	<b>sigset_t</b>	<b>uc_sigmask</b>	The set of signals that are blocked when this
11356			context is active.
11357	<b>stack_t</b>	<b>uc_stack</b>	The stack used by this context.
11358	<b>mcontext_t</b>	<b>uc_mcontext</b>	A machine-specific representation of the saved
11359			context.

11360 The <signal.h> header shall define the **stack\_t** type as a structure, which shall include at least  
11361 the following members:

11362	<b>void</b>	<b>*ss_sp</b>	Stack base or pointer.
11363	<b>size_t</b>	<b>ss_size</b>	Stack size.
11364	<b>int</b>	<b>ss_flags</b>	Flags.

11365	CX	The <signal.h> header shall define the <b>siginfo_t</b> type as a structure, which shall include at least		
11366		the following members:		
11367	CX	int	si_signo	Signal number.
11368		int	si_code	Signal code.
11369	XSI	int	si_errno	If non-zero, an <i>errno</i> value associated with
11370				this signal, as described in <errno.h>.
11371	CX	pid_t	si_pid	Sending process ID.
11372		uid_t	si_uid	Real user ID of sending process.
11373		void	*si_addr	Address of faulting instruction.
11374		int	si_status	Exit value or signal.
11375	OB XSR	long	si_band	Band event for SIGPOLL.
11376	CX	union sigval	si_value	Signal value.
11377	CX	The <signal.h> header shall define the symbolic constants in the <b>Code</b> column of the following		
11378		table for use as values of <i>si_code</i> that are signal-specific or non-signal-specific reasons why the		
11379		signal was generated.		

Signal	Code	Reason
CX	SIGILL	ILL_ILLOPC Illegal opcode. ILL_ILLOPN Illegal operand. ILL_ILLADR Illegal addressing mode. ILL_ILLTRP Illegal trap. ILL_PRVOPC Privileged opcode. ILL_PRVREG Privileged register. ILL_COPROC Coprocessor error. ILL_BADSTK Internal stack error.
	SIGFPE	FPE_INTDIV Integer divide by zero. FPE_INTOVF Integer overflow. FPE_FLTDIV Floating-point divide by zero. FPE_FLTOVF Floating-point overflow. FPE_FLTUND Floating-point underflow. FPE_FLTRES Floating-point inexact result. FPE_FLTINV Invalid floating-point operation. FPE_FLTSUB Subscript out of range.
	SIGSEGV	SEGV_MAPERR Address not mapped to object. SEGV_ACCERR Invalid permissions for mapped object.
	SIGBUS	BUS_ADRALN Invalid address alignment. BUS_ADRERR Nonexistent physical address. BUS_OBJERR Object-specific hardware error.
	SIGTRAP	TRAP_BRKPT Process breakpoint. TRAP_TRACE Process trace trap.
	SIGCHLD	CLD_EXITED Child has exited. CLD_KILLED Child has terminated abnormally and did not create a <b>core</b> file. CLD_DUMPED Child has terminated abnormally and created a <b>core</b> file. CLD_TRAPPED Traced child has trapped. CLD_STOPPED Child has stopped. CLD_CONTINUED Stopped child has continued.
	SIGPOLL	POLL_IN Data input available. POLL_OUT Output buffers available. POLL_MSG Input message available. POLL_ERR I/O error. POLL_PRI High priority input available. POLL_HUP Device disconnected.
	Any	SI_USER Signal sent by <i>kill()</i> . SI_QUEUE Signal sent by <i>sigqueue()</i> . SI_TIMER Signal generated by expiration of a timer set by <i>timer_settime()</i> . SI_ASYNCIO Signal generated by completion of an asynchronous I/O request. SI_MESGQ Signal generated by arrival of a message on an empty message queue
CX	Implementations may support additional <i>si_code</i> values not included in this list, may generate values included in this list under circumstances other than those described in this list, and may contain extensions or limitations that prevent some values from being generated. Implementations do not generate a different value from the ones described in this list for circumstances described in this list.	

11428 CX In addition, the following signal-specific information shall be available:

Signal	Member	Value
SIGILL SIGFPE	<b>void * <i>si_addr</i></b>	Address of faulting instruction.
SIGSEGV SIGBUS	<b>void * <i>si_addr</i></b>	Address of faulting memory reference.
SIGCHLD	<b>pid_t <i>si_pid</i></b> <b>int <i>si_status</i></b>  <b>uid_t <i>si_uid</i></b>	Child process ID. If <i>si_code</i> is equal to CLD_EXITED, then <i>si_status</i> holds the exit value of the process; otherwise, it is equal to the signal that caused the process to change state. The exit value in <i>si_status</i> shall be equal to the full exit value (that is, the value passed to <i>_exit()</i> , <i>_Exit()</i> , or <i>exit()</i> , or returned from <i>main()</i> ); it shall not be limited to the least significant eight bits of the value. Real user ID of the process that sent the signal.
SIGPOLL	<b>long <i>si_band</i></b>	Band event for POLL_IN, POLL_OUT, or POLL_MSG

11443 For some implementations, the value of *si\_addr* may be inaccurate.

11444 The following shall be declared as functions and may also be defined as macros. Function  
11445 prototypes shall be provided.

```

11446 CX int kill(pid_t, int);
11447 XSI int killpg(pid_t, int);
11448 CX void psiginfo(const siginfo_t *, const char *);
11449 void psignal(int, const char *);
11450 int pthread_kill(pthread_t, int);
11451 int pthread_sigmask(int, const sigset_t *restrict,
11452 sigset_t *restrict);
11453 int raise(int);
11454 CX int sigaction(int, const struct sigaction *restrict,
11455 struct sigaction *restrict);
11456 int sigaddset(sigset_t *, int);
11457 XSI int sigaltstack(const stack_t *restrict, stack_t *restrict);
11458 CX int sigdelset(sigset_t *, int);
11459 int sigemptyset(sigset_t *);
11460 int sigfillset(sigset_t *);
11461 OB XSI int sighold(int);
11462 int sigignore(int);
11463 int siginterrupt(int, int);
11464 CX int sigismember(const sigset_t *, int);
11465 void (*signal(int, void (*)(int)))(int);
11466 OB XSI int sigpause(int);
11467 CX int sigpending(sigset_t *);
11468 int sigprocmask(int, const sigset_t *restrict, sigset_t *restrict);
11469 int sigqueue(pid_t, int, union sigval);
11470 OB XSI int sigrelse(int);
11471 void (*sigset(int, void (*)(int)))(int);
11472 CX int sigsuspend(const sigset_t *);
11473 int sigtimedwait(const sigset_t *restrict, siginfo_t *restrict,
11474 const struct timespec *restrict);
11475 int sigwait(const sigset_t *restrict, int *restrict);
11476 int sigwaitinfo(const sigset_t *restrict, siginfo_t *restrict);

```

11477 CX Inclusion of the <signal.h> header may make visible all symbols from the <time.h> header.

## 11478 APPLICATION USAGE

11479 On systems not supporting the XSI option, the *si\_pid* and *si\_uid* members of **siginfo\_t** are only  
 11480 required to be valid when *si\_code* is SI\_USER or SI\_QUEUE. On XSI-conforming systems, they  
 11481 are also valid for all *si\_code* values less than or equal to 0; however, it is unspecified whether  
 11482 SI\_USER and SI\_QUEUE have values less than or equal to zero, and therefore XSI applications  
 11483 should check whether *si\_code* has the value SI\_USER or SI\_QUEUE or is less than or equal to 0 to  
 11484 tell whether *si\_pid* and *si\_uid* are valid.

## 11485 RATIONALE

11486 None.

## 11487 FUTURE DIRECTIONS

11488 The SIGPOLL and SIGPROF signals may be removed in a future version.

## 11489 SEE ALSO

11490 <errno.h>, <stropts.h>, <sys/types.h>, <time.h>

11491 XSH Section 2.2 (on page 472), *alarm()*, *ioctl()*, *kill()*, *killpg()*, *psiginfo()*, *pthread\_kill()*,  
 11492 *pthread\_sigmask()*, *raise()*, *sigaction()*, *sigaddset()*, *sigaltstack()*, *sigdelset()*, *sigemptyset()*,  
 11493 *sigfillset()*, *sighold()*, *siginterrupt()*, *sigismember()*, *signal()*, *sigpending()*, *sigqueue()*, *sigsuspend()*,  
 11494 *sigtimedwait()*, *sigwait()*, *timer\_create()*, *wait()*, *waitid()*

11495 XCU *kill* +

## 11496 CHANGE HISTORY

11497 First released in Issue 1.

### 11498 Issue 5

11499 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX  
 11500 Threads Extension.

11501 The default action for SIGURG is changed from i to iii. The function prototype for *sigmask()* is  
 11502 removed.

### 11503 Issue 6

11504 The Open Group Corrigendum U035/2 is applied. In the DESCRIPTION, the wording for  
 11505 abnormal termination is clarified.

11506 The Open Group Corrigendum U028/8 is applied, correcting the prototype for the *sigset()*  
 11507 function.

11508 The Open Group Corrigendum U026/3 is applied, correcting the type of the *sigev\_notify\_function*  
 11509 function member of the **sigevent** structure.

11510 The following new requirements on POSIX implementations derive from alignment with the  
 11511 Single UNIX Specification:

- 11512 • The SIGCHLD, SIGCONT, SIGSTOP, SIGTSTP, SIGTTIN, and SIGTTOU signals are now  
 11513 mandated. This is also a FIPS requirement.
- 11514 • The **pid\_t** definition is mandated.

11515 The RT markings are changed to RTS to denote that the semantics are part of the Realtime  
 11516 Signals Extension option.

11517 The **restrict** keyword is added to the prototypes for *sigaction()*, *sigaltstack()*, *sigprocmask()*,  
 11518 *sigtimedwait()*, *sigwait()*, and *sigwaitinfo()*.

11519 IEEE PASC Interpretation 1003.1 #85 is applied, adding the statement that symbols from  
 11520 <time.h> may be made visible when <signal.h> is included.

11521 Extensions beyond the ISO C standard are marked.

11522 IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/14 is applied, changing the descriptive  
 11523 text for members of the **sigaction** structure.

11524 IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/15 is applied, correcting the definition of  
 11525 the *sa\_sigaction* member of the **sigaction** structure.

11526 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/24 is applied, reworking the ordering of  
 11527 the **siginfo\_t** type structure in the DESCRIPTION. This is an editorial change and no normative  
 11528 change is intended.

11529 **Issue 7**

11530 SD5-XBD-ERN-5 is applied.

11531 SD5-XBD-ERN-39 is applied, removing the **sigstack** structure which should have been removed  
 11532 at the same time as the LEGACY *sigstack()* function.

11533 SD5-XBD-ERN-56 is applied, adding a reference to <sys/types.h> for the **size\_t** type.

11534 Austin Group Interpretation 1003.1-2001 #034 is applied.

11535 The **ucontext\_t** and **mcontext\_t** structures are added here from the obsolescent <ucontext.h>  
 11536 header.

11537 The *psiginfo()* and *psignal()* functions are added from The Open Group Technical Standard, 2006,  
 11538 Extended API Set Part 1.

11539 The SIGPOLL and SIGPROF signals and text relating to the XSI STREAMS option are marked  
 11540 obsolescent.

11541 The SA\_RESETHAND, SA\_RESTART, SA\_SIGINFO, SA\_NOCLDWAIT, and SA\_NODEFER  
 11542 constants are moved from the XSI option to the Base.

11543 Functionality relating to the Realtime Signals Extension option is moved to the Base.

11544 This reference page is clarified with respect to macros and symbolic constants, and declarations  
 11545 for the **pthread\_attr\_t**, **pthread\_t**, and **uid\_t** types and the **timespec** structure are added.

11546 SIGRTMIN and SIGRTMAX are required to be positive integer expressions.

11547 The APPLICATION USAGE section is updated to describe the *si\_pid* and *si\_uid* members of  
 11548 **siginfo\_t**.

11549 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0062 [208], XBD/TC1-2008/0063 [80],  
 11550 and XBD/TC1-2008/0064 [157] are applied.



## 11551 NAME

11552 spawn.h — spawn (ADVANCED REALTIME)

## 11553 SYNOPSIS

11554 SPN `#include <spawn.h>`

## 11555 DESCRIPTION

11556 The <spawn.h> header shall define the **posix\_spawnattr\_t** and **posix\_spawn\_file\_actions\_t**  
 11557 types used in performing spawn operations.

11558 The <spawn.h> header shall define the **mode\_t** and **pid\_t** types as described in <sys/types.h>.

11559 The <spawn.h> header shall define the **sigset\_t** type as described in <signal.h>.

11560 The tag **sched\_param** shall be declared as naming an incomplete structure type, the contents of  
 11561 which are described in the <sched.h> header.

11562 The <spawn.h> header shall define the following symbolic constants for use as the flags that  
 11563 may be set in a **posix\_spawnattr\_t** object using the *posix\_spawnattr\_setflags()* function:

11564 POSIX\_SPAWN\_RESETIDS  
 11565 POSIX\_SPAWN\_SETPGROUP  
 11566 PS POSIX\_SPAWN\_SETSCHEDPARAM  
 11567 POSIX\_SPAWN\_SETSCHEDULER  
 11568 POSIX\_SPAWN\_SETSIGDEF  
 11569 POSIX\_SPAWN\_SETSIGMASK

11570 The following shall be declared as functions and may also be defined as macros. Function  
 11571 prototypes shall be provided.

```

11572 int    posix_spawn(pid_t *restrict, const char *restrict,
11573                  const posix_spawn_file_actions_t *,
11574                  const posix_spawnattr_t *restrict, char *const [restrict],
11575                  char *const [restrict]);
11576 int    posix_spawn_file_actions_addclose(posix_spawn_file_actions_t *,
11577                  int);
11578 int    posix_spawn_file_actions_adddup2(posix_spawn_file_actions_t *,
11579                  int, int);
11580 int    posix_spawn_file_actions_addopen(posix_spawn_file_actions_t *restrict,
11581                  int, const char *restrict, int, mode_t);
11582 int    posix_spawn_file_actions_destroy(posix_spawn_file_actions_t *);
11583 int    posix_spawn_file_actions_init(posix_spawn_file_actions_t *);
11584 int    posix_spawnattr_destroy(posix_spawnattr_t *);
11585 int    posix_spawnattr_getflags(const posix_spawnattr_t *restrict,
11586                  short *restrict);
11587 int    posix_spawnattr_getpgroup(const posix_spawnattr_t *restrict,
11588                  pid_t *restrict);
11589 PS int    posix_spawnattr_getschedparam(const posix_spawnattr_t *restrict,
11590                  struct sched_param *restrict);
11591 int    posix_spawnattr_getschedpolicy(const posix_spawnattr_t *restrict,
11592                  int *restrict);
11593 int    posix_spawnattr_getsigdefault(const posix_spawnattr_t *restrict,
11594                  sigset_t *restrict);
11595 int    posix_spawnattr_getsigmask(const posix_spawnattr_t *restrict,
11596                  sigset_t *restrict);
11597 int    posix_spawnattr_init(posix_spawnattr_t *);

```



```

11598     int    posix_spawnattr_setflags(posix_spawnattr_t *, short);
11599     int    posix_spawnattr_setpgroup(posix_spawnattr_t *, pid_t);
11600 PS     int    posix_spawnattr_setschedparam(posix_spawnattr_t *restrict,
11601         const struct sched_param *restrict);
11602     int    posix_spawnattr_setschedpolicy(posix_spawnattr_t *, int);
11603     int    posix_spawnattr_setsigdefault(posix_spawnattr_t *restrict,
11604         const sigset_t *restrict);
11605     int    posix_spawnattr_setsigmask(posix_spawnattr_t *restrict,
11606         const sigset_t *restrict);
11607     int    posix_spawn(pid_t *restrict, const char *restrict,
11608         const posix_spawn_file_actions_t *,
11609         const posix_spawnattr_t *restrict,
11610         char *const [restrict], char *const [restrict]);
11611
11612     Inclusion of the <spawn.h> header may make visible symbols defined in the <sched.h> and
     <signal.h> headers.

```

#### 11613 APPLICATION USAGE

11614 None.

#### 11615 RATIONALE

11616 None.

#### 11617 FUTURE DIRECTIONS

11618 None.

#### 11619 SEE ALSO

11620 [<sched.h>](#), [<semaphore.h>](#), [<signal.h>](#), [<sys/types.h>](#)

11621 XSH [posix\\_spawn\(\)](#), [posix\\_spawn\\_file\\_actions\\_addclose\(\)](#), [posix\\_spawn\\_file\\_actions\\_adddup2\(\)](#),  
11622 [posix\\_spawn\\_file\\_actions\\_destroy\(\)](#), [posix\\_spawnattr\\_destroy\(\)](#), [posix\\_spawnattr\\_getflags\(\)](#),  
11623 [posix\\_spawnattr\\_getpgroup\(\)](#), [posix\\_spawnattr\\_getschedparam\(\)](#), [posix\\_spawnattr\\_getschedpolicy\(\)](#),  
11624 [posix\\_spawnattr\\_getsigdefault\(\)](#), [posix\\_spawnattr\\_getsigmask\(\)](#)

#### 11625 CHANGE HISTORY

11626 First released in Issue 6. Included for alignment with IEEE Std 1003.1d-1999.

11627 The **restrict** keyword is added to the prototypes for [posix\\_spawn\(\)](#),  
11628 [posix\\_spawn\\_file\\_actions\\_addopen\(\)](#), [posix\\_spawnattr\\_getsigdefault\(\)](#), [posix\\_spawnattr\\_getflags\(\)](#),  
11629 [posix\\_spawnattr\\_getpgroup\(\)](#), [posix\\_spawnattr\\_getschedparam\(\)](#), [posix\\_spawnattr\\_getschedpolicy\(\)](#),  
11630 [posix\\_spawnattr\\_getsigmask\(\)](#), [posix\\_spawnattr\\_setsigdefault\(\)](#), [posix\\_spawnattr\\_setschedparam\(\)](#),  
11631 [posix\\_spawnattr\\_setsigmask\(\)](#), and [posix\\_spawnnp\(\)](#).

#### 11632 Issue 7

11633 This reference page is clarified with respect to macros and symbolic constants, and declarations  
11634 for the **mode\_t**, **pid\_t**, and **sigset\_t** types are added.

## 11635 NAME

11636        `stdarg.h` — handle variable argument list

## 11637 SYNOPSIS

```
11638        #include <stdarg.h>
11639
11639        void va_start(va_list ap, argN);
11640        void va_copy(va_list dest, va_list src);
11641        type va_arg(va_list ap, type);
11642        void va_end(va_list ap);
```

## 11643 DESCRIPTION

11644 CX        The functionality described on this reference page is aligned with the ISO C standard. Any  
 11645        conflict between the requirements described here and the ISO C standard is unintentional. This  
 11646        volume of POSIX.1-2008 defers to the ISO C standard.

11647        The **<stdarg.h>** header shall contain a set of macros which allows portable functions that accept  
 11648        variable argument lists to be written. Functions that have variable argument lists (such as  
 11649        `printf()`) but do not use these macros are inherently non-portable, as different systems use  
 11650        different argument-passing conventions.

11651        The **<stdarg.h>** header shall define the **va\_list** type for variables used to traverse the list.

11652        The `va_start()` macro is invoked to initialize *ap* to the beginning of the list before any calls to  
 11653        `va_arg()`.

11654        The `va_copy()` macro initializes *dest* as a copy of *src*, as if the `va_start()` macro had been applied  
 11655        to *dest* followed by the same sequence of uses of the `va_arg()` macro as had previously been used  
 11656        to reach the present state of *src*. Neither the `va_copy()` nor `va_start()` macro shall be invoked to  
 11657        reinitialize *dest* without an intervening invocation of the `va_end()` macro for the same *dest*.

11658        The object *ap* may be passed as an argument to another function; if that function invokes the  
 11659        `va_arg()` macro with parameter *ap*, the value of *ap* in the calling function is unspecified and shall  
 11660        be passed to the `va_end()` macro prior to any further reference to *ap*. The parameter *argN* is the  
 11661        identifier of the rightmost parameter in the variable parameter list in the function definition (the  
 11662        one just before the `...`). If the parameter *argN* is declared with the **register** storage class, with a  
 11663        function type or array type, or with a type that is not compatible with the type that results after  
 11664        application of the default argument promotions, the behavior is undefined.

11665        The `va_arg()` macro shall return the next argument in the list pointed to by *ap*. Each invocation  
 11666        of `va_arg()` modifies *ap* so that the values of successive arguments are returned in turn. The *type*  
 11667        parameter shall be a type name specified such that the type of a pointer to an object that has the  
 11668        specified type can be obtained simply by postfixing a `'*'` to type. If there is no actual next  
 11669        argument, or if *type* is not compatible with the type of the actual next argument (as promoted  
 11670        according to the default argument promotions), the behavior is undefined, except for the  
 11671        following cases:

- 11672            • One type is a signed integer type, the other type is the corresponding unsigned integer  
 11673            type, and the value is representable in both types.
- 11674            • One type is a pointer to **void** and the other is a pointer to a character type.
- 11675 XSI        • Both types are pointers.

11676        Different types can be mixed, but it is up to the routine to know what type of argument is  
 11677        expected.

11678        The `va_end()` macro is used to clean up; it invalidates *ap* for use (unless `va_start()` or `va_copy()` is  
 11679        invoked again).

Each invocation of the *va\_start()* and *va\_copy()* macros shall be matched by a corresponding invocation of the *va\_end()* macro in the same function.

Multiple traversals, each bracketed by *va\_start()* ... *va\_end()*, are possible.

#### EXAMPLES

This example is a possible implementation of *execl()*:

```
#include <stdarg.h>

#define MAXARGS 31

/*
 * execl is called by
 * execl(file, arg1, arg2, ..., (char *) (0));
 */
int execl(const char *file, const char *args, ...)
{
    va_list ap;
    char *array[MAXARGS + 1];
    int argno = 0;

    va_start(ap, args);
    while (args != 0 && argno < MAXARGS)
    {
        array[argno++] = args;
        args = va_arg(ap, const char *);
    }
    array[argno] = (char *) 0;
    va_end(ap);
    return execv(file, array);
}
```

#### APPLICATION USAGE

It is up to the calling routine to communicate to the called routine how many arguments there are, since it is not always possible for the called routine to determine this in any other way. For example, *execl()* is passed a null pointer to signal the end of the list. The *printf()* function can tell how many arguments are there by the *format* argument.

#### RATIONALE

None.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

XSH *exec*, *fprintf()*

#### CHANGE HISTORY

First released in Issue 4. Derived from the ANSI C standard.

#### Issue 6

This reference page is updated to align with the ISO/IEC 9899:1999 standard.

11721 **NAME**

11722       stdbool.h — boolean type and values

11723 **SYNOPSIS**

11724       #include &lt;stdbool.h&gt;

11725 **DESCRIPTION**

11726 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
11727       conflict between the requirements described here and the ISO C standard is unintentional. This  
11728       volume of POSIX.1-2008 defers to the ISO C standard.

11729       The &lt;stdbool.h&gt; header shall define the following macros:

11730       bool       Expands to **\_Bool**.

11731       true       Expands to the integer constant 1.

11732       false       Expands to the integer constant 0.

11733       \_\_bool\_true\_false\_are\_defined

11734       Expands to the integer constant 1.

11735       An application may undefine and then possibly redefine the macros bool, true, and false.

11736 **APPLICATION USAGE**

11737       None.

11738 **RATIONALE**

11739       None.

11740 **FUTURE DIRECTIONS**

11741       The ability to undefine and redefine the macros bool, true, and false is an obsolescent feature  
11742       and may be removed in a future version.

11743 **SEE ALSO**

11744       None.

11745 **CHANGE HISTORY**

11746       First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

11747 **NAME**11748        `stddef.h` — standard type definitions11749 **SYNOPSIS**11750        `#include <stddef.h>`11751 **DESCRIPTION**

11752 CX        The functionality described on this reference page is aligned with the ISO C standard. Any  
 11753            conflict between the requirements described here and the ISO C standard is unintentional. This  
 11754            volume of POSIX.1-2008 defers to the ISO C standard.

11755        The `<stddef.h>` header shall define the following macros:

11756 CX        **NULL**        Null pointer constant. The macro shall expand to an integer constant expression  
 11757            with the value 0 cast to type `void *`.

11758        `offsetof(type, member-designator)`

11759            Integer constant expression of type `size_t`, the value of which is the offset in bytes  
 11760            to the structure member (*member-designator*), from the beginning of its structure  
 11761            (*type*).

11762        The `<stddef.h>` header shall define the following types:11763        **ptrdiff\_t**        Signed integer type of the result of subtracting two pointers.

11764        **wchar\_t**        Integer type whose range of values can represent distinct codes for all members of  
 11765            the largest extended character set specified among the supported locales; the null  
 11766            character shall have the code value zero. Each member of the basic character set  
 11767            shall have a code value equal to its value when used as the lone character in an  
 11768            integer character constant if an implementation does not define  
 11769            `__STDC_MB_MIGHT_NEQ_WC__`.

11770        **size\_t**        Unsigned integer type of the result of the `sizeof` operator.

11771        The implementation shall support one or more programming environments in which the widths  
 11772        of `ptrdiff_t`, `size_t`, and `wchar_t` are no greater than the width of type `long`. The names of these  
 11773        programming environments can be obtained using the `confstr()` function or the `getconf` utility.

11774 **APPLICATION USAGE**

11775        None.

11776 **RATIONALE**

11777        The ISO C standard does not require the `NULL` macro to include the cast to type `void *` and  
 11778        specifies that the `NULL` macro be implementation-defined. POSIX.1-2008 requires the cast and  
 11779        therefore need not be implementation-defined.

11780 **FUTURE DIRECTIONS**

11781        None.

11782 **SEE ALSO**11783        `<sys/types.h>`, `<wchar.h>`11784        XSH `confstr()`11785        XCU `getconf`11786 **CHANGE HISTORY**

11787        First released in Issue 4. Derived from the ANSI C standard.

11788 **Issue 7**

11789 This reference page is clarified with respect to macros and symbolic constants.

11790 SD5-XBD-ERN-53 is applied, updating the definition of **wchar\_t** to align with  
11791 ISO/IEC 9899:1999 standard, Technical Corrigendum 3.

11792 **NAME**

11793       stdint.h — integer types

11794 **SYNOPSIS**

11795       #include <stdint.h>

11796 **DESCRIPTION**

11797 CX       Some of the functionality described on this reference page extends the ISO C standard.  
11798       Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to  
11799       enable the visibility of these symbols in this header.

11800       The <stdint.h> header shall declare sets of integer types having specified widths, and shall  
11801       define corresponding sets of macros. It shall also define macros that specify limits of integer  
11802       types corresponding to types defined in other standard headers.

11803       **Note:**     The “width” of an integer type is the number of bits used to store its value in a pure binary  
11804       system; the actual type may use more bits than that (for example, a 28-bit type could be stored  
11805       in 32 bits of actual storage). An  $N$ -bit signed type has values in the range  $-2^{N-1}$  or  $1-2^{N-1}$  to  
11806        $2^{N-1}-1$ , while an  $N$ -bit unsigned type has values in the range 0 to  $2^N-1$ .

11807       Types are defined in the following categories:

- 11808       • Integer types having certain exact widths
- 11809       • Integer types having at least certain specified widths
- 11810       • Fastest integer types having at least certain specified widths
- 11811       • Integer types wide enough to hold pointers to objects
- 11812       • Integer types having greatest width

11813       (Some of these types may denote the same type.)

11814       Corresponding macros specify limits of the declared types and construct suitable constants.

11815       For each type described herein that the implementation provides, the <stdint.h> header shall  
11816       declare that **typedef** name and define the associated macros. Conversely, for each type described  
11817       herein that the implementation does not provide, the <stdint.h> header shall not declare that  
11818       **typedef** name, nor shall it define the associated macros. An implementation shall provide those  
11819       types described as required, but need not provide any of the others (described as optional).

11820 **Integer Types**

11821       When **typedef** names differing only in the absence or presence of the initial  $u$  are defined, they  
11822       shall denote corresponding signed and unsigned types as described in the ISO/IEC 9899:1999  
11823       standard, Section 6.2.5; an implementation providing one of these corresponding types shall also  
11824       provide the other.

11825       In the following descriptions, the symbol  $N$  represents an unsigned decimal integer with no  
11826       leading zeros (for example, 8 or 24, but not 04 or 048).

- 11827       • Exact-width integer types

11828       The **typedef** name **int $N$ \_t** designates a signed integer type with width  $N$ , no padding bits,  
11829       and a two’s-complement representation. Thus, **int8\_t** denotes a signed integer type with a  
11830       width of exactly 8 bits.

11831       The **typedef** name **uint $N$ \_t** designates an unsigned integer type with width  $N$ . Thus,  
11832       **uint24\_t** denotes an unsigned integer type with a width of exactly 24 bits.

11833 CX The following types are required:

11834 **int8\_t**  
 11835 **int16\_t**  
 11836 **int32\_t**  
 11837 **uint8\_t**  
 11838 **uint16\_t**  
 11839 **uint32\_t**

11840 If an implementation provides integer types with width 64 that meet these requirements,  
 11841 then the following types are required:

11842 **int64\_t**  
 11843 **uint64\_t**

11844 CX In particular, this will be the case if any of the following are true:

- 11845 — The implementation supports the `_POSIX_V7_ILP32_OFFBIG` programming  
 11846 environment and the application is being built in the `_POSIX_V7_ILP32_OFFBIG`  
 11847 programming environment (see the Shell and Utilities volume of POSIX.1-2008, c99,  
 11848 Programming Environments).
- 11849 — The implementation supports the `_POSIX_V7_LP64_OFF64` programming  
 11850 environment and the application is being built in the `_POSIX_V7_LP64_OFF64`  
 11851 programming environment.
- 11852 — The implementation supports the `_POSIX_V7_LPBIG_OFFBIG` programming  
 11853 environment and the application is being built in the `_POSIX_V7_LPBIG_OFFBIG`  
 11854 programming environment.

11855 All other types of this form are optional.

11856 • Minimum-width integer types

11857 The **typedef** name **int\_leastN\_t** designates a signed integer type with a width of at least *N*,  
 11858 such that no signed integer type with lesser size has at least the specified width. Thus,  
 11859 **int\_least32\_t** denotes a signed integer type with a width of at least 32 bits.

11860 The **typedef** name **uint\_leastN\_t** designates an unsigned integer type with a width of at  
 11861 least *N*, such that no unsigned integer type with lesser size has at least the specified width.  
 11862 Thus, **uint\_least16\_t** denotes an unsigned integer type with a width of at least 16 bits.

11863 The following types are required:

11864 **int\_least8\_t**  
 11865 **int\_least16\_t**  
 11866 **int\_least32\_t**  
 11867 **int\_least64\_t**  
 11868 **uint\_least8\_t**  
 11869 **uint\_least16\_t**  
 11870 **uint\_least32\_t**  
 11871 **uint\_least64\_t**

11872 All other types of this form are optional.



- 11873
- Fastest minimum-width integer types
- 11874 Each of the following types designates an integer type that is usually fastest to operate  
11875 with among all integer types that have at least the specified width.
- 11876 The designated type is not guaranteed to be fastest for all purposes; if the implementation  
11877 has no clear grounds for choosing one type over another, it will simply pick some integer  
11878 type satisfying the signedness and width requirements.
- 11879 The **typedef** name **int\_fastN\_t** designates the fastest signed integer type with a width of at  
11880 least *N*. The **typedef** name **uint\_fastN\_t** designates the fastest unsigned integer type with  
11881 a width of at least *N*.
- 11882 The following types are required:
- 11883 **int\_fast8\_t**  
11884 **int\_fast16\_t**  
11885 **int\_fast32\_t**  
11886 **int\_fast64\_t**  
11887 **uint\_fast8\_t**  
11888 **uint\_fast16\_t**  
11889 **uint\_fast32\_t**  
11890 **uint\_fast64\_t**
- 11891 All other types of this form are optional.
- 11892
- Integer types capable of holding object pointers
- 11893 The following type designates a signed integer type with the property that any valid  
11894 pointer to **void** can be converted to this type, then converted back to a pointer to **void**, and  
11895 the result will compare equal to the original pointer:
- 11896 **intptr\_t**
- 11897 The following type designates an unsigned integer type with the property that any valid  
11898 pointer to **void** can be converted to this type, then converted back to a pointer to **void**, and  
11899 the result will compare equal to the original pointer:
- 11900 **uintptr\_t**
- 11901 XSI On XSI-conformant systems, the **intptr\_t** and **uintptr\_t** types are required; otherwise, they  
11902 are optional.
- 11903
- Greatest-width integer types
- 11904 The following type designates a signed integer type capable of representing any value of  
11905 any signed integer type:
- 11906 **intmax\_t**
- 11907 The following type designates an unsigned integer type capable of representing any value  
11908 of any unsigned integer type:
- 11909 **uintmax\_t**
- 11910 These types are required.

11911 **Note:** Applications can test for optional types by using the corresponding limit macro from [Limits of](#)  
 11912 [Specified-Width Integer Types](#).

### 11913 **Limits of Specified-Width Integer Types**

11914 The following macros specify the minimum and maximum limits of the types declared in the  
 11915 **<stdint.h>** header. Each macro name corresponds to a similar type name in [Integer Types](#) (on  
 11916 page 347).

11917 Each instance of any defined macro shall be replaced by a constant expression suitable for use in  
 11918 **#if** preprocessing directives, and this expression shall have the same type as would an  
 11919 expression that is an object of the corresponding type converted according to the integer  
 11920 promotions. Its implementation-defined value shall be equal to or greater in magnitude  
 11921 (absolute value) than the corresponding value given below, with the same sign, except where  
 11922 stated to be exactly the given value.

- 11923 • Limits of exact-width integer types
  - 11924 — Minimum values of exact-width signed integer types:
    - 11925 {INTN\_MIN} Exactly  $-(2^{N-1})$
  - 11926 — Maximum values of exact-width signed integer types:
    - 11927 {INTN\_MAX} Exactly  $2^{N-1} - 1$
  - 11928 — Maximum values of exact-width unsigned integer types:
    - 11929 {UINTN\_MAX} Exactly  $2^N - 1$
- 11930 • Limits of minimum-width integer types
  - 11931 — Minimum values of minimum-width signed integer types:
    - 11932 {INT\_LEASTN\_MIN}  $-(2^{N-1} - 1)$
  - 11933 — Maximum values of minimum-width signed integer types:
    - 11934 {INT\_LEASTN\_MAX}  $2^{N-1} - 1$
  - 11935 — Maximum values of minimum-width unsigned integer types:
    - 11936 {UINT\_LEASTN\_MAX}  $2^N - 1$
- 11937 • Limits of fastest minimum-width integer types
  - 11938 — Minimum values of fastest minimum-width signed integer types:
    - 11939 {INT\_FASTN\_MIN}  $-(2^{N-1} - 1)$
  - 11940 — Maximum values of fastest minimum-width signed integer types:
    - 11941 {INT\_FASTN\_MAX}  $2^{N-1} - 1$
  - 11942 — Maximum values of fastest minimum-width unsigned integer types:
    - 11943 {UINT\_FASTN\_MAX}  $2^N - 1$
- 11944 • Limits of integer types capable of holding object pointers
  - 11945 — Minimum value of pointer-holding signed integer type:
    - 11946 {INTPTR\_MIN}  $-(2^{15} - 1)$

- 11947 — Maximum value of pointer-holding signed integer type:
- 11948 {INTPTR\_MAX}  $2^{15} - 1$
- 11949 — Maximum value of pointer-holding unsigned integer type:
- 11950 {UINTPTR\_MAX}  $2^{16} - 1$
- 11951 • Limits of greatest-width integer types
- 11952 — Minimum value of greatest-width signed integer type:
- 11953 {INTMAX\_MIN}  $-(2^{63} - 1)$
- 11954 — Maximum value of greatest-width signed integer type:
- 11955 {INTMAX\_MAX}  $2^{63} - 1$
- 11956 — Maximum value of greatest-width unsigned integer type:
- 11957 {UINTMAX\_MAX}  $2^{64} - 1$

### Limits of Other Integer Types

The following macros specify the minimum and maximum limits of integer types corresponding to types defined in other standard headers.

Each instance of these macros shall be replaced by a constant expression suitable for use in `#if` preprocessing directives, and this expression shall have the same type as would an expression that is an object of the corresponding type converted according to the integer promotions. Its implementation-defined value shall be equal to or greater in magnitude (absolute value) than the corresponding value given below, with the same sign.

- 11966 • Limits of **ptrdiff\_t**:
- 11967 {PTRDIFF\_MIN}  $-65\,535$
- 11968 {PTRDIFF\_MAX}  $+65\,535$
- 11969 • Limits of **sig\_atomic\_t**:
- 11970 {SIG\_ATOMIC\_MIN} See below.
- 11971 {SIG\_ATOMIC\_MAX} See below.
- 11972 • Limit of **size\_t**:
- 11973 {SIZE\_MAX}  $65\,535$
- 11974 • Limits of **wchar\_t**:
- 11975 {WCHAR\_MIN} See below.
- 11976 {WCHAR\_MAX} See below.
- 11977 • Limits of **wint\_t**:
- 11978 {WINT\_MIN} See below.
- 11979 {WINT\_MAX} See below.

If **sig\_atomic\_t** (see the `<signal.h>` header) is defined as a signed integer type, the value of {SIG\_ATOMIC\_MIN} shall be no greater than  $-127$  and the value of {SIG\_ATOMIC\_MAX} shall be no less than  $127$ ; otherwise, **sig\_atomic\_t** shall be defined as an unsigned integer type, and the value of {SIG\_ATOMIC\_MIN} shall be  $0$  and the value of {SIG\_ATOMIC\_MAX} shall be no

11984 less than 255.

11985 If **wchar\_t** (see the <stddef.h> header) is defined as a signed integer type, the value of  
11986 {WCHAR\_MIN} shall be no greater than -127 and the value of {WCHAR\_MAX} shall be no less  
11987 than 127; otherwise, **wchar\_t** shall be defined as an unsigned integer type, and the value of  
11988 {WCHAR\_MIN} shall be 0 and the value of {WCHAR\_MAX} shall be no less than 255.

11989 If **wint\_t** (see the <wchar.h> header) is defined as a signed integer type, the value of  
11990 {WINT\_MIN} shall be no greater than -32767 and the value of {WINT\_MAX} shall be no less  
11991 than 32767; otherwise, **wint\_t** shall be defined as an unsigned integer type, and the value of  
11992 {WINT\_MIN} shall be 0 and the value of {WINT\_MAX} shall be no less than 65535.

### 11993 **Macros for Integer Constant Expressions**

11994 The following macros expand to integer constant expressions suitable for initializing objects that  
11995 have integer types corresponding to types defined in the <stdint.h> header. Each macro name  
11996 corresponds to a similar type name listed under *Minimum-width integer types* and *Greatest-width*  
11997 *integer types*.

11998 Each invocation of one of these macros shall expand to an integer constant expression suitable  
11999 for use in #if preprocessing directives. The type of the expression shall have the same type as  
12000 would an expression that is an object of the corresponding type converted according to the  
12001 integer promotions. The value of the expression shall be that of the argument.

12002 The argument in any instance of these macros shall be an unsuffixed integer constant with a  
12003 value that does not exceed the limits for the corresponding type.

- 12004 • Macros for minimum-width integer constant expressions

12005 The macro **INTN\_C(value)** shall expand to an integer constant expression corresponding to  
12006 the type **int\_leastN\_t**. The macro **UINTN\_C(value)** shall expand to an integer constant  
12007 expression corresponding to the type **uint\_leastN\_t**. For example, if **uint\_least64\_t** is a  
12008 name for the type **unsigned long long**, then **UINT64\_C(0x123)** might expand to the integer  
12009 constant 0x123ULL.

- 12010 • Macros for greatest-width integer constant expressions

12011 The following macro expands to an integer constant expression having the value specified  
12012 by its argument and the type **intmax\_t**:

12013 **INTMAX\_C(value)**

12014 The following macro expands to an integer constant expression having the value specified  
12015 by its argument and the type **uintmax\_t**:

12016 **UINTMAX\_C(value)**

### 12017 **APPLICATION USAGE**

12018 None.

### 12019 **RATIONALE**

12020 The <stdint.h> header is a subset of the <inttypes.h> header more suitable for use in  
12021 freestanding environments, which might not support the formatted I/O functions. In some  
12022 environments, if the formatted conversion support is not wanted, using this header instead of  
12023 the <inttypes.h> header avoids defining such a large number of macros.

12024 As a consequence of adding **int8\_t**, the following are true:

12025     • A byte is exactly 8 bits.

12026     • {CHAR\_BIT} has the value 8, {SCHAR\_MAX} has the value 127, {SCHAR\_MIN} has the

12027       value -128, and {UCHAR\_MAX} has the value 255.

12028     (The POSIX standard explicitly requires 8-bit char and two's-complement arithmetic.)

12029 **FUTURE DIRECTIONS**

12030     **typedef** names beginning with **int** or **uint** and ending with **\_t** may be added to the types defined

12031     in the **<stdint.h>** header. Macro names beginning with **INT** or **UINT** and ending with **\_MAX**,

12032     **\_MIN**, or **\_C** may be added to the macros defined in the **<stdint.h>** header.

12033 **SEE ALSO**

12034     [\*\*<inttypes.h>\*\*](#), [\*\*<signal.h>\*\*](#), [\*\*<stddef.h>\*\*](#), [\*\*<wchar.h>\*\*](#)

12035     XSH [Section 2.2](#) (on page 472)

12036 **CHANGE HISTORY**

12037     First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

12038     ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is applied.

12039 **Issue 7**

12040     ISO/IEC 9899:1999 standard, Technical Corrigendum 3 #40 is applied.

12041     SD5-XBD-ERN-67 is applied.

12042 **NAME**

12043       stdio.h — standard buffered input/output

12044 **SYNOPSIS**

12045       #include &lt;stdio.h&gt;

12046 **DESCRIPTION**

12047 CX       Some of the functionality described on this reference page extends the ISO C standard.  
 12048       Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to  
 12049       enable the visibility of these symbols in this header.

12050       The <stdio.h> header shall define the following data types through **typedef**:12051       **FILE**               A structure containing information about a file.

12052       **fpos\_t**            A non-array type containing all information needed to specify uniquely  
 12053       every position within a file.

12054       **off\_t**             As described in <sys/types.h>.12055       **size\_t**            As described in <stddef.h>.12056 CX       **ssize\_t**       As described in <sys/types.h>.12057 CX       **va\_list**       As described in <stdarg.h>.

12058       The <stdio.h> header shall define the following macros which shall expand to integer constant  
 12059       expressions:

12060 CX       **BUFSIZ**       Size of <stdio.h> buffers.   This shall expand to a positive value.12061 CX       **L\_ctermid**     Maximum size of character array to hold *ctermid()* output.12062 OB       **L\_tmpnam**     Maximum size of character array to hold *tmpnam()* output.

12063       The <stdio.h> header shall define the following macros which shall expand to integer constant  
 12064       expressions with distinct values:

12065       **\_IOFBF**           Input/output fully buffered.12066       **\_IOLBF**           Input/output line buffered.12067       **\_IONBF**           Input/output unbuffered.

12068       The <stdio.h> header shall define the following macros which shall expand to integer constant  
 12069       expressions with distinct values:

12070       **SEEK\_CUR**       Seek relative to current position.12071       **SEEK\_END**       Seek relative to end-of-file.12072       **SEEK\_SET**       Seek relative to start-of-file.

12073       The <stdio.h> header shall define the following macros which shall expand to integer constant  
 12074       expressions denoting implementation limits:

12075       {FILENAME\_MAX}   Maximum size in bytes of the longest pathname that the implementation  
 12076       guarantees can be opened.

12077       {FOPEN\_MAX}     Number of streams which the implementation guarantees can be open  
 12078       simultaneously. The value is at least eight.

12079 OB       {TMP\_MAX}     Minimum number of unique filenames generated by *tmpnam()*.  
 12080       Maximum number of times an application can call *tmpnam()* reliably. The  
 12081       value of {TMP\_MAX} is at least 25.

12082	OB XSI	On XSI-conformant systems, the value of {TMP_MAX} is at least 10 000.
12083		The <stdio.h> header shall define the following macro which shall expand to an integer constant
12084		expression with type <b>int</b> and a negative value:
12085		EOF                      End-of-file return value.
12086		The <stdio.h> header shall define NULL as described in <stddef.h>.
12087		The <stdio.h> header shall define the following macro which shall expand to a string constant:
12088	OB XSI	P_tmpdir                Default directory prefix for <i>tempnam()</i> .
12089		The <stdio.h> header shall define the following macros which shall expand to expressions of
12090		type “pointer to <b>FILE</b> ” that point to the <b>FILE</b> objects associated, respectively, with the standard
12091		error, input, and output streams:
12092		<i>stderr</i> Standard error output stream.
12093		<i>stdin</i> Standard input stream.
12094		<i>stdout</i> Standard output stream.
12095		The following shall be declared as functions and may also be defined as macros. Function
12096		prototypes shall be provided.
12097		void            clearerr(FILE *);
12098	CX	char           *ctermid(char *);
12099		int            dprintf(int, const char *restrict, ...)
12100		int            fclose(FILE *);
12101	CX	FILE          *fdopen(int, const char *);
12102		int            feof(FILE *);
12103		int            ferror(FILE *);
12104		int            fflush(FILE *);
12105		int            fgetc(FILE *);
12106		int            fgetpos(FILE *restrict, fpos_t *restrict);
12107		char          *fgets(char *restrict, int, FILE *restrict);
12108	CX	int            fileno(FILE *);
12109		void          flockfile(FILE *);
12110		FILE          *fmemopen(void *restrict, size_t, const char *restrict);
12111		FILE          *fopen(const char *restrict, const char *restrict);
12112		int            fprintf(FILE *restrict, const char *restrict, ...);
12113		int            fputc(int, FILE *);
12114		int            fputs(const char *restrict, FILE *restrict);
12115		size_t        fread(void *restrict, size_t, size_t, FILE *restrict);
12116		FILE          *freopen(const char *restrict, const char *restrict,
12117		FILE *restrict);
12118		int            fscanf(FILE *restrict, const char *restrict, ...);
12119		int            fseek(FILE *, long, int);
12120	CX	int            fseeko(FILE *, off_t, int);
12121		int            fsetpos(FILE *, const fpos_t *);
12122		long          ftell(FILE *);
12123	CX	off_t         ftello(FILE *);
12124		int            ftrylockfile(FILE *);
12125		void          funlockfile(FILE *);
12126		size_t        fwrite(const void *restrict, size_t, size_t, FILE *restrict);
12127		int            getc(FILE *);

```

12128      int      getchar(void);
12129 CX      int     getc_unlocked(FILE *);
12130      int      getchar_unlocked(void);
12131      ssize_t   getdelim(char **restrict, size_t *restrict, int,
12132                        FILE *restrict);
12133      ssize_t   getline(char **restrict, size_t *restrict, FILE *restrict);
12134 OB      char     *gets(char *);
12135 CX      FILE     *open_memstream(char **, size_t *);
12136      int      pclose(FILE *);
12137      void      perror(const char *);
12138 CX      FILE     *popen(const char *, const char *);
12139      int      printf(const char *restrict, ...);
12140      int      putc(int, FILE *);
12141      int      putchar(int);
12142 CX      int      putc_unlocked(int, FILE *);
12143      int      putchar_unlocked(int);
12144      int      puts(const char *);
12145      int      remove(const char *);
12146      int      rename(const char *, const char *);
12147 CX      int      renameat(int, const char *, int, const char *);
12148      void      rewind(FILE *);
12149      int      scanf(const char *restrict, ...);
12150      void      setbuf(FILE *restrict, char *restrict);
12151      int      setvbuf(FILE *restrict, char *restrict, int, size_t);
12152      int      snprintf(char *restrict, size_t, const char *restrict, ...);
12153      int      sprintf(char *restrict, const char *restrict, ...);
12154      int      sscanf(const char *restrict, const char *restrict, ...);
12155 OB XSI      char  *tempnam(const char *, const char *);
12156      FILE     *tmpfile(void);
12157 OB      char  *tmpnam(char *);
12158      int      ungetc(int, FILE *);
12159 CX      int      vdprintf(int, const char *restrict, va_list);
12160      int      vfprintf(FILE *restrict, const char *restrict, va_list);
12161      int      vfscanf(FILE *restrict, const char *restrict, va_list);
12162      int      vprintf(const char *restrict, va_list);
12163      int      vscanf(const char *restrict, va_list);
12164      int      vsnprintf(char *restrict, size_t, const char *restrict,
12165                        va_list);
12166      int      vsprintf(char *restrict, const char *restrict, va_list);
12167      int      vsscanf(const char *restrict, const char *restrict, va_list);
12168 CX      Inclusion of the <stdio.h> header may also make visible all symbols from <stddef.h>.

```

## 12169 APPLICATION USAGE

12170 Since standard I/O streams may use an underlying file descriptor to access the file associated  
12171 with a stream, application developers need to be aware that {FOPEN\_MAX} streams may not be  
12172 available if file descriptors are being used to access files that are not associated with streams.

## 12173 RATIONALE

12174 There is a conflict between the ISO C standard and the POSIX definition of the {TMP\_MAX}  
12175 macro that is addressed by ISO/IEC 9899:1999 standard, Defect Report 336. The POSIX standard  
12176 is in alignment with the public record of the response to the Defect Report. This change has not  
12177 yet been published as part of the ISO C standard.



12178 **FUTURE DIRECTIONS**

12179 None.

12180 **SEE ALSO**12181 [<stdarg.h>](#), [<stddef.h>](#), [<sys/types.h>](#)

12182 XSH Section 2.2 (on page 472), [clearerr\(\)](#), [ctermid\(\)](#), [fclose\(\)](#), [fdopen\(\)](#), [feof\(\)](#), [ferror\(\)](#), [fflush\(\)](#),  
 12183 [fgetc\(\)](#), [fgetpos\(\)](#), [fgets\(\)](#), [fileno\(\)](#), [flockfile\(\)](#), [fmemopen\(\)](#), [fopen\(\)](#), [fprintf\(\)](#), [fputc\(\)](#), [fputs\(\)](#), [fread\(\)](#),  
 12184 [freopen\(\)](#), [fscanf\(\)](#), [fseek\(\)](#), [fsetpos\(\)](#), [ftell\(\)](#), [fwrite\(\)](#), [getc\(\)](#), [getchar\(\)](#), [getc\\_unlocked\(\)](#), [getdelim\(\)](#),  
 12185 [getopt\(\)](#), [gets\(\)](#), [open\\_memstream\(\)](#), [pclose\(\)](#), [perror\(\)](#), [popen\(\)](#), [putc\(\)](#), [putchar\(\)](#), [puts\(\)](#), [remove\(\)](#),  
 12186 [rename\(\)](#), [rewind\(\)](#), [setbuf\(\)](#), [setvbuf\(\)](#), [stdin](#), [system\(\)](#), [tempnam\(\)](#), [tmpfile\(\)](#), [tmpnam\(\)](#), [ungetc\(\)](#),  
 12187 [vfprintf\(\)](#), [vfscanf\(\)](#)

12188 **CHANGE HISTORY**

12189 First released in Issue 1. Derived from Issue 1 of the SVID.

12190 **Issue 5**

12191 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

12192 Large File System extensions are added.

12193 The constant `L_cuserid` and the external variables `optarg`, `opterr`, `optind`, and `optopt` are marked as  
 12194 extensions and LEGACY.

12195 The `cuserid()` and `getopt()` functions are marked LEGACY.12196 **Issue 6**

12197 The constant `L_cuserid` and the external variables `optarg`, `opterr`, `optind`, and `optopt` are removed  
 12198 as they were previously marked LEGACY.

12199 The `cuserid()`, `getopt()`, and `getw()` functions are removed as they were previously marked  
 12200 LEGACY.

12201 Several functions are marked as part of the Thread-Safe Functions option.

12202 This reference page is updated to align with the ISO/IEC 9899:1999 standard. Note that the  
 12203 description of the `fpos_t` type is now explicitly updated to exclude array types.

12204 Extensions beyond the ISO C standard are marked.

12205 **Issue 7**

12206 Austin Group Interpretation 1003.1-2001 #172 is applied, adding rationale about a conflict for the  
 12207 definition of `{TMP_MAX}` with the ISO C standard.

12208 SD5-XBD-ERN-99 is applied, adding APPLICATION USAGE.

12209 The `dprintf()`, `fmemopen()`, `getdelim()`, `getline()`, `open_memstream()`, and `vdprintf()` functions are  
 12210 added from The Open Group Technical Standard, 2006, Extended API Set Part 1.

12211 The `renameat()` function is added from The Open Group Technical Standard, 2006, Extended API  
 12212 Set Part 2.

12213 The `gets()`, `tmpnam()`, and `tempnam()` functions and the `L_tmpnam` macro are marked  
 12214 obsolescent.

12215 This reference page is clarified with respect to macros and symbolic constants, and a declaration  
 12216 for the `off_t` type is added.

12217 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0065 [291,427] is applied.

12218 **NAME**

12219        stdlib.h — standard library definitions

12220 **SYNOPSIS**

12221        #include &lt;stdlib.h&gt;

12222 **DESCRIPTION**

12223 CX       Some of the functionality described on this reference page extends the ISO C standard.  
 12224       Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to  
 12225       enable the visibility of these symbols in this header.

12226       The <stdlib.h> header shall define the following macros which shall expand to integer constant  
 12227       expressions:

12228       EXIT\_FAILURE   Unsuccessful termination for *exit()*; evaluates to a non-zero value.

12229       EXIT\_SUCCESS   Successful termination for *exit()*; evaluates to 0.

12230       {RAND\_MAX}     Maximum value returned by *rand()*; at least 32 767.

12231       The <stdlib.h> header shall define the following macro which shall expand to a positive integer  
 12232       expression with type **size\_t**:

12233       {MB\_CUR\_MAX}   Maximum number of bytes in a character specified by the current locale  
 12234       (category *LC\_CTYPE*).

12235 CX       In the POSIX locale the value of {MB\_CUR\_MAX} shall be 1.       +

12236       The <stdlib.h> header shall define NULL as described in <stddef.h>.

12237       The <stdlib.h> header shall define the following data types through **typedef**:

12238       **div\_t**           Structure type returned by the *div()* function.

12239       **ldiv\_t**          Structure type returned by the *ldiv()* function.

12240       **lldiv\_t**         Structure type returned by the *lldiv()* function.

12241       **size\_t**          As described in <stddef.h>.

12242       **wchar\_t**         As described in <stddef.h>.

12243 CX       In addition, the <stdlib.h> header shall define the following symbolic constants and macros as  
 12244       described in <sys/wait.h>:

12245       WEXITSTATUS

12246       WIFEXITED

12247       WIFSIGNALED

12248       WIFSTOPPED

12249       WNOHANG

12250       WSTOPSIG

12251       WTERMSIG

12252       WUNTRACED

12253       The following shall be declared as functions and may also be defined as macros. Function  
 12254       prototypes shall be provided.

12255       void            \_Exit(int);

12256 XSI       long       a64l(const char \*);

12257       void            abort(void);

12258       int             abs(int);

```

12259     int          atexit(void (*)(void));
12260     double        atof(const char *);
12261     int           atoi(const char *);
12262     long          atol(const char *);
12263     long long     atoll(const char *);
12264     void          *bsearch(const void *, const void *, size_t, size_t,
12265                             int (*)(const void *, const void *));
12266     void          *calloc(size_t, size_t);
12267     div_t         div(int, int);
12268 XSI     double        drand48(void);
12269     double        erand48(unsigned short [3]);
12270     void          exit(int);
12271     void          free(void *);
12272     char          *getenv(const char *);
12273 CX     int           getsubopt(char **, char *const *, char **);
12274 XSI     int           grantpt(int);
12275     char          *initstate(unsigned, char *, size_t);
12276     long          jrand48(unsigned short [3]);
12277     char          *l64a(long);
12278     long          labs(long);
12279 XSI     void          lcong48(unsigned short [7]);
12280     ldiv_t        ldiv(long, long);
12281     long long     llabs(long long);
12282     lldiv_t       lldiv(long long, long long);
12283 XSI     long          lrand48(void);
12284     void          *malloc(size_t);
12285     int           mblen(const char *, size_t);
12286     size_t        mbstowcs(wchar_t *restrict, const char *restrict, size_t);
12287     int           mbtowc(wchar_t *restrict, const char *restrict, size_t);
12288 CX     char          *mkdtemp(char *);
12289     int           mkstemp(char *);
12290 XSI     long          mrand48(void);
12291     long          nrand48(unsigned short [3]);
12292 ADV     int           posix_memalign(void **, size_t, size_t);
12293 XSI     int           posix_openpt(int);
12294     char          *ptsname(int);
12295     int           putenv(char *);
12296     void          qsort(void *, size_t, size_t, int (*)(const void *,
12297                                                         const void *));
12298     int           rand(void);
12299 OB CX     int          rand_r(unsigned *);
12300 XSI     long          random(void);
12301     void          *realloc(void *, size_t);
12302 XSI     char          *realpath(const char *restrict, char *restrict);
12303     unsigned short *seed48(unsigned short [3]);
12304 CX     int           setenv(const char *, const char *, int);
12305 XSI     void          setkey(const char *);
12306     char          *setstate(char *);
12307     void          srand(unsigned);
12308 XSI     void          srand48(long);
12309     void          srandom(unsigned);
12310     double        strtod(const char *restrict, char **restrict);

```

```

12311 float      strtouf(const char *restrict, char **restrict);
12312 long       strtoul(const char *restrict, char **restrict, int);
12313 long double strtold(const char *restrict, char **restrict);
12314 long long  strtoll(const char *restrict, char **restrict, int);
12315 unsigned long strtoul(const char *restrict, char **restrict, int);
12316 unsigned long long
12317 strtoull(const char *restrict, char **restrict, int);
12318 int        system(const char *);
12319 XSI int     unlockpt(int);
12320 CX int     unsetenv(const char *);
12321 size_t     wcstombs(char *restrict, const wchar_t *restrict, size_t);
12322 int        wctomb(char *, wchar_t);

12323 CX Inclusion of the <stdlib.h> header may also make visible all symbols from <stddef.h>,
12324 <limits.h>, <math.h>, and <sys/wait.h>.

```

## 12325 APPLICATION USAGE

12326 None.

## 12327 RATIONALE

12328 None.

## 12329 FUTURE DIRECTIONS

12330 None.

## 12331 SEE ALSO

12332 [<limits.h>](#), [<math.h>](#), [<stddef.h>](#), [<sys/types.h>](#), [<sys/wait.h>](#)

12333 XSH Section 2.2 (on page 472), [\\_Exit\(\)](#), [a64l\(\)](#), [abort\(\)](#), [abs\(\)](#), [atexit\(\)](#), [atof\(\)](#), [atoi\(\)](#), [atol\(\)](#),  
12334 [bsearch\(\)](#), [calloc\(\)](#), [div\(\)](#), [drand48\(\)](#), [exit\(\)](#), [free\(\)](#), [getenv\(\)](#), [getsubopt\(\)](#), [grantpt\(\)](#), [initstate\(\)](#), [labs\(\)](#),  
12335 [ldiv\(\)](#), [malloc\(\)](#), [mblen\(\)](#), [mbstowcs\(\)](#), [mbtowc\(\)](#), [mkdtemp\(\)](#), [posix\\_memalign\(\)](#), [posix\\_openpt\(\)](#),  
12336 [ptsname\(\)](#), [putenv\(\)](#), [qsort\(\)](#), [rand\(\)](#), [realloc\(\)](#), [realpath\(\)](#), [setenv\(\)](#), [setkey\(\)](#), [strtod\(\)](#), [strtol\(\)](#),  
12337 [strtoul\(\)](#), [system\(\)](#), [unlockpt\(\)](#), [unsetenv\(\)](#), [wcstombs\(\)](#), [wctomb\(\)](#)

## 12338 CHANGE HISTORY

12339 First released in Issue 3.

### 12340 Issue 5

12341 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

12342 The [ttypslot\(\)](#) and [valloc\(\)](#) functions are marked LEGACY.

12343 The type of the third argument to [initstate\(\)](#) is changed from **int** to **size\_t**. The type of the return  
12344 value from [setstate\(\)](#) is changed from **char \*** to **char \***, and the type of the first argument is  
12345 changed from **char \*** to **const char \***.

### 12346 Issue 6

12347 The Open Group Corrigendum U021/1 is applied, correcting the prototype for [realpath\(\)](#) to be  
12348 consistent with the reference page.

12349 The Open Group Corrigendum U028/13 is applied, correcting the prototype for [putenv\(\)](#) to be  
12350 consistent with the reference page.

12351 The [rand\\_r\(\)](#) function is marked as part of the Thread-Safe Functions option.

12352 Function prototypes for [setenv\(\)](#) and [unsetenv\(\)](#) are added.

12353 The [posix\\_memalign\(\)](#) function is added for alignment with IEEE Std 1003.1d-1999.

12354 This reference page is updated to align with the ISO/IEC 9899:1999 standard.

- 12355 The *ecvt()*, *fcvt()*, *gcvt()*, and *mktemp()* functions are marked LEGACY.
- 12356 The *ttyslot()* and *valloc()* functions are removed as they were previously marked LEGACY.
- 12357 Extensions beyond the ISO C standard are marked.
- 12358 **Issue 7**
- 12359 SD5-XBD-ERN-79 and SD5-XBD-ERN-105 are applied.
- 12360 The LEGACY functions are removed.
- 12361 The *mkdtemp()* function is added from The Open Group Technical Standard, 2006, Extended API
- 12362 Set Part 1.
- 12363 The *rand\_r()* function is marked obsolescent.
- 12364 This reference page is clarified with respect to macros and symbolic constants.
- 12365 The type of the first argument to *setstate()* is changed from **const char \*** to **char \***.
- 12366 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0066 [197] is applied.

## 12367 NAME

12368 string.h — string operations

## 12369 SYNOPSIS

12370 #include &lt;string.h&gt;

## 12371 DESCRIPTION

12372 CX Some of the functionality described on this reference page extends the ISO C standard.  
 12373 Applications shall define the appropriate feature test macro (see XSH Section 2.2, on page 472) to  
 12374 enable the visibility of these symbols in this header.

12375 The <string.h> header shall define NULL and size\_t as described in <stddef.h>.

12376 CX The <string.h> header shall define the locale\_t type as described in <locale.h>.

12377 The following shall be declared as functions and may also be defined as macros. Function  
 12378 prototypes shall be provided for use with ISO C standard compilers.

```

12379 XSI void *memcpy(void *restrict, const void *restrict, int, size_t);
12380 void *memchr(const void *, int, size_t);
12381 int memcmp(const void *, const void *, size_t);
12382 void *memcpy(void *restrict, const void *restrict, size_t);
12383 void *memmove(void *, const void *, size_t);
12384 void *memset(void *, int, size_t);
12385 CX char *strcpy(char *restrict, const char *restrict);
12386 char *strncpy(char *restrict, const char *restrict, size_t);
12387 char *strcat(char *restrict, const char *restrict);
12388 char *strchr(const char *, int);
12389 int strcmp(const char *, const char *);
12390 int strcoll(const char *, const char *);
12391 CX int strcoll_l(const char *, const char *, locale_t);
12392 char *strcpy(char *restrict, const char *restrict);
12393 size_t strcspn(const char *, const char *);
12394 CX char *strdup(const char *);
12395 char *strerror(int);
12396 CX char *strerror_l(int, locale_t);
12397 int strerror_r(int, char *, size_t);
12398 size_t strlen(const char *);
12399 char *strncat(char *restrict, const char *restrict, size_t);
12400 int strncmp(const char *, const char *, size_t);
12401 char *strncpy(char *restrict, const char *restrict, size_t);
12402 CX char *strndup(const char *, size_t);
12403 size_t strnlen(const char *, size_t);
12404 char *strpbrk(const char *, const char *);
12405 char *strrchr(const char *, int);
12406 CX char *strsignal(int);
12407 size_t strspn(const char *, const char *);
12408 char *strstr(const char *, const char *);
12409 char *strtok(char *restrict, const char *restrict);
12410 CX char *strtok_r(char *restrict, const char *restrict, char **restrict);
12411 size_t strxfrm(char *restrict, const char *restrict, size_t);
12412 CX size_t strxfrm_l(char *restrict, const char *restrict,
12413 size_t, locale_t);

```

12414	CX	Inclusion of the <string.h> header may also make visible all symbols from <stddef.h>.
12415		<b>APPLICATION USAGE</b>
12416		None.
12417		<b>RATIONALE</b>
12418		None.
12419		<b>FUTURE DIRECTIONS</b>
12420		None.
12421		<b>SEE ALSO</b>
12422		<locale.h>, <stddef.h>, <sys/types.h>
12423		XSH Section 2.2 (on page 472), <i>memcpy()</i> , <i>memchr()</i> , <i>memcmp()</i> , <i>memcpy()</i> , <i>memmove()</i> ,
12424		<i>memset()</i> , <i>strcat()</i> , <i>strchr()</i> , <i>strcmp()</i> , <i>strcoll()</i> , <i>strcpy()</i> , <i>strcspn()</i> , <i>strdup()</i> , <i>strerror()</i> , <i>strlen()</i> ,
12425		<i>strncat()</i> , <i>strncmp()</i> , <i>strncpy()</i> , <i>strpbrk()</i> , <i>strrchr()</i> , <i>strsignal()</i> , <i>strspn()</i> , <i>strstr()</i> , <i>strtok()</i> , <i>strxfrm()</i>
12426		<b>CHANGE HISTORY</b>
12427		First released in Issue 1. Derived from Issue 1 of the SVID.
12428	<b>Issue 5</b>	
12429		The DESCRIPTION is updated for alignment with the POSIX Threads Extension.
12430	<b>Issue 6</b>	
12431		The <i>strtok_r()</i> function is marked as part of the Thread-Safe Functions option.
12432		This reference page is updated to align with the ISO/IEC 9899:1999 standard.
12433		The <i>strerror_r()</i> function is added in response to IEEE PASC Interpretation 1003.1c #39.
12434	<b>Issue 7</b>	
12435		SD5-XBD-ERN-15 is applied, correcting the prototype for the <i>strerror_r()</i> function.
12436		The <i>stpncpy()</i> , <i>stpncpy()</i> , <i>strndup()</i> , <i>strnlen()</i> , and <i>strsignal()</i> functions are added from The Open
12437		Group Technical Standard, 2006, Extended API Set Part 1.
12438		The <i>strcoll_l()</i> , <i>strerror_l()</i> , and <i>strxfrm_l()</i> functions are added from The Open Group Technical
12439		Standard, 2006, Extended API Set Part 4.
12440		This reference page is clarified with respect to macros and symbolic constants, and a declaration
12441		for the <i>locale_t</i> type is added.

12442 **NAME**

12443 strings.h — string operations

12444 **SYNOPSIS**

12445 #include &lt;strings.h&gt;

12446 **DESCRIPTION**

12447 The following shall be declared as functions and may also be defined as macros. Function  
 12448 prototypes shall be provided for use with ISO C standard compilers.

```
12449 XSI      int      ffs(int);
12450          int      strcasecmp(const char *, const char *);
12451          int      strcasecmp_l(const char *, const char *, locale_t);
12452          int      strncasecmp(const char *, const char *, size_t);
12453          int      strncasecmp_l(const char *, const char *, size_t, locale_t);
```

12454 The <strings.h> header shall define the **locale\_t** type as described in <locale.h>.12455 The <strings.h> header shall define the **size\_t** type as described in <sys/types.h>.12456 **APPLICATION USAGE**

12457 None.

12458 **RATIONALE**

12459 None.

12460 **FUTURE DIRECTIONS**

12461 None.

12462 **SEE ALSO**

12463 &lt;locale.h&gt;, &lt;sys/types.h&gt;

12464 XSH *ffs()*, *strcasecmp()*12465 **CHANGE HISTORY**

12466 First released in Issue 4, Version 2.

12467 **Issue 6**

12468 The Open Group Corrigendum U021/2 is applied, correcting the prototype for *index()* to be  
 12469 consistent with the reference page.

12470 The *bcmp()*, *bcopy()*, *bzero()*, *index()*, and *rindex()* functions are marked LEGACY.12471 **Issue 7**12472 SD5-XBD-ERN-56 is applied, adding a reference to <sys/types.h> for the **size\_t** type.

12473 The LEGACY functions are removed.

12474 The &lt;strings.h&gt; header is moved from the XSI option to the Base.

12475 The *strcasecmp\_l()* and *strncasecmp\_l()* functions are added from The Open Group Technical  
 12476 Standard, 2006, Extended API Set Part 4.

12477 A declaration for the **locale\_t** type is added.



12478 **NAME**

12479       stropts.h — STREAMS interface (**STREAMS**)

12480 **SYNOPSIS**

12481 OB XSR   #include <stropts.h>

12482 **DESCRIPTION**

12483       The <stropts.h> header shall define the **bandinfo** structure, which shall include at least the  
12484       following members:

12485       int                bi\_flag    Flushing type.  
12486       unsigned char    bi\_pri    Priority band.

12487       The <stropts.h> header shall define the **strpeek** structure, which shall include at least the  
12488       following members:

12489       struct strbuf    ctlbuf    The control portion of the message.  
12490       struct strbuf    databuf   The data portion of the message.  
12491       t\_uscalar\_t     flags     RS\_HIPRI or 0.

12492       The <stropts.h> header shall define the **strbuf** structure, which shall include at least the  
12493       following members:

12494       char   \*buf       Pointer to buffer.  
12495       int    len        Length of data.  
12496       int    maxlen    Maximum buffer length.

12497       The <stropts.h> header shall define the **strfdinsert** structure, which shall include at least the  
12498       following members:

12499       struct strbuf    ctlbuf    The control portion of the message.  
12500       struct strbuf    databuf   The data portion of the message.  
12501       int             fildes     File descriptor of the other STREAM.  
12502       t\_uscalar\_t     flags     RS\_HIPRI or 0.  
12503       int             offset     Relative location of the stored value.

12504       The <stropts.h> header shall define the **striocctl** structure, which shall include at least the  
12505       following members:

12506       int    ic\_cmd       *ioctl()* command.  
12507       char   \*ic\_dp       Pointer to buffer.  
12508       int    ic\_len       Length of data.  
12509       int    ic\_timeout   Timeout for response.

12510       The <stropts.h> header shall define the **strrecvfd** structure, which shall include at least the  
12511       following members:

12512       int    fd        Received file descriptor.  
12513       gid\_t   gid       GID of sender.  
12514       uid\_t   uid       UID of sender.

12515       The <stropts.h> header shall define the **uid\_t** and **gid\_t** types through **typedef**, as described in  
12516       <sys/types.h>.

12517       The <stropts.h> header shall define the **t\_scalar\_t** and **t\_uscalar\_t** types, respectively, as signed  
12518       and unsigned opaque types of equal length of at least 32 bits.

12519       The <stropts.h> header shall define the **str\_list** structure, which shall include at least the  
12520       following members:

```

12521 struct str_mlist *sl_modlist STREAMS module names.
12522 int sl_nmods Number of STREAMS module names.
12523 The <stropts.h> header shall define the str_mlist structure, which shall include at least the
12524 following member:
12525 char l_name[FMNAMESZ+1] A STREAMS module name.
12526 The <stropts.h> header shall define at least the following symbolic constants for use as the
12527 request argument to ioctl():
12528 I_ATMARK Is the top message “marked”?
12529 I_CANPUT Is a band writable?
12530 I_CKBAND See if any messages exist in a band.
12531 I_FDINSERT Send implementation-defined information about another STREAM.
12532 I_FIND Look for a STREAMS module.
12533 I_FLUSH Flush a STREAM.
12534 I_FLUSHBAND Flush one band of a STREAM.
12535 I_GETBAND Get the band of the top message on a STREAM.
12536 I_GETCLTIME Get close time delay.
12537 I_GETSIG Retrieve current notification signals.
12538 I_GRDOPT Get the read mode.
12539 I_GWROPT Get the write mode.
12540 I_LINK Connect two STREAMS.
12541 I_LIST Get all the module names on a STREAM.
12542 I_LOOK Get the top module name.
12543 I_NREAD Size the top message.
12544 I_PEEK Peek at the top message on a STREAM.
12545 I_PLINK Persistently connect two STREAMS.
12546 I_POP Pop a STREAMS module.
12547 I_PUNLINK Dismantle a persistent STREAMS link.
12548 I_PUSH Push a STREAMS module.
12549 I_RECVFD Get a file descriptor sent via I_SENDFD.
12550 I_SENDFD Pass a file descriptor through a STREAMS pipe.
12551 I_SETCLTIME Set close time delay.
12552 I_SETSIG Ask for notification signals.
12553 I_SRDOPT Set the read mode.
12554 I_STR Send a STREAMS ioctl().

```

12555	I_SWROPT	Set the write mode.
12556	I_UNLINK	Disconnect two STREAMs.
12557	The <stropts.h> header shall define at least the following symbolic constant for use with	
12558	I_LOOK:	
12559	FMNAMESZ	The minimum size in bytes of the buffer referred to by the <i>arg</i> argument.
12560	The <stropts.h> header shall define at least the following symbolic constants for use with	
12561	I_FLUSH:	
12562	FLUSHR	Flush read queues.
12563	FLUSHRW	Flush read and write queues.
12564	FLUSHW	Flush write queues.
12565	The <stropts.h> header shall define at least the following symbolic constants for use with	
12566	I_SETSIG:	
12567	S_BANDURG	When used in conjunction with S_RDBAND, SIGURG is generated instead of SIGPOLL when a priority message reaches the front of the STREAM head read queue.
12568		
12569		
12570	S_ERROR	Notification of an error condition reaches the STREAM head.
12571	S_HANGUP	Notification of a hangup reaches the STREAM head.
12572	S_HIPRI	A high-priority message is present on a STREAM head read queue.
12573	S_INPUT	A message, other than a high-priority message, has arrived at the head of a STREAM head read queue.
12574		
12575	S_MSG	A STREAMS signal message that contains the SIGPOLL signal reaches the front of the STREAM head read queue.
12576		
12577	S_OUTPUT	The write queue for normal data (priority band 0) just below the STREAM head is no longer full. This notifies the process that there is room on the queue for sending (or writing) normal data downstream.
12578		
12579		
12580	S_RDBAND	A message with a non-zero priority band has arrived at the head of a STREAM head read queue.
12581		
12582	S_RDNORM	A normal (priority band set to 0) message has arrived at the head of a STREAM head read queue.
12583		
12584	S_WRBAND	The write queue for a non-zero priority band just below the STREAM head is no longer full.
12585		
12586	S_WRNORM	Equivalent to S_OUTPUT.
12587	The <stropts.h> header shall define at least the following symbolic constant for use with	
12588	I_PEEK:	
12589	RS_HIPRI	Only look for high-priority messages.
12590	The <stropts.h> header shall define at least the following symbolic constants for use with	
12591	I_SRDOPT:	
12592	RMSGD	Message-discard mode.

12593	RMSGN	Message-non-discard mode.
12594	RNORM	Byte-STREAM mode, the default.
12595	RPROTDAT	Deliver the control part of a message as data when a process issues a <i>read()</i> .
12596	RPROTDIS	Discard the control part of a message, delivering any data part, when a process issues a <i>read()</i> .
12597		
12598	RPROTNORM	Fail <i>read()</i> with [EBADMSG] if a message containing a control part is at the front of the STREAM head read queue.
12599		
12600	The <stropts.h> header shall define at least the following symbolic constant for use with	
12601	L_SWOPT:	
12602	SNDZERO	Send a zero-length message downstream when a <i>write()</i> of 0 bytes occurs.
12603	The <stropts.h> header shall define at least the following symbolic constants for use with	
12604	L_ATMARK:	
12605	ANYMARK	Check if the message is marked.
12606	LASTMARK	Check if the message is the last one marked on the queue.
12607	The <stropts.h> header shall define at least the following symbolic constant for use with	
12608	L_UNLINK:	
12609	MUXID_ALL	Unlink all STREAMs linked to the STREAM associated with <i>files</i> .
12610	The <stropts.h> header shall define the following symbolic constants for <i>getmsg()</i> , <i>getpmsg()</i> ,	
12611	<i>putmsg()</i> , and <i>putpmsg()</i> :	
12612	MORECTL	More control information is left in message.
12613	MOREDATA	More data is left in message.
12614	MSG_ANY	Receive any message.
12615	MSG_BAND	Receive message from specified band.
12616	MSG_HIPRI	Send/receive high-priority message.
12617	The <stropts.h> header may make visible all of the symbols from <unistd.h>.	
12618	The <stropts.h> header may also define macros for message types using names that start with	
12619	M_.	
12620	The following shall be declared as functions and may also be defined as macros. Function	
12621	prototypes shall be provided.	
12622	int	fattach(int, const char *);
12623	int	fdetach(const char *);
12624	int	getmsg(int, struct strbuf *restrict, struct strbuf *restrict,
12625		int *restrict);
12626	int	getpmsg(int, struct strbuf *restrict, struct strbuf *restrict,
12627		int *restrict, int *restrict);
12628	int	ioctl(int, int, ...);
12629	int	isastream(int);
12630	int	putmsg(int, const struct strbuf *, const struct strbuf *, int);
12631	int	putpmsg(int, const struct strbuf *, const struct strbuf *, int,
12632		int);

12633 **APPLICATION USAGE**

12634 None.

12635 **RATIONALE**

12636 None.

12637 **FUTURE DIRECTIONS**

12638 None.

12639 **SEE ALSO**

12640 [<sys/types.h>](#), [<unistd.h>](#)

12641 XSH [close\(\)](#), [fattach\(\)](#), [fcntl\(\)](#), [fdetach\(\)](#), [getmsg\(\)](#), [ioctl\(\)](#), [isastream\(\)](#), [open\(\)](#), [pipe\(\)](#), [read\(\)](#), [poll\(\)](#),

12642 [putmsg\(\)](#), [signal\(\)](#), [write\(\)](#)

12643 **CHANGE HISTORY**

12644 First released in Issue 4, Version 2.

12645 **Issue 5**

12646 The *flags* members of the **strpeek** and **strfdinsert** structures are changed from **type long** to  
12647 **t\_uscalar\_t**.

12648 **Issue 6**

12649 This header is marked as part of the XSI STREAMS Option Group.

12650 The **restrict** keyword is added to the prototypes for [getmsg\(\)](#) and [getpmsg\(\)](#).

12651 **Issue 7**

12652 SD5-XBD-ERN-87 is applied, correcting an error in the **strrecvfd** structure.

12653 The **<stropts.h>** header is marked obsolescent.

12654 This reference page is clarified with respect to macros and symbolic constants.

12655 **NAME**

12656 sys/ipc.h — XSI interprocess communication access structure

12657 **SYNOPSIS**12658 XSI `#include <sys/ipc.h>`12659 **DESCRIPTION**

12660 The **<sys/ipc.h>** header is used by three mechanisms for XSI interprocess communication (IPC):  
 12661 messages, semaphores, and shared memory. All use a common structure type, **ipc\_perm**, to pass  
 12662 information used in determining permission to perform an IPC operation.

12663 The **<sys/ipc.h>** header shall define the **ipc\_perm** structure, which shall include the following  
 12664 members:

12665	uid_t	uid	Owner's user ID.
12666	gid_t	gid	Owner's group ID.
12667	uid_t	cuid	Creator's user ID.
12668	gid_t	cgid	Creator's group ID.
12669	mode_t	mode	Read/write permission.

12670 The **<sys/ipc.h>** header shall define the **uid\_t**, **gid\_t**, **mode\_t**, and **key\_t** types as described in  
 12671 **<sys/types.h>**.

12672 The **<sys/ipc.h>** header shall define the following symbolic constants.

12673 Mode bits:

12674	IPC_CREAT	Create entry if key does not exist.
12675	IPC_EXCL	Fail if key exists.
12676	IPC_NOWAIT	Error if request must wait.

12677 Keys:

12678	IPC_PRIVATE	Private key.
-------	-------------	--------------

12679 Control commands:

12680	IPC_RMID	Remove identifier.
12681	IPC_SET	Set options.
12682	IPC_STAT	Get options.

12683 The following shall be declared as a function and may also be defined as a macro. A function  
 12684 prototype shall be provided.

12685 `key_t ftok(const char *, int);`

12686 **APPLICATION USAGE**

12687 None.

12688 **RATIONALE**

12689 None.

12690 **FUTURE DIRECTIONS**

12691 None.

12692 **SEE ALSO**

12693 [<sys/types.h>](#)

12694 XSH [frok\(\)](#)

12695 **CHANGE HISTORY**

12696 First released in Issue 2. Derived from System V Release 2.0.

12697 **Issue 7**

12698 This reference page is clarified with respect to macros and symbolic constants.

12699 **NAME**

12700 sys/mman.h — memory management declarations

12701 **SYNOPSIS**

12702 #include &lt;sys/mman.h&gt;

12703 **DESCRIPTION**12704 The <sys/mman.h> header shall define the following symbolic constants for use as protection  
12705 options:

12706 PROT\_EXEC Page can be executed.

12707 PROT\_NONE Page cannot be accessed.

12708 PROT\_READ Page can be read.

12709 PROT\_WRITE Page can be written.

12710 The &lt;sys/mman.h&gt; header shall define the following symbolic constants for use as flag options:

12711 MAP\_FIXED Interpret *addr* exactly.

12712 MAP\_PRIVATE Changes are private.

12713 MAP\_SHARED Share changes.

12714 XSI|SIO The <sys/mman.h> header shall define the following symbolic constants for the *msync()*  
12715 function:

12716 MS\_ASYNC Perform asynchronous writes.

12717 MS\_INVALIDATE Invalidate mappings.

12718 MS\_SYNC Perform synchronous writes.

12719 ML The <sys/mman.h> header shall define the following symbolic constants for the *mlockall()*  
12720 function:

12721 MCL\_CURRENT Lock currently mapped pages.

12722 MCL\_FUTURE Lock pages that become mapped.

12723 The <sys/mman.h> header shall define the symbolic constant MAP\_FAILED which shall have  
12724 type **void \*** and shall be used to indicate a failure from the *mmap()* function .12725 ADV If the Advisory Information option is supported, the <sys/mman.h> header shall define  
12726 symbolic constants for the *advice* argument to the *posix\_madvise()* function as follows:

12727 POSIX\_MADV\_DONTNEED

12728 The application expects that it will not access the specified range in the near future.

12729 POSIX\_MADV\_NORMAL

12730 The application has no advice to give on its behavior with respect to the specified range. It  
12731 is the default characteristic if no advice is given for a range of memory.

12732 POSIX\_MADV\_RANDOM

12733 The application expects to access the specified range in a random order.

12734 POSIX\_MADV\_SEQUENTIAL

12735 The application expects to access the specified range sequentially from lower addresses to  
12736 higher addresses.



12737		POSIX_MADV_WILLNEED
12738		The application expects to access the specified range in the near future.
12739	TYM	The <sys/mman.h> header shall define the following symbolic constants for use as flags for the <i>posix_typed_mem_open()</i> function:
12740		
12741		POSIX_TYPED_MEM_ALLOCATE
12742		Allocate on <i>mmap()</i> .
12743		POSIX_TYPED_MEM_ALLOCATE_CONTIG
12744		Allocate contiguously on <i>mmap()</i> .
12745		POSIX_TYPED_MEM_MAP_ALLOCATABLE
12746		Map on <i>mmap()</i> , without affecting allocatability.
12747		The <sys/mman.h> header shall define the <b>mode_t</b> , <b>off_t</b> , and <b>size_t</b> types as described in
12748		<sys/types.h>.
12749	TYM	The <sys/mman.h> header shall define the <b>posix_typed_mem_info</b> structure, which shall
12750		include at least the following member:
12751		<b>size_t</b> <i>posix_tmi_length</i> Maximum length which may be allocated
12752		from a typed memory object.
12753		The following shall be declared as functions and may also be defined as macros. Function
12754		prototypes shall be provided.
12755	MLR	<b>int</b> <i>mlock</i> (const void *, <b>size_t</b> );
12756	ML	<b>int</b> <i>mlockall</i> ( <b>int</b> );
12757		<b>void</b> * <i>mmap</i> (void *, <b>size_t</b> , <b>int</b> , <b>int</b> , <b>int</b> , <b>off_t</b> );
12758		<b>int</b> <i>mprotect</i> (void *, <b>size_t</b> , <b>int</b> );
12759	XSI SIO	<b>int</b> <i>msync</i> (void *, <b>size_t</b> , <b>int</b> );
12760	MLR	<b>int</b> <i>munlock</i> (const void *, <b>size_t</b> );
12761	ML	<b>int</b> <i>munlockall</i> (void);
12762		<b>int</b> <i>munmap</i> (void *, <b>size_t</b> );
12763	ADV	<b>int</b> <i>posix_madvise</i> (void *, <b>size_t</b> , <b>int</b> );
12764	TYM	<b>int</b> <i>posix_mem_offset</i> (const void *restrict, <b>size_t</b> , <b>off_t</b> *restrict,
12765		<b>size_t</b> *restrict, <b>int</b> *restrict);
12766		<b>int</b> <i>posix_typed_mem_get_info</i> ( <b>int</b> , struct <i>posix_typed_mem_info</i> *);
12767		<b>int</b> <i>posix_typed_mem_open</i> (const char *, <b>int</b> , <b>int</b> );
12768	SHM	<b>int</b> <i>shm_open</i> (const char *, <b>int</b> , <b>mode_t</b> );
12769		<b>int</b> <i>shm_unlink</i> (const char *);

12770 **APPLICATION USAGE**

12771 None.

12772 **RATIONALE**

12773 None.

12774 **FUTURE DIRECTIONS**

12775 None.

12776 **SEE ALSO**12777 [`<sys/types.h>`](#)

12778 XSH [`mlock\(\)`](#), [`mlockall\(\)`](#), [`mmap\(\)`](#), [`mprotect\(\)`](#), [`msync\(\)`](#), [`munmap\(\)`](#), [`posix\_madvise\(\)`](#),  
12779 [`posix\_mem\_offset\(\)`](#), [`posix\_typed\_mem\_get\_info\(\)`](#), [`posix\_typed\_mem\_open\(\)`](#), [`shm\_open\(\)`](#),  
12780 [`shm\_unlink\(\)`](#)

12781 **CHANGE HISTORY**

12782 First released in Issue 4, Version 2.

12783 **Issue 5**

12784 Updated for alignment with the POSIX Realtime Extension.

12785 **Issue 6**

12786 The [`<sys/mman.h>`](#) header is marked as dependent on support for either the Memory Mapped  
12787 Files, Process Memory Locking, or Shared Memory Objects options.

12788 The following changes are made for alignment with IEEE Std 1003.1j-2000:

- 12789 • The TYM margin code is added to the list of margin codes for the [`<sys/mman.h>`](#) header  
12790 line, as well as for other lines.
- 12791 • The POSIX\_TYPED\_MEM\_ALLOCATE, POSIX\_TYPED\_MEM\_ALLOCATE\_CONTIG,  
12792 and POSIX\_TYPED\_MEM\_MAP\_ALLOCATABLE flags are added.
- 12793 • The [`posix\_tmi\_length`](#) structure is added.
- 12794 • The [`posix\_mem\_offset\(\)`](#), [`posix\_typed\_mem\_get\_info\(\)`](#), and [`posix\_typed\_mem\_open\(\)`](#) functions  
12795 are added.

12796 The **restrict** keyword is added to the prototype for [`posix\_mem\_offset\(\)`](#).12797 IEEE PASC Interpretation 1003.1 #102 is applied, adding the prototype for [`posix\_madvise\(\)`](#).

12798 IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/16 is applied, correcting margin code and  
12799 shading errors for the [`mlock\(\)`](#) and [`munlock\(\)`](#) functions.

12800 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/34 is applied, changing the margin code  
12801 for the [`mmap\(\)`](#) function from MF|SHM to MC3 (notation for MF|SHM|TYM).

12802 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/36 is applied, changing the margin code  
12803 for the [`munmap\(\)`](#) function from MF|SHM to MC3 (notation for MF|SHM|TYM).

12804 **Issue 7**

12805 SD5-XBD-ERN-5 is applied, rewriting the DESCRIPTION.

12806 Functionality relating to the Memory Protection and Memory Mapped Files options is moved to  
12807 the Base.

12808 This reference page is clarified with respect to macros and symbolic constants.

12809 **NAME**

12810 sys/msg.h — XSI message queue structures

12811 **SYNOPSIS**12812 XSI `#include <sys/msg.h>`12813 **DESCRIPTION**12814 The <sys/msg.h> header shall define the following data types through **typedef**:12815 **msgqnum\_t** Used for the number of messages in the message queue.12816 **msglen\_t** Used for the number of bytes allowed in a message queue.12817 These types shall be unsigned integer types that are able to store values at least as large as a type  
12818 **unsigned short**.12819 The <sys/msg.h> header shall define the following symbolic constant as a message operation  
12820 flag:12821 **MSG\_NOERROR** No error if big message.12822 The <sys/msg.h> header shall define the **msqid\_ds** structure, which shall include the following  
12823 members:

12824	<code>struct ipc_perm</code>	<code>msg_perm</code>	Operation permission structure.
12825	<code>msgqnum_t</code>	<code>msg_qnum</code>	Number of messages currently on queue.
12826	<code>msglen_t</code>	<code>msg_qbytes</code>	Maximum number of bytes allowed on queue.
12827	<code>pid_t</code>	<code>msg_lspid</code>	Process ID of last <i>msgsnd()</i> .
12828	<code>pid_t</code>	<code>msg_lrpid</code>	Process ID of last <i>msgrcv()</i> .
12829	<code>time_t</code>	<code>msg_stime</code>	Time of last <i>msgsnd()</i> .
12830	<code>time_t</code>	<code>msg_rtime</code>	Time of last <i>msgrcv()</i> .
12831	<code>time_t</code>	<code>msg_ctime</code>	Time of last change.

12832 The <sys/msg.h> header shall define the **pid\_t**, **size\_t**, **ssize\_t**, and **time\_t** types as described in  
12833 <sys/types.h>.12834 The following shall be declared as functions and may also be defined as macros. Function  
12835 prototypes shall be provided.

```

12836 int      msgctl(int, int, struct msqid_ds *);
12837 int      msgget(key_t, int);
12838 ssize_t  msgrcv(int, void *, size_t, long, int);
12839 int      msgsnd(int, const void *, size_t, int);

```

12840 In addition, the &lt;sys/msg.h&gt; header shall include the &lt;sys/ipc.h&gt; header.

12841 **APPLICATION USAGE**

12842 None.

12843 **RATIONALE**

12844 None.

12845 **FUTURE DIRECTIONS**

12846 None.

12847 **SEE ALSO**12848 [<sys/ipc.h>](#), [<sys/types.h>](#)12849 XSH [msgctl\(\)](#), [msgget\(\)](#), [msgrcv\(\)](#), [msgsnd\(\)](#)

12850 **CHANGE HISTORY**

12851 First released in Issue 2. Derived from System V Release 2.0.

12852 **Issue 7**

12853 Austin Group Interpretation 1003.1-2001 #179 is applied.

12854 This reference page is clarified with respect to macros and symbolic constants.

## 12855 NAME

12856 sys/resource.h — definitions for XSI resource operations

## 12857 SYNOPSIS

12858 XSI `#include <sys/resource.h>`

## 12859 DESCRIPTION

12860 The <sys/resource.h> header shall define the following symbolic constants as possible values of  
12861 the *which* argument of *getpriority()* and *setpriority()*:

12862 PRIO\_PROCESS Identifies the *who* argument as a process ID.

12863 PRIO\_PGRP Identifies the *who* argument as a process group ID.

12864 PRIO\_USER Identifies the *who* argument as a user ID.

12865 The <sys/resource.h> header shall define the following type through **typedef**:

12866 **rlim\_t** Unsigned integer type used for limit values.

12867 The <sys/resource.h> header shall define the following symbolic constants, which shall have  
12868 values suitable for use in **#if** preprocessing directives:

12869 RLIM\_INFINITY A value of **rlim\_t** indicating no limit.

12870 RLIM\_SAVED\_MAX A value of type **rlim\_t** indicating an unrepresentable saved hard  
12871 limit.

12872 RLIM\_SAVED\_CUR A value of type **rlim\_t** indicating an unrepresentable saved soft limit.

12873 On implementations where all resource limits are representable in an object of type **rlim\_t**,  
12874 RLIM\_SAVED\_MAX and RLIM\_SAVED\_CUR need not be distinct from RLIM\_INFINITY.

12875 The <sys/resource.h> header shall define the following symbolic constants as possible values of  
12876 the *who* parameter of *getrusage()*:

12877 RUSAGE\_SELF Returns information about the current process.

12878 RUSAGE\_CHILDREN Returns information about children of the current process.

12879 The <sys/resource.h> header shall define the **rlimit** structure, which shall include at least the  
12880 following members:

12881 **rlim\_t** **rlim\_cur** The current (soft) limit.

12882 **rlim\_t** **rlim\_max** The hard limit.

12883 The <sys/resource.h> header shall define the **rusage** structure, which shall include at least the  
12884 following members:

12885 **struct timeval** **ru\_utime** User time used.

12886 **struct timeval** **ru\_stime** System time used.

12887 The <sys/resource.h> header shall define the **timeval** structure as described in <sys/time.h>.

12888 The <sys/resource.h> header shall define the following symbolic constants as possible values for  
12889 the *resource* argument of *getrlimit()* and *setrlimit()*:

12890 RLIMIT\_CORE Limit on size of **core** file.

12891 RLIMIT\_CPU Limit on CPU time per process.

12892	RLIMIT_DATA	Limit on data segment size.
12893	RLIMIT_FSIZE	Limit on file size.
12894	RLIMIT_NOFILE	Limit on number of open files.
12895	RLIMIT_STACK	Limit on stack size.
12896	RLIMIT_AS	Limit on address space size.
12897	The following shall be declared as functions and may also be defined as macros. Function	
12898	prototypes shall be provided.	
12899	<code>int getpriority(int, id_t);</code>	
12900	<code>int getrlimit(int, struct rlimit *);</code>	
12901	<code>int getrusage(int, struct rusage *);</code>	
12902	<code>int setpriority(int, id_t, int);</code>	
12903	<code>int setrlimit(int, const struct rlimit *);</code>	
12904	The <sys/resource.h> header shall define the <code>id_t</code> type through <b>typedef</b> , as described in	
12905	<sys/types.h>.	
12906	Inclusion of the <sys/resource.h> header may also make visible all symbols from <sys/time.h>.	
12907	<b>APPLICATION USAGE</b>	
12908	None.	
12909	<b>RATIONALE</b>	
12910	None.	
12911	<b>FUTURE DIRECTIONS</b>	
12912	None.	
12913	<b>SEE ALSO</b>	
12914	<a href="#">&lt;sys/time.h&gt;</a> , <a href="#">&lt;sys/types.h&gt;</a>	
12915	XSH <a href="#">getpriority()</a> , <a href="#">getrlimit()</a> , <a href="#">getrusage()</a>	
12916	<b>CHANGE HISTORY</b>	
12917	First released in Issue 4, Version 2.	
12918	<b>Issue 5</b>	
12919	Large File System extensions are added.	
12920	<b>Issue 7</b>	
12921	This reference page is clarified with respect to macros and symbolic constants.	

12922 **NAME**

12923 sys/select.h — select types

12924 **SYNOPSIS**

12925 #include <sys/select.h>

12926 **DESCRIPTION**

12927 The <sys/select.h> header shall define the **timeval** structure, which shall include at least the  
12928 following members:

12929	time_t	tv_sec	Seconds.
12930	suseconds_t	tv_usec	Microseconds.

12931 The <sys/select.h> header shall define the **time\_t** and **suseconds\_t** types as described in  
12932 <sys/types.h>.

12933 The <sys/select.h> header shall define the **sigset\_t** type as described in <signal.h>.

12934 The <sys/select.h> header shall define the **timespec** structure as described in <time.h>.

12935 The <sys/select.h> header shall define the **fd\_set** type as a structure.

12936 The <sys/select.h> header shall define the following symbolic constant, which shall have a value  
12937 suitable for use in #if preprocessing directives:

12938 FD\_SETSIZE Maximum number of file descriptors in an **fd\_set** structure.

12939 The following shall be declared as functions, defined as macros, or both. If functions are  
12940 declared, function prototypes shall be provided.

```
12941 void FD_CLR(int, fd_set *);
12942 int FD_ISSET(int, fd_set *);
12943 void FD_SET(int, fd_set *);
12944 void FD_ZERO(fd_set *);
```

12945 If implemented as macros, these may evaluate their arguments more than once, so applications  
12946 should ensure that the arguments they supply are never expressions with side-effects.

12947 The following shall be declared as functions and may also be defined as macros. Function  
12948 prototypes shall be provided.

```
12949 int pselect(int, fd_set *restrict, fd_set *restrict, fd_set *restrict,
12950             const struct timespec *restrict, const sigset_t *restrict);
12951 int select(int, fd_set *restrict, fd_set *restrict, fd_set *restrict,
12952            struct timeval *restrict);
```

12953 Inclusion of the <sys/select.h> header may make visible all symbols from the headers  
12954 <signal.h> and <time.h>.

12955 **APPLICATION USAGE**

12956 None.

12957 **RATIONALE**

12958 None.

12959 **FUTURE DIRECTIONS**

12960 None.

12961 **SEE ALSO**12962 [<signal.h>](#), [<sys/time.h>](#), [<sys/types.h>](#), [<time.h>](#)12963 XSH [pselect\(\)](#)12964 **CHANGE HISTORY**

12965 First released in Issue 6. Derived from IEEE Std 1003.1g-2000.

12966 The requirement for the **fd\_set** structure to have a member *fds\_bits* has been removed as per The  
12967 Open Group Base Resolution bwg2001-005.12968 **Issue 7**

12969 SD5-XBD-ERN-6 is applied, reordering the DESCRIPTION.

12970 This reference page is clarified with respect to macros and symbolic constants.



12971 **NAME**

12972 sys/sem.h — XSI semaphore facility

12973 **SYNOPSIS**12974 XSI `#include <sys/sem.h>`12975 **DESCRIPTION**12976 The **<sys/sem.h>** header shall define the following symbolic constant for use as a semaphore  
12977 operation flag:

12978 SEM\_UNDO Set up adjust on exit entry.

12979 The **<sys/sem.h>** header shall define the following symbolic constants for use as commands for  
12980 the *semctl()* function:12981 GETNCNT Get *semmcnt*.12982 GETPID Get *sempid*.12983 GETVAL Get *semval*.12984 GETALL Get all cases of *semval*.12985 GETZCNT Get *semzcnt*.12986 SETVAL Set *semval*.12987 SETALL Set all cases of *semval*.12988 The **<sys/sem.h>** header shall define the **semid\_ds** structure, which shall include the following  
12989 members:

12990 struct ipc\_perm sem\_perm Operation permission structure.

12991 unsigned short sem\_nsems Number of semaphores in set.

12992 time\_t sem\_otime Last *semop()* time.12993 time\_t sem\_ctime Last time changed by *semctl()*.12994 The **<sys/sem.h>** header shall define the **pid\_t**, **size\_t**, and **time\_t** types as described in  
12995 **<sys/types.h>**.12996 A semaphore shall be represented by an anonymous structure, which shall include the following  
12997 members:

12998 unsigned short semval Semaphore value.

12999 pid\_t sempid Process ID of last operation.

13000 unsigned short semncnt Number of processes waiting for *semval*  
13001 to become greater than current value.13002 unsigned short semzcnt Number of processes waiting for *semval*  
13003 to become 0.13004 The **<sys/sem.h>** header shall define the **sembuf** structure, which shall include the following  
13005 members:

13006 unsigned short sem\_num Semaphore number.

13007 short sem\_op Semaphore operation.

13008 short sem\_flg Operation flags.

13009 The following shall be declared as functions and may also be defined as macros. Function  
13010 prototypes shall be provided.

13011 int semctl(int, int, int, ...);

13012           int     semget(key\_t, int, int);  
13013           int     semop(int, struct sembuf \*, size\_t);

13014           In addition, the <sys/sem.h> header shall include the <sys/ipc.h> header.

#### 13015 APPLICATION USAGE

13016           None.

#### 13017 RATIONALE

13018           None.

#### 13019 FUTURE DIRECTIONS

13020           None.

#### 13021 SEE ALSO

13022           <sys/ipc.h>, <sys/types.h>

13023           XSH *semctl()*, *semget()*, *semop()*

#### 13024 CHANGE HISTORY

13025           First released in Issue 2. Derived from System V Release 2.0.

#### 13026 Issue 7

13027           Austin Group Interpretation 1003.1-2001 #179 is applied.

13028           This reference page is clarified with respect to macros and symbolic constants.

## 13029 NAME

13030 sys/shm.h — XSI shared memory facility

## 13031 SYNOPSIS

13032 XSI `#include <sys/shm.h>`

## 13033 DESCRIPTION

13034 The <sys/shm.h> header shall define the following symbolic constants:

13035 SHM\_RDONLY Attach read-only (else read-write).

13036 SHM\_RND Round attach address to SHMLBA.

13037 SHMLBA Segment low boundary address multiple.

13038 The <sys/shm.h> header shall define the following data types through **typedef**:

13039 **shmatt\_t** Unsigned integer used for the number of current attaches that must be able to  
13040 store values at least as large as a type **unsigned short**.

13041 The <sys/shm.h> header shall define the **shmid\_ds** structure, which shall include the following  
13042 members:

13043	struct ipc_perm	shm_perm	Operation permission structure.
13044	size_t	shm_segsz	Size of segment in bytes.
13045	pid_t	shm_lpid	Process ID of last shared memory operation.
13046	pid_t	shm_cpid	Process ID of creator.
13047	shmatt_t	shm_nattch	Number of current attaches.
13048	time_t	shm_atime	Time of last <i>shmat</i> ( ).
13049	time_t	shm_dtime	Time of last <i>shmdt</i> ( ).
13050	time_t	shm_ctime	Time of last change by <i>shmctl</i> ( ).

13051 The <sys/shm.h> header shall define the **pid\_t**, **size\_t**, and **time\_t** types as described in  
13052 <sys/types.h>.

13053 The following shall be declared as functions and may also be defined as macros. Function  
13054 prototypes shall be provided.

```
13055 void *shmat(int, const void *, int);
13056 int shmctl(int, int, struct shmid_ds *);
13057 int shmdt(const void *);
13058 int shmget(key_t, size_t, int);
```

13059 In addition, the <sys/shm.h> header shall include the <sys/ipc.h> header.

## 13060 APPLICATION USAGE

13061 None.

## 13062 RATIONALE

13063 None.

## 13064 FUTURE DIRECTIONS

13065 None.

## 13066 SEE ALSO

13067 <sys/ipc.h>, <sys/types.h>

13068 XSH *shmat*( ), *shmctl*( ), *shmdt*( ), *shmget*( )

13069 **CHANGE HISTORY**

13070 First released in Issue 2. Derived from System V Release 2.0.

13071 **Issue 5**

13072 The type of *shm\_segsz* is changed from **int** to **size\_t**.

13073 **Issue 7**

13074 Austin Group Interpretation 1003.1-2001 #179 is applied.

13075 This reference page is clarified with respect to macros and symbolic constants.

13076 **NAME**

13077 sys/socket.h — main sockets header

13078 **SYNOPSIS**

13079 #include &lt;sys/socket.h&gt;

13080 **DESCRIPTION**13081 The <sys/socket.h> header shall define the **socklen\_t** type, which is an integer type of width of  
13082 at least 32 bits; see APPLICATION USAGE.13083 The <sys/socket.h> header shall define the **sa\_family\_t** unsigned integer type.13084 The <sys/socket.h> header shall define the **sockaddr** structure, which shall include at least the  
13085 following members:13086 sa\_family\_t sa\_family Address family.  
13087 char sa\_data[] Socket address (variable-length data).13088 The **sockaddr** structure is used to define a socket address which is used in the *bind()*, *connect()*,  
13089 *getpeername()*, *getsockname()*, *recvfrom()*, and *sendto()* functions.13090 The <sys/socket.h> header shall define the **sockaddr\_storage** structure, which shall be:

- 13091
- Large enough to accommodate all supported protocol-specific address structures
  - Aligned at an appropriate boundary so that pointers to it can be cast as pointers to  
13092 protocol-specific address structures and used to access the fields of those structures  
13093 without alignment problems
- 13094

13095 The **sockaddr\_storage** structure shall include at least the following members:

13096 sa\_family\_t ss\_family

13097 When a pointer to a **sockaddr\_storage** structure is cast as a pointer to a **sockaddr** structure, the  
13098 *ss\_family* field of the **sockaddr\_storage** structure shall map onto the *sa\_family* field of the  
13099 **sockaddr** structure. When a pointer to a **sockaddr\_storage** structure is cast as a pointer to a  
13100 protocol-specific address structure, the *ss\_family* field shall map onto a field of that structure that  
13101 is of type **sa\_family\_t** and that identifies the protocol's address family.13102 The <sys/socket.h> header shall define the **msghdr** structure, which shall include at least the  
13103 following members:13104 void \*msg\_name Optional address.  
13105 socklen\_t msg\_namelen Size of address.  
13106 struct iovec \*msg\_iov Scatter/gather array.  
13107 int msg\_iovlen Members in *msg\_iov*.  
13108 void \*msg\_control Ancillary data; see below.  
13109 socklen\_t msg\_controllen Ancillary data buffer *len*.  
13110 int msg\_flags Flags on received message.13111 The **msghdr** structure is used to minimize the number of directly supplied parameters to the  
13112 *recvmsg()* and *sendmsg()* functions. This structure is used as a *value-result* parameter in the  
13113 *recvmsg()* function and *value* only for the *sendmsg()* function.13114 The <sys/socket.h> header shall define the **iovec** structure as described in <sys/uio.h>.13115 The <sys/socket.h> header shall define the **cmsg\_hdr** structure, which shall include at least the  
13116 following members:13117 socklen\_t cmsg\_len Data byte count, including the **cmsg\_hdr**.  
13118 int cmsg\_level Originating protocol.

13119           int            cmsg\_type    Protocol-specific type.

13120   The **cmsg\_hdr** structure is used for storage of ancillary data object information.

13121   Ancillary data consists of a sequence of pairs, each consisting of a **cmsg\_hdr** structure followed  
 13122   by a data array. The data array contains the ancillary data message, and the **cmsg\_hdr** structure  
 13123   contains descriptive information that allows an application to correctly parse the data.

13124   The values for *cmsg\_level* shall be legal values for the *level* argument to the *getsockopt()* and  
 13125   *setsockopt()* functions. The system documentation shall specify the *cmsg\_type* definitions for the  
 13126   supported protocols.

13127   Ancillary data is also possible at the socket level. The **<sys/socket.h>** header shall define the  
 13128   following symbolic constant for use as the *cmsg\_type* value when *cmsg\_level* is SOL\_SOCKET:

13129   SCM\_RIGHTS           Indicates that the data array contains the access rights to be sent or  
 13130                           received.

13131   The **<sys/socket.h>** header shall define the following macros to gain access to the data arrays in  
 13132   the ancillary data associated with a message header:

13133   CMMSG\_DATA(*cmsg*)  
 13134       If the argument is a pointer to a **cmsg\_hdr** structure, this macro shall return an unsigned  
 13135       character pointer to the data array associated with the **cmsg\_hdr** structure.

13136   CMMSG\_NXTHDR(*mhdr, cmsg*)  
 13137       If the first argument is a pointer to a **msghdr** structure and the second argument is a pointer  
 13138       to a **cmsg\_hdr** structure in the ancillary data pointed to by the *msg\_control* field of that  
 13139       **msghdr** structure, this macro shall return a pointer to the next **cmsg\_hdr** structure, or a null  
 13140       pointer if this structure is the last **cmsg\_hdr** in the ancillary data.

13141   CMMSG\_FIRSTHDR(*mhdr*)  
 13142       If the argument is a pointer to a **msghdr** structure, this macro shall return a pointer to the  
 13143       first **cmsg\_hdr** structure in the ancillary data associated with this **msghdr** structure, or a null  
 13144       pointer if there is no ancillary data associated with the **msghdr** structure.

13145   The **<sys/socket.h>** header shall define the **linger** structure, which shall include at least the  
 13146   following members:

13147   int   l\_onoff    Indicates whether linger option is enabled.  
 13148   int   l\_linger   Linger time, in seconds.

13149   The **<sys/socket.h>** header shall define the following symbolic constants with distinct values:

13150   SOCK\_DGRAM        Datagram socket.

13151   RS   SOCK\_RAW       Raw Protocol Interface.

13152   SOCK\_SEQPACKET   Sequenced-packet socket.

13153   SOCK\_STREAM       Byte-stream socket.

13154   The **<sys/socket.h>** header shall define the following symbolic constant for use as the *level*  
 13155   argument of *setsockopt()* and *getsockopt()*.

13156   SOL\_SOCKET        Options to be accessed at socket level, not protocol level.

13157   The **<sys/socket.h>** header shall define the following symbolic constants with distinct values for  
 13158   use as the *option\_name* argument in *getsockopt()* or *setsockopt()* calls (see XSH [Section 2.10.16](#), on  
 13159   page 528):

13160	SO_ACCEPTCONN	Socket is accepting connections.
13161	SO_BROADCAST	Transmission of broadcast messages is supported.
13162	SO_DEBUG	Debugging information is being recorded.
13163	SO_DONTROUTE	Bypass normal routing.
13164	SO_ERROR	Socket error status.
13165	SO_KEEPALIVE	Connections are kept alive with periodic messages.
13166	SO_LINGER	Socket lingers on close.
13167	SO_OOBLIN	Out-of-band data is transmitted in line.
13168	SO_RCVBUF	Receive buffer size.
13169	SO_RCVLOWAT	Receive “low water mark”.
13170	SO_RCVTIMEO	Receive timeout.
13171	SO_REUSEADDR	Reuse of local addresses is supported.
13172	SO_SNDBUF	Send buffer size.
13173	SO_SNDLOWAT	Send “low water mark”.
13174	SO_SNDTIMEO	Send timeout.
13175	SO_TYPE	Socket type.
13176	The <sys/socket.h> header shall define the following symbolic constant for use as the maximum	
13177	<i>backlog</i> queue length which may be specified by the <i>backlog</i> field of the <i>listen()</i> function:	
13178	SOMAXCONN	The maximum <i>backlog</i> queue length.
13179	The <sys/socket.h> header shall define the following symbolic constants with distinct values for	
13180	use as the valid values for the <i>msg_flags</i> field in the <b>msghdr</b> structure, or the <i>flags</i> parameter in	
13181	<i>recv()</i> , <i>recvfrom()</i> , <i>recvmsg()</i> , <i>send()</i> , <i>sendmsg()</i> , or <i>sendto()</i> calls:	
13182	MSG_CTRUNC	Control data truncated.
13183	MSG_DONTROUTE	Send without using routing tables.
13184	MSG_EOR	Terminates a record (if supported by the protocol).
13185	MSG_OOB	Out-of-band data.
13186	MSG_NOSIGNAL	No SIGPIPE generated when an attempt to send is made on a stream-
13187		oriented socket that is no longer connected.
13188	MSG_PEEK	Leave received data in queue.
13189	MSG_TRUNC	Normal data truncated.
13190	MSG_WAITALL	Attempt to fill the read buffer.
13191	The <sys/socket.h> header shall define the following symbolic constants with distinct values:	
13192	AF_INET	Internet domain sockets for use with IPv4 addresses.
13193	IP6 AF_INET6	Internet domain sockets for use with IPv6 addresses.
13194	AF_UNIX	UNIX domain sockets.

13195 AF\_UNSPEC Unspecified.

13196 The value of AF\_UNSPEC shall be 0. +

13197 The <sys/socket.h> header shall define the following symbolic constants with distinct values:

13198 SHUT\_RD Disables further receive operations.

13199 SHUT\_RDWR Disables further send and receive operations.

13200 SHUT\_WR Disables further send operations.

13201 The <sys/socket.h> header shall define the **size\_t** and **ssize\_t** types as described in

13202 <sys/types.h>.

13203 The following shall be declared as functions and may also be defined as macros. Function

13204 prototypes shall be provided.

```

13205 int accept(int, struct sockaddr *restrict, socklen_t *restrict);
13206 int bind(int, const struct sockaddr *, socklen_t);
13207 int connect(int, const struct sockaddr *, socklen_t);
13208 int getpeername(int, struct sockaddr *restrict, socklen_t *restrict);
13209 int getsockname(int, struct sockaddr *restrict, socklen_t *restrict);
13210 int getsockopt(int, int, int, void *restrict, socklen_t *restrict);
13211 int listen(int, int);
13212 ssize_t recv(int, void *, size_t, int);
13213 ssize_t recvfrom(int, void *restrict, size_t, int,
13214 struct sockaddr *restrict, socklen_t *restrict);
13215 ssize_t recvmsg(int, struct msghdr *, int);
13216 ssize_t send(int, const void *, size_t, int);
13217 ssize_t sendmsg(int, const struct msghdr *, int);
13218 ssize_t sendto(int, const void *, size_t, int, const struct sockaddr *,
13219 socklen_t);
13220 int setsockopt(int, int, int, const void *, socklen_t);
13221 int shutdown(int, int);
13222 int socketatmark(int);
13223 int socket(int, int, int);
13224 int socketpair(int, int, int, int [2]);

```

13225 Inclusion of <sys/socket.h> may also make visible all symbols from <sys/uio.h>.

## 13226 APPLICATION USAGE

13227 To forestall portability problems, it is recommended that applications not use values larger than

13228  $2^{31} - 1$  for the **socklen\_t** type.

13229 The **sockaddr\_storage** structure solves the problem of declaring storage for automatic variables

13230 which is both large enough and aligned enough for storing the socket address data structure of

13231 any family. For example, code with a file descriptor and without the context of the address

13232 family can pass a pointer to a variable of this type, where a pointer to a socket address structure

13233 is expected in calls such as *getpeername()*, and determine the address family by accessing the

13234 received content after the call.

13235 The example below illustrates a data structure which aligns on a 64-bit boundary. An

13236 implementation-defined field *\_ss\_align* following *\_ss\_pad1* is used to force a 64-bit alignment

13237 which covers proper alignment good enough for needs of at least **sockaddr\_in6** (IPv6) and

13238 **sockaddr\_in** (IPv4) address data structures. The size of padding field *\_ss\_pad1* depends on the

13239 chosen alignment boundary. The size of padding field *\_ss\_pad2* depends on the value of overall

13240 size chosen for the total size of the structure. This size and alignment are represented in the



above example by implementation-defined (not required) constants `_SS_MAXSIZE` (chosen value 128) and `_SS_ALIGNMENT` (with chosen value 8). Constants `_SS_PAD1SIZE` (derived value 6) and `_SS_PAD2SIZE` (derived value 112) are also for illustration and not required. The implementation-defined definitions and structure field names above start with an <underscore> to denote implementation private name space. Portable code is not expected to access or reference those fields or constants.

```

/*
 * Desired design of maximum size and alignment.
 */
#define _SS_MAXSIZE 128
/* Implementation-defined maximum size. */
#define _SS_ALIGNSIZE (sizeof(int64_t))
/* Implementation-defined desired alignment. */

/*
 * Definitions used for sockaddr_storage structure paddings design.
 */
#define _SS_PAD1SIZE (_SS_ALIGNSIZE - sizeof(sa_family_t))
#define _SS_PAD2SIZE (_SS_MAXSIZE - (sizeof(sa_family_t)+ \
                                   _SS_PAD1SIZE + _SS_ALIGNSIZE))
struct sockaddr_storage {
    sa_family_t  ss_family; /* Address family. */
/*
 * Following fields are implementation-defined.
 */
    char _ss_pad1[_SS_PAD1SIZE];
    /* 6-byte pad; this is to make implementation-defined
       pad up to alignment field that follows explicit in
       the data structure. */
    int64_t _ss_align; /* Field to force desired structure
                       storage alignment. */
    char _ss_pad2[_SS_PAD2SIZE];
    /* 112-byte pad to achieve desired size,
       _SS_MAXSIZE value minus size of ss_family
       __ss_pad1, __ss_align fields is 112. */
};

```

#### RATIONALE

None.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

<sys/types.h>, <sys/uio.h>

XSH *accept()*, *bind()*, *connect()*, *getpeername()*, *getsockname()*, *getsockopt()*, *listen()*, *recv()*, *recvfrom()*, *recvmsg()*, *send()*, *sendmsg()*, *sendto()*, *setsockopt()*, *shutdown()*, *socketatmark()*, *socket()*, *socketpair()*

#### CHANGE HISTORY

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

The **restrict** keyword is added to the prototypes for *accept()*, *getpeername()*, *getsockname()*, *getsockopt()*, and *recvfrom()*.

13289 **Issue 7**

13290 SD5-XBD-ERN-56 is applied, adding a reference to **<sys/types.h>** for the **ssize\_t** type.

13291 SD5-XBD-ERN-62 is applied.

13292 The MSG\_NOSIGNAL symbolic constant is added from The Open Group Technical Standard,  
13293 2006, Extended API Set Part 2.

13294 This reference page is clarified with respect to macros and symbolic constants, and a declaration  
13295 for the **size\_t** type is added.

13296 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0067 [355] is applied.

13297 **NAME**13298 sys/stat.h — data returned by the `stat()` function13299 **SYNOPSIS**

13300 #include &lt;sys/stat.h&gt;

13301 **DESCRIPTION**13302 The <sys/stat.h> header shall define the structure of the data returned by the `fstat()`, `lstat()`, and  
13303 `stat()` functions.13304 The <sys/stat.h> header shall define the **stat** structure, which shall include at least the following  
13305 members:

13306	<code>dev_t st_dev</code>	Device ID of device containing file.
13307	<code>ino_t st_ino</code>	File serial number.
13308	<code>mode_t st_mode</code>	Mode of file (see below).
13309	<code>nlink_t st_nlink</code>	Number of hard links to the file.
13310	<code>uid_t st_uid</code>	User ID of file.
13311	<code>gid_t st_gid</code>	Group ID of file.
13312	XSI <code>dev_t st_rdev</code>	Device ID (if file is character or block special).
13313	<code>off_t st_size</code>	For regular files, the file size in bytes.
13314		For symbolic links, the length in bytes of the
13315		pathname contained in the symbolic link.
13316	SHM	For a shared memory object, the length in bytes.
13317	TYM	For a typed memory object, the length in bytes.
13318		For other file types, the use of this field is
13319		unspecified.
13320	<code>struct timespec st_atim</code>	Last data access timestamp.
13321	<code>struct timespec st_mtim</code>	Last data modification timestamp.
13322	<code>struct timespec st_ctim</code>	Last file status change timestamp.
13323	XSI <code>blksize_t st_blksize</code>	A file system-specific preferred I/O block size
13324		for this object. In some file system types, this
13325		may vary from file to file.
13326	<code>blkcnt_t st_blocks</code>	Number of blocks allocated for this object.

13327 The `st_ino` and `st_dev` fields taken together uniquely identify the file within the system.13328 XSI The <sys/stat.h> header shall define the **blkcnt\_t**, **blksize\_t**, **dev\_t**, **ino\_t**, **mode\_t**, **nlink\_t**,  
13329 **uid\_t**, **gid\_t**, **off\_t**, and **time\_t** types as described in <sys/types.h>.13330 The <sys/stat.h> header shall define the **timespec** structure as described in <time.h>. Times  
13331 shall be given in seconds since the Epoch.13332 Which structure members have meaningful values depends on the type of file. For further  
13333 information, see the descriptions of `fstat()`, `lstat()`, and `stat()` in the System Interfaces volume of  
13334 POSIX.1-2008.13335 For compatibility with earlier versions of this standard, the `st_atime` macro shall be defined with  
13336 the value `st_atim.tv_sec`. Similarly, `st_ctime` and `st_mtime` shall be defined as macros with the  
13337 values `st_ctim.tv_sec` and `st_mtim.tv_sec`, respectively.

The <sys/stat.h> header shall define the following symbolic constants for the file types encoded in type **mode\_t**. The values shall be suitable for use in **#if** preprocessing directives:

S_IFMT	Type of file.
S_IFBLK	Block special.
S_IFCHR	Character special.
S_IFIFO	FIFO special.
S_IFREG	Regular.
S_IFDIR	Directory.
S_IFLNK	Symbolic link.
S_IFSOCK	Socket.

The <sys/stat.h> header shall define the following symbolic constants for the file mode bits encoded in type **mode\_t**, with the indicated numeric values. These macros shall expand to an expression which has a type that allows them to be used, either singly or OR'ed together, as the third argument to *open()* without the need for a **mode\_t** cast. The values shall be suitable for use in **#if** preprocessing directives.

Name	Numeric Value	Description
S_IRWXU	0700	Read, write, execute/search by owner.
S_IRUSR	0400	Read permission, owner.
S_IWUSR	0200	Write permission, owner.
S_IXUSR	0100	Execute/search permission, owner.
S_IRWXG	070	Read, write, execute/search by group.
S_IRGRP	040	Read permission, group.
S_IWGRP	020	Write permission, group.
S_IXGRP	010	Execute/search permission, group.
S_IRWXO	07	Read, write, execute/search by others.
S_IROTH	04	Read permission, others.
S_IWOTH	02	Write permission, others.
S_IXOTH	01	Execute/search permission, others.
S_ISUID	04000	Set-user-ID on execution.
S_ISGID	02000	Set-group-ID on execution.
S_ISVTX	01000	On directories, restricted deletion flag.

The following macros shall be provided to test whether a file is of the specified type. The value *m* supplied to the macros is the value of *st\_mode* from a **stat** structure. The macro shall evaluate to a non-zero value if the test is true; 0 if the test is false.

S_ISBLK( <i>m</i> )	Test for a block special file.
S_ISCHR( <i>m</i> )	Test for a character special file.
S_ISDIR( <i>m</i> )	Test for a directory.
S_ISFIFO( <i>m</i> )	Test for a pipe or FIFO special file.
S_ISREG( <i>m</i> )	Test for a regular file.

13377		<code>S_ISLNK(<i>m</i>)</code>	Test for a symbolic link.
13378		<code>S_ISSOCK(<i>m</i>)</code>	Test for a socket.
13379		The implementation may implement message queues, semaphores, or shared memory objects as distinct file types. The following macros shall be provided to test whether a file is of the specified type. The value of the <i>buf</i> argument supplied to the macros is a pointer to a <b>stat</b> structure. The macro shall evaluate to a non-zero value if the specified object is implemented as a distinct file type and the specified file type is contained in the <b>stat</b> structure referenced by <i>buf</i> . Otherwise, the macro shall evaluate to zero.	
13380			
13381			
13382			
13383			
13384			
13385		<code>S_TYPEISMQ(<i>buf</i>)</code>	Test for a message queue.
13386		<code>S_TYPEISSEM(<i>buf</i>)</code>	Test for a semaphore.
13387		<code>S_TYPEISSHM(<i>buf</i>)</code>	Test for a shared memory object.
13388	TYM	The implementation may implement typed memory objects as distinct file types, and the following macro shall test whether a file is of the specified type. The value of the <i>buf</i> argument supplied to the macros is a pointer to a <b>stat</b> structure. The macro shall evaluate to a non-zero value if the specified object is implemented as a distinct file type and the specified file type is contained in the <b>stat</b> structure referenced by <i>buf</i> . Otherwise, the macro shall evaluate to zero.	
13389			
13390			
13391			
13392			
13393		<code>S_TYPEISTMO(<i>buf</i>)</code>	Test macro for a typed memory object.
13394		The <sys/stat.h> header shall define the following symbolic constants as distinct integer values outside of the range [0,999 999 999], for use with the <i>futimens()</i> and <i>utimensat()</i> functions:	
13395			
13396		<code>UTIME_NOW</code>	
13397		<code>UTIME_OMIT</code>	
13398		The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.	
13399			
13400		<code>int</code>	<code>chmod(const char *, mode_t);</code>
13401		<code>int</code>	<code>fchmod(int, mode_t);</code>
13402		<code>int</code>	<code>fchmodat(int, const char *, mode_t, int);</code>
13403		<code>int</code>	<code>fstat(int, struct stat *);</code>
13404		<code>int</code>	<code>fstatat(int, const char *restrict, struct stat *restrict, int);</code>
13405		<code>int</code>	<code>futimens(int, const struct timespec [2]);</code>
13406		<code>int</code>	<code>lstat(const char *restrict, struct stat *restrict);</code>
13407		<code>int</code>	<code>mkdir(const char *, mode_t);</code>
13408		<code>int</code>	<code>mkdirat(int, const char *, mode_t);</code>
13409		<code>int</code>	<code>mkfifo(const char *, mode_t);</code>
13410		<code>int</code>	<code>mkfifoat(int, const char *, mode_t);</code>
13411	XSI	<code>int</code>	<code>mknod(const char *, mode_t, dev_t);</code>
13412		<code>int</code>	<code>mknodat(int, const char *, mode_t, dev_t);</code>
13413		<code>int</code>	<code>stat(const char *restrict, struct stat *restrict);</code>
13414		<code>mode_t</code>	<code>umask(mode_t);</code>
13415		<code>int</code>	<code>utimensat(int, const char *, const struct timespec [2], int);</code>
13416		Inclusion of the <sys/stat.h> header may make visible all symbols from the <time.h> header. +	

## APPLICATION USAGE

Use of the macros is recommended for determining the type of a file.

## RATIONALE

A conforming C-language application must include **<sys/stat.h>** for functions that have arguments or return values of type **mode\_t**, so that symbolic values for that type can be used. An alternative would be to require that these constants are also defined by including **<sys/types.h>**.

The **S\_ISUID** and **S\_ISGID** bits may be cleared on any write, not just on *open()*, as some historical implementations do.

System calls that update the time entry fields in the **stat** structure must be documented by the implementors. POSIX-conforming systems should not update the time entry fields for functions listed in the System Interfaces volume of POSIX.1-2008 unless the standard requires that they do, except in the case of documented extensions to the standard.

Upon assignment, file timestamps are immediately converted to the resolution of the file system by truncation (i.e., the recorded time can be older than the actual time). For example, if the file system resolution is 1 microsecond, then a conforming *stat()* must always return an *st\_mtim.tv\_nsec* that is a multiple of 1000. Some older implementations returned higher-resolution timestamps while the *inode* information was cached, and then spontaneously truncated the *tv\_nsec* fields when they were stored to and retrieved from disk, but this behavior does not conform.

Note that *st\_dev* must be unique within a Local Area Network (LAN) in a “system” made up of multiple computers’ file systems connected by a LAN.

Networked implementations of a POSIX-conforming system must guarantee that all files visible within the file tree (including parts of the tree that may be remotely mounted from other machines on the network) on each individual processor are uniquely identified by the combination of the *st\_ino* and *st\_dev* fields.

The unit for the *st\_blocks* member of the **stat** structure is not defined within POSIX.1-2008. In some implementations it is 512 bytes. It may differ on a file system basis. There is no correlation between values of the *st\_blocks* and *st\_blksize*, and the *f\_bsize* (from **<sys/statvfs.h>**) structure members.

Traditionally, some implementations defined the multiplier for *st\_blocks* in **<sys/param.h>** as the symbol **DEV\_BSIZE**.

Some earlier versions of this standard did not specify values for the file mode bit macros. The expectation was that some implementors might choose to use a different encoding for these bits than the traditional one, and that new applications would use symbolic file modes instead of numeric. This version of the standard specifies the traditional encoding, in recognition that nearly 20 years after the first publication of this standard numeric file modes are still in widespread use by application developers, and that all conforming implementations still use the traditional encoding.

## FUTURE DIRECTIONS

No new **S\_IFMT** symbolic names for the file type values of **mode\_t** will be defined by POSIX.1-2008; if new file types are required, they will only be testable through *S\_ISxx()* or *S\_TYPEISxxx()* macros instead.

## SEE ALSO

<sys/statvfs.h>, <sys/types.h>, <time.h>

XSH *chmod()*, *fchmod()*, *fstat()*, *fstatat()*, *futimens()*, *mkdir()*, *mkfifo()*, *mknod()*, *umask()*

## CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

### Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension.

The type of *st\_blksize* is changed from **long** to **blksize\_t**; the type of *st\_blocks* is changed from **long** to **blkcnt\_t**.

### Issue 6

The S\_TYPEISMQ(), S\_TYPEISSEM(), and S\_TYPEISSHM() macros are unconditionally mandated.

The Open Group Corrigendum U035/4 is applied. In the DESCRIPTION, the types **blksize\_t** and **blkcnt\_t** have been described.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **dev\_t**, **ino\_t**, **mode\_t**, **nlink\_t**, **uid\_t**, **gid\_t**, **off\_t**, and **time\_t** types are mandated.

S\_IFSOCK and S\_ISSOCK are added for sockets.

The description of **stat** structure members is changed to reflect contents when file type is a symbolic link.

The test macro S\_TYPEISTMO is added for alignment with IEEE Std 1003.1j-2000.

The **restrict** keyword is added to the prototypes for *lstat()* and *stat()*.

The *lstat()* function is made mandatory.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/17 is applied, adding text regarding the *st\_blocks* member of the **stat** structure to the RATIONALE.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/25 is applied, adding to the DESCRIPTION that the **timespec** structure may be defined as described in the <time.h> header.

### Issue 7

SD5-XSH-ERN-161 is applied, updating the DESCRIPTION to clarify that the descriptions of the interfaces should be consulted in order to determine which structure members have meaningful values.

The *fchmodat()*, *fstatat()*, *mkdirat()*, *mkfifoat()*, *mknodat()*, and *utimensat()* functions are added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

The *futimens()* function is added.

This reference page is clarified with respect to macros and symbolic constants.

Changes are made related to support for finegrained timestamps and the **UTIME\_NOW** and **UTIME\_OMIT** symbolic constants are added.

POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0068 [207] is applied.



13498 **NAME**

13499 sys/statvfs.h — VFS File System information structure

13500 **SYNOPSIS**

13501 #include &lt;sys/statvfs.h&gt;

13502 **DESCRIPTION**13503 The <sys/statvfs.h> header shall define the **statvfs** structure, which shall include at least the  
13504 following members:

13505	unsigned long	f_bsize	File system block size.
13506	unsigned long	f_frsize	Fundamental file system block size.
13507	fsblkcnt_t	f_blocks	Total number of blocks on file system in units of <i>f_frsize</i> .
13508	fsblkcnt_t	f_bfree	Total number of free blocks.
13509	fsblkcnt_t	f_bavail	Number of free blocks available to non-privileged process.
13510			
13511	fsfilcnt_t	f_files	Total number of file serial numbers.
13512	fsfilcnt_t	f_ffree	Total number of free file serial numbers.
13513	fsfilcnt_t	f_favail	Number of file serial numbers available to non-privileged process.
13514			
13515	unsigned long	f_fsid	File system ID.
13516	unsigned long	f_flag	Bit mask of <i>f_flag</i> values.
13517	unsigned long	f_namemax	Maximum filename length.

13518 The <sys/statvfs.h> header shall define the **fsblkcnt\_t** and **fsfilcnt\_t** types as described in  
13519 <sys/types.h>.13520 The <sys/statvfs.h> header shall define the following symbolic constants for the *f\_flag* member:

13521 ST\_RDONLY Read-only file system.

13522 ST\_NOSUID Does not support the semantics of the ST\_ISUID and ST\_ISGID file mode bits.

13523 The following shall be declared as functions and may also be defined as macros. Function  
13524 prototypes shall be provided.

13525 int fstatvfs(int, struct statvfs \*);

13526 int statvfs(const char \*restrict, struct statvfs \*restrict);

13527 **APPLICATION USAGE**

13528 None.

13529 **RATIONALE**

13530 None.

13531 **FUTURE DIRECTIONS**

13532 None.

13533 **SEE ALSO**

13534 &lt;sys/types.h&gt;

13535 XSH *fstatvfs()*13536 **CHANGE HISTORY**

13537 First released in Issue 4, Version 2.

13538 **Issue 5**13539 The type of *f\_blocks*, *f\_bfree*, and *f\_bavail* is changed from **unsigned long** to **fsblkcnt\_t**; the type of  
13540 *f\_files*, *f\_ffree*, and *f\_favail* is changed from **unsigned long** to **fsfilcnt\_t**.



13541 **Issue 6**

13542 The Open Group Corrigendum U035/5 is applied. In the DESCRIPTION, the types **fsblkcnt\_t**  
13543 and **fsfilcnt\_t** have been described.

13544 The **restrict** keyword is added to the prototype for *statvfs()*.

13545 IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/18 is applied, changing the description of  
13546 ST\_NOSUID from “Does not support *setuid()*/*setgid()* semantics” to “Does not support the  
13547 semantics of the ST\_ISUID and ST\_ISGID file mode bits”.

13548 **Issue 7**

13549 The <sys/statvfs.h> header is moved from the XSI option to the Base.

13550 This reference page is clarified with respect to macros and symbolic constants.

## 13551 NAME

13552 sys/time.h — time types

## 13553 SYNOPSIS

13554 XSI `#include <sys/time.h>`

## 13555 DESCRIPTION

13556 The **<sys/time.h>** header shall define the **timeval** structure, which shall include at least the  
 13557 following members:

13558	time_t	tv_sec	Seconds.
13559	suseconds_t	tv_usec	Microseconds.

13560 OB The **<sys/time.h>** header shall define the **itimerval** structure, which shall include at least the  
 13561 following members:

13562	struct timeval	it_interval	Timer interval.
13563	struct timeval	it_value	Current value.

13564 The **<sys/time.h>** header shall define the **time\_t** and **suseconds\_t** types as described in  
 13565 **<sys/types.h>**.

13566 The **<sys/time.h>** header shall define the **fd\_set** type as described in **<sys/select.h>**.

13567 OB The **<sys/time.h>** header shall define the following symbolic constants for the *which* argument of  
 13568 *getitimer()* and *setitimer()*:

13569	ITIMER_REAL	Decrements in real time.
13570	ITIMER_VIRTUAL	Decrements in process virtual time.
13571	ITIMER_PROF	Decrements both in process virtual time and when the system is running 13572 on behalf of the process.

13573 The **<sys/time.h>** header shall define the following as described in **<sys/select.h>**:

13574 *FD\_CLR()*  
 13575 *FD\_ISSET()*  
 13576 *FD\_SET()*  
 13577 *FD\_ZERO()*  
 13578 *FD\_SETSIZE*

13579 The following shall be declared as functions and may also be defined as macros. Function  
 13580 prototypes shall be provided.

13581 OB `int getitimer(int, struct itimerval *);`  
 13582 `int gettimeofday(struct timeval *restrict, void *restrict);`  
 13583 `int setitimer(int, const struct itimerval *restrict,`  
 13584 `struct itimerval *restrict);`  
 13585 `int select(int, fd_set *restrict, fd_set *restrict, fd_set *restrict,`  
 13586 `struct timeval *restrict);`  
 13587 `int utimes(const char *, const struct timeval [2]);`

13588 Inclusion of the **<sys/time.h>** header may make visible all symbols from the **<sys/select.h>**  
 13589 header.

13590 **APPLICATION USAGE**

13591 None.

13592 **RATIONALE**

13593 None.

13594 **FUTURE DIRECTIONS**

13595 None.

13596 **SEE ALSO**

13597 <sys/select.h>, <sys/types.h>

13598 XSH *futimens()*, *getitimer()*, *gettimeofday()*, *pselect()*

13599 **CHANGE HISTORY**

13600 First released in Issue 4, Version 2.

13601 **Issue 5**

13602 The type of *tv\_usec* is changed from **long** to **suseconds\_t**.

13603 **Issue 6**

13604 The **restrict** keyword is added to the prototypes for *gettimeofday()*, *select()*, and *setitimer()*.

13605 The note is added that inclusion of this header may also make symbols visible from  
13606 <sys/select.h>.

13607 The *utimes()* function is marked LEGACY.

13608 **Issue 7**

13609 This reference page is clarified with respect to macros and symbolic constants.

13610 **NAME**

13611 sys/times.h — file access and modification times structure

13612 **SYNOPSIS**

13613 #include &lt;sys/times.h&gt;

13614 **DESCRIPTION**13615 The <sys/times.h> header shall define the **tms** structure, which is returned by *times()* and shall  
13616 include at least the following members:

13617 clock\_t tms\_utime User CPU time.

13618 clock\_t tms\_stime System CPU time.

13619 clock\_t tms\_cutime User CPU time of terminated child processes.

13620 clock\_t tms\_cstime System CPU time of terminated child processes.

13621 The <sys/times.h> header shall define the **clock\_t** type as described in <sys/types.h>.13622 The following shall be declared as a function and may also be defined as a macro. A function  
13623 prototype shall be provided.

13624 clock\_t times(struct tms \*);

13625 **APPLICATION USAGE**

13626 None.

13627 **RATIONALE**

13628 None.

13629 **FUTURE DIRECTIONS**

13630 None.

13631 **SEE ALSO**

13632 &lt;sys/types.h&gt;

13633 XSH *times()*13634 **CHANGE HISTORY**

13635 First released in Issue 1. Derived from Issue 1 of the SVID.

13636 **NAME**

13637 sys/types.h — data types

13638 **SYNOPSIS**

13639 #include <sys/types.h>

13640 **DESCRIPTION**

13641 The <sys/types.h> header shall define at least the following types:

13642	<b>blkcnt_t</b>	Used for file block counts.
13643	<b>blksize_t</b>	Used for block sizes.
13644	<b>clock_t</b>	Used for system times in clock ticks or CLOCKS_PER_SEC; see <time.h>.
13645		
13646	<b>clockid_t</b>	Used for clock ID type in the clock and timer functions.
13647	<b>dev_t</b>	Used for device IDs.
13648	<b>fsblkcnt_t</b>	Used for file system block counts.
13649	<b>fsfilcnt_t</b>	Used for file system file counts.
13650	<b>gid_t</b>	Used for group IDs.
13651	<b>id_t</b>	Used as a general identifier; can be used to contain at least a <b>pid_t</b> ,
13652		<b>uid_t</b> , or <b>gid_t</b> .
13653	<b>ino_t</b>	Used for file serial numbers.
13654	XSI <b>key_t</b>	Used for XSI interprocess communication.
13655	<b>mode_t</b>	Used for some file attributes.
13656	<b>nlink_t</b>	Used for link counts.
13657	<b>off_t</b>	Used for file sizes.
13658	<b>pid_t</b>	Used for process IDs and process group IDs.
13659	<b>pthread_attr_t</b>	Used to identify a thread attribute object.
13660	<b>pthread_barrier_t</b>	Used to identify a barrier.
13661	<b>pthread_barrierattr_t</b>	Used to define a barrier attributes object.
13662	<b>pthread_cond_t</b>	Used for condition variables.
13663	<b>pthread_condattr_t</b>	Used to identify a condition attribute object.
13664	<b>pthread_key_t</b>	Used for thread-specific data keys.
13665	<b>pthread_mutex_t</b>	Used for mutexes.
13666	<b>pthread_mutexattr_t</b>	Used to identify a mutex attribute object.
13667	<b>pthread_once_t</b>	Used for dynamic package initialization.
13668	<b>pthread_rwlock_t</b>	Used for read-write locks.
13669	<b>pthread_rwlockattr_t</b>	Used for read-write lock attributes.
13670	<b>pthread_spinlock_t</b>	Used to identify a spin lock.

13671		<b>pthread_t</b>	Used to identify a thread.
13672		<b>size_t</b>	Used for sizes of objects.
13673		<b>ssize_t</b>	Used for a count of bytes or an error indication.
13674		<b>suseconds_t</b>	Used for time in microseconds. -
13675		<b>time_t</b>	Used for time in seconds.
13676		<b>timer_t</b>	Used for timer ID returned by <i>timer_create()</i> .
13677	OB TRC	<b>trace_attr_t</b>	Used to identify a trace stream attributes object
13678	OB TRC	<b>trace_event_id_t</b>	Used to identify a trace event type.
13679	OB TEF	<b>trace_event_set_t</b>	Used to identify a trace event type set.
13680	OB TRC	<b>trace_id_t</b>	Used to identify a trace stream.
13681		<b>uid_t</b>	Used for user IDs.
13682		All of the types shall be defined as arithmetic types of an appropriate length, with the following exceptions:	
13683			
13684		<b>pthread_attr_t</b>	
13685		<b>pthread_barrier_t</b>	
13686		<b>pthread_barrierattr_t</b>	
13687		<b>pthread_cond_t</b>	
13688		<b>pthread_condattr_t</b>	
13689		<b>pthread_key_t</b>	
13690		<b>pthread_mutex_t</b>	
13691		<b>pthread_mutexattr_t</b>	
13692		<b>pthread_once_t</b>	
13693		<b>pthread_rwlock_t</b>	
13694		<b>pthread_rwlockattr_t</b>	
13695		<b>pthread_spinlock_t</b>	
13696		<b>pthread_t</b>	
13697		<b>timer_t</b>	
13698	OB TRC	<b>trace_attr_t</b>	
13699		<b>trace_event_id_t</b>	
13700	OB TEF	<b>trace_event_set_t</b>	
13701	OB TRC	<b>trace_id_t</b>	
13702		Additionally:	
13703		• <b>mode_t</b> shall be an integer type.	
13704		• <b>dev_t</b> shall be an integer type.	
13705		• <b>nlink_t</b> , <b>uid_t</b> , <b>gid_t</b> , and <b>id_t</b> shall be integer types.	
13706		• <b>blkcnt_t</b> and <b>off_t</b> shall be signed integer types.	
13707		• <b>fsblkcnt_t</b> , <b>fsfilcnt_t</b> , and <b>ino_t</b> shall be defined as unsigned integer types.	
13708		• <b>size_t</b> shall be an unsigned integer type.	
13709		• <b>blksize_t</b> , <b>pid_t</b> , and <b>ssize_t</b> shall be signed integer types.	

13710 CX           • **clock\_t** shall be an integer or real-floating type. **time\_t** shall be an integer type.

13711           The type **ssize\_t** shall be capable of storing values at least in the range  $[-1, \{\text{SSIZE\_MAX}\}]$ .

13712 XSI           The type **suseconds\_t** shall be a signed integer type capable of storing values at least in the

13713           range  $[-1, 1\,000\,000]$ .

13714           The implementation shall support one or more programming environments in which the widths

13715           of **blksize\_t**, **pid\_t**, **size\_t**, **ssize\_t**, and **suseconds\_t** are no greater than the width of type **long**.

13716           The names of these programming environments can be obtained using the *confstr()* function or

13717           the *getconf* utility.

13718           There are no defined comparison or assignment operators for the following types:

13719           **pthread\_attr\_t**

13720           **pthread\_barrier\_t**

13721           **pthread\_barrierattr\_t**

13722           **pthread\_cond\_t**

13723           **pthread\_condattr\_t**

13724           **pthread\_mutex\_t**

13725           **pthread\_mutexattr\_t**

13726           **pthread\_rwlock\_t**

13727           **pthread\_rwlockattr\_t**

13728           **pthread\_spinlock\_t**

13729           **timer\_t**

13730 OB TRC       **trace\_attr\_t**

## 13731 APPLICATION USAGE

13732           None.

## 13733 RATIONALE

13734           None.

## 13735 FUTURE DIRECTIONS

13736           None.

## 13737 SEE ALSO

13738           <time.h>

13739           XSH *confstr()*

13740           XCU *getconf*

## 13741 CHANGE HISTORY

13742           First released in Issue 1. Derived from Issue 1 of the SVID.

### 13743 Issue 5

13744           The **clockid\_t** and **timer\_t** types are defined for alignment with the POSIX Realtime Extension.

13745           The types **blkcnt\_t**, **blksize\_t**, **fsblkcnt\_t**, **fsfilcnt\_t**, and **suseconds\_t** are added.

13746           Large File System extensions are added.

13747           Updated for alignment with the POSIX Threads Extension.

### 13748 Issue 6

13749           The **pthread\_barrier\_t**, **pthread\_barrierattr\_t**, and **pthread\_spinlock\_t** types are added for

13750           alignment with IEEE Std 1003.1j-2000.

13751           The margin code is changed from XSI to THR for the **pthread\_rwlock\_t** and

13752 **pthread\_rwlockattr\_t** types as Read-Write Locks have been absorbed into the POSIX Threads  
13753 option. The threads types are marked THR.

13754 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/26 is applied, adding **pthread\_t** to the list  
13755 of types that are not required to be arithmetic types, thus allowing **pthread\_t** to be defined as a  
13756 structure.

13757 **Issue 7**

13758 Austin Group Interpretation 1003.1-2001 #033 is applied, requiring **key\_t** to be an arithmetic  
13759 type.

13760 The Trace option types are marked obsolescent.

13761 The **clock\_t** and **id\_t** types are moved from the XSI option to the Base.

13762 The **pthread\_barrier\_t** and **pthread\_barrierattr\_t** types are moved from the Barriers option to  
13763 the Base.

13764 The **pthread\_spinlock\_t** type is moved from the Spin Locks option to the Base.

13765 Functionality relating to the Timers and Threads options is moved to the Base.

13766 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0069 [210], XBD/TC1-2008/0070 [28],  
13767 XBD/TC1-2008/0071 [376], XBD/TC1-2008/0072 [210], and XBD/TC1-2008/0073 [327] are  
13768 applied.



13769 **NAME**

13770 sys/uio.h — definitions for vector I/O operations

13771 **SYNOPSIS**13772 XSI `#include <sys/uio.h>`13773 **DESCRIPTION**13774 The <sys/uio.h> header shall define the **iovec** structure, which shall include at least the  
13775 following members:13776 `void *iov_base` Base address of a memory region for input or output.13777 `size_t iov_len` The size of the memory pointed to by *iov\_base*.13778 The <sys/uio.h> header uses the **iovec** structure for scatter/gather I/O.13779 The <sys/uio.h> header shall define the **ssize\_t** and **size\_t** types as described in <sys/types.h>.13780 The following shall be declared as functions and may also be defined as macros. Function  
13781 prototypes shall be provided.13782 `ssize_t readv(int, const struct iovec *, int);`13783 `ssize_t writev(int, const struct iovec *, int);`13784 **APPLICATION USAGE**13785 The implementation can put a limit on the number of scatter/gather elements which can be  
13786 processed in one call. The symbol {IOV\_MAX} defined in <limits.h> should always be used to  
13787 learn about the limits instead of assuming a fixed value.13788 **RATIONALE**13789 Traditionally, the maximum number of scatter/gather elements the system can process in one  
13790 call were described by the symbolic value {UIO\_MAXIOV}. In IEEE Std 1003.1-2001 this value is  
13791 replaced by the constant {IOV\_MAX} which can be found in <limits.h>.13792 **FUTURE DIRECTIONS**

13793 None.

13794 **SEE ALSO**

13795 &lt;limits.h&gt;, &lt;sys/types.h&gt;

13796 XSH *read()*, *readv()*, *write()*, *writev()*13797 **CHANGE HISTORY**

13798 First released in Issue 4, Version 2.

13799 **Issue 6**

13800 Text referring to scatter/gather I/O is added to the DESCRIPTION.

13801 **NAME**

13802 sys/un.h — definitions for UNIX domain sockets

13803 **SYNOPSIS**

13804 #include &lt;sys/un.h&gt;

13805 **DESCRIPTION**

13806 The <sys/un.h> header shall define the **sockaddr\_un** structure, which shall include at least the  
 13807 following members:

13808 sa\_family\_t sun\_family Address family.  
 13809 char sun\_path[] Socket pathname.

13810 The **sockaddr\_un** structure is used to store addresses for UNIX domain sockets. Pointers to this  
 13811 type shall be cast by applications to **struct sockaddr \*** for use with socket functions.

13812 The <sys/un.h> header shall define the **sa\_family\_t** type as described in <sys/socket.h>.

13813 **APPLICATION USAGE**

13814 The size of *sun\_path* has intentionally been left undefined. This is because different  
 13815 implementations use different sizes. For example, 4.3 BSD uses a size of 108, and 4.4 BSD uses a  
 13816 size of 104. Since most implementations originate from BSD versions, the size is typically in the  
 13817 range 92 to 108.

13818 Applications should not assume a particular length for *sun\_path* or assume that it can hold  
 13819 {\_POSIX\_PATH\_MAX} bytes (256).

13820 Although applications are required to initialize all members (including any non-standard ones) +  
 13821 of a **sockaddr\_in6** structure (see <netinet/in.h>, on page 305), the same is not required for the +  
 13822 **sockaddr\_un** structure, since historically many applications only initialized the standard +  
 13823 members. Despite this, applications are encouraged to initialize **sockaddr\_un** structures in a +  
 13824 manner similar to the required initialization of **sockaddr\_in6** structures.

13825 **RATIONALE**

13826 None.

13827 **FUTURE DIRECTIONS**

13828 None.

13829 **SEE ALSO**

13830 &lt;netinet/in.h&gt;, &lt;sys/socket.h&gt;

13831 XSH *bind()*, *socket()*, *socketpair()*13832 **CHANGE HISTORY**

13833 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

13834 **Issue 7**

13835 The value for {\_POSIX\_PATH\_MAX} is updated to 256.

13836 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0074 [355] is applied.

## 13837 NAME

13838 sys/utsname.h — system name structure

## 13839 SYNOPSIS

13840 #include <sys/utsname.h>

## 13841 DESCRIPTION

13842 The <sys/utsname.h> header shall define the structure **utsname** which shall include at least the  
13843 following members:

13844	char	sysname[ ]	Name of this implementation of the operating system.
13845	char	nodename[ ]	Name of this node within the communications
13846			network to which this node is attached, if any.
13847	char	release[ ]	Current release level of this implementation.
13848	char	version[ ]	Current version level of this release.
13849	char	machine[ ]	Name of the hardware type on which the system is running.

13850 The character arrays are of unspecified size, but the data stored in them shall be terminated by a  
13851 null byte.

13852 The following shall be declared as a function and may also be defined as a macro:

13853 int uname(struct utsname \*);

## 13854 APPLICATION USAGE

13855 None.

## 13856 RATIONALE

13857 None.

## 13858 FUTURE DIRECTIONS

13859 None.

## 13860 SEE ALSO

13861 XSH *uname()*

## 13862 CHANGE HISTORY

13863 First released in Issue 1. Derived from Issue 1 of the SVID.

## 13864 Issue 6

13865 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/27 is applied, changing the description of  
13866 *nodename* within the **utsname** structure from “an implementation-defined communications  
13867 network” to “the communications network to which this node is attached, if any”.

13868 **NAME**

13869 sys/wait.h — declarations for waiting

13870 **SYNOPSIS**

13871 #include &lt;sys/wait.h&gt;

13872 **DESCRIPTION**13873 The <sys/wait.h> header shall define the following symbolic constants for use with *waitpid()*:13874 XSI **WCONTINUED** Report status of continued child process.13875 **WNOHANG** Do not hang if no status is available; return immediately.13876 **WUNTRACED** Report status of stopped child process.

13877 The &lt;sys/wait.h&gt; header shall define the following macros for analysis of process status values:

13878 **WEXITSTATUS** Return exit status.13879 XSI **WIFCONTINUED** True if child has been continued.13880 **WIFEXITED** True if child exited normally.13881 **WIFSIGNALED** True if child exited due to uncaught signal.13882 **WIFSTOPPED** True if child is currently stopped.13883 **WSTOPSIG** Return signal number that caused process to stop.13884 **WTERMSIG** Return signal number that caused process to terminate.13885 The <sys/wait.h> header shall define the following symbolic constants as possible values for the  
13886 *options* argument to *waitid()*:13887 **WEXITED** Wait for processes that have exited.13888 **WNOWAIT** Keep the process whose status is returned in *infop* in a waitable state.13889 **WSTOPPED** Status is returned for any child that has stopped upon receipt of a signal.13890 XSI The **WCONTINUED** and **WNOHANG** constants, described above for *waitpid()*, can also be  
13891 used with *waitid()*.13892 The type **idtype\_t** shall be defined as an enumeration type whose possible values shall include  
13893 at least the following:13894 **P\_ALL**13895 **P\_PGID**13896 **P\_PID**13897 The <sys/wait.h> header shall define the **id\_t** and **pid\_t** types as described in <sys/types.h>.13898 The <sys/wait.h> header shall define the **siginfo\_t** type and the **sigval** union as described in |  
13899 <signal.h>.

13900 Inclusion of the &lt;sys/wait.h&gt; header may also make visible all symbols from &lt;signal.h&gt;.

13901 The following shall be declared as functions and may also be defined as macros. Function  
13902 prototypes shall be provided.13903 **pid\_t** wait(int \*);13904 **int** waitid(idtype\_t, id\_t, siginfo\_t \*, int);13905 **pid\_t** waitpid(pid\_t, int \*, int);

13906 **APPLICATION USAGE**

13907 None.

13908 **RATIONALE**

13909 None.

13910 **FUTURE DIRECTIONS**

13911 None.

13912 **SEE ALSO**

13913 [<signal.h>](#), [<sys/resource.h>](#), [<sys/types.h>](#)

13914 XSH *wait()*, *waitid()*

13915 **CHANGE HISTORY**

13916 First released in Issue 3.

13917 Included for alignment with the POSIX.1-1988 standard.

13918 **Issue 6**

13919 The *wait3()* function is removed.

13920 **Issue 7**

13921 The *waitid()* function and symbolic constants for its *options* argument are moved to the Base.

13922 The description of the WNOHANG constant is clarified.

13923 The requirement for [<sys/wait.h>](#) to define the **rusage** structure as described in [+<sys/resource.h>](#) is removed, and [<sys/wait.h>](#) is no longer allowed to make visible all symbols [+<sys/resource.h>](#).  
13924  
13925

13926 **NAME**

13927       syslog.h — definitions for system error logging

13928 **SYNOPSIS**

13929 XSI       #include &lt;syslog.h&gt;

13930 **DESCRIPTION**13931       The <syslog.h> header shall define the following symbolic constants, zero or more of which  
13932       may be OR'ed together to form the *logopt* option of *openlog()*:

13933       LOG\_PID               Log the process ID with each message.

13934       LOG\_CONS             Log to the system console on error.

13935       LOG\_NDELAY           Connect to syslog daemon immediately.

13936       LOG\_ODELAY           Delay open until *syslog()* is called.

13937       LOG\_NOWAIT           Do not wait for child processes.

13938       The <syslog.h> header shall define the following symbolic constants for use as the *facility*  
13939       argument to *openlog()*:

13940       LOG\_KERN             Reserved for message generated by the system.

13941       LOG\_USER             Message generated by a process.

13942       LOG\_MAIL             Reserved for message generated by mail system.

13943       LOG\_NEWS             Reserved for message generated by news system.

13944       LOG\_UUCP             Reserved for message generated by UUCP system.

13945       LOG\_DAEMON           Reserved for message generated by system daemon.

13946       LOG\_AUTH             Reserved for message generated by authorization daemon.

13947       LOG\_CRON             Reserved for message generated by clock daemon.

13948       LOG\_LPR              Reserved for message generated by printer system.

13949       LOG\_LOCAL0           Reserved for local use.

13950       LOG\_LOCAL1           Reserved for local use.

13951       LOG\_LOCAL2           Reserved for local use.

13952       LOG\_LOCAL3           Reserved for local use.

13953       LOG\_LOCAL4           Reserved for local use.

13954       LOG\_LOCAL5           Reserved for local use.

13955       LOG\_LOCAL6           Reserved for local use.

13956       LOG\_LOCAL7           Reserved for local use.

13957       The <syslog.h> header shall define the following macros for constructing the *maskpri* argument  
13958       to *setlogmask()*. The following macros expand to an expression of type **int** when the argument  
13959       *pri* is an expression of type **int**:13960       LOG\_MASK(*pri*)       A mask for priority *pri*.13961       The <syslog.h> header shall define the following symbolic constants for use as the *priority*  
13962       argument of *syslog()*:

13963	LOG_EMERG	A panic condition was reported to all processes.
13964	LOG_ALERT	A condition that should be corrected immediately.
13965	LOG_CRIT	A critical condition.
13966	LOG_ERR	An error message.
13967	LOG_WARNING	A warning message.
13968	LOG_NOTICE	A condition requiring special handling.
13969	LOG_INFO	A general information message.
13970	LOG_DEBUG	A message useful for debugging programs.
13971	The following shall be declared as functions and may also be defined as macros. Function	
13972	prototypes shall be provided.	
13973	void closelog(void);	
13974	void openlog(const char *, int, int);	
13975	int setlogmask(int);	
13976	void syslog(int, const char *, ...);	
13977	<b>APPLICATION USAGE</b>	
13978	None.	
13979	<b>RATIONALE</b>	
13980	None.	
13981	<b>FUTURE DIRECTIONS</b>	
13982	None.	
13983	<b>SEE ALSO</b>	
13984	XSH <i>closelog()</i>	
13985	<b>CHANGE HISTORY</b>	
13986	First released in Issue 4, Version 2.	
13987	<b>Issue 5</b>	
13988	Moved from X/Open UNIX to BASE.	
13989	<b>Issue 7</b>	
13990	This reference page is clarified with respect to macros and symbolic constants.	

13991 **NAME**

13992 tar.h — extended tar definitions

13993 **SYNOPSIS**

13994 #include &lt;tar.h&gt;

13995 **DESCRIPTION**

13996 The &lt;tar.h&gt; header shall define the following symbolic constants with the indicated values.

13997 General definitions:

Name	Value	Description
TMAGIC	"ustar"	Used in the magic field in the <b>ustar</b> header block, including the trailing null byte.
TMAGLEN	6	Length in octets of the magic field.
TVERSION	"00"	Used in the version field in the <b>ustar</b> header block, excluding the trailing null byte.
TVERSLEN	2	Length in octets of the version field.

14005 *Typeflag* field definitions:

Name	Value	Description
REGTYPE	'0'	Regular file.
AREGTYPE	'\0'	Regular file.
LNKTYPE	'1'	Link.
SYMTYPE	'2'	Symbolic link.
CHRTYPE	'3'	Character special.
BLKTYPE	'4'	Block special.
DIRTYPE	'5'	Directory.
FIFOTYPE	'6'	FIFO special.
CONTTYPE	'7'	Reserved.

14016 *Mode* field bit definitions (octal):

Name	Value	Description
TSUID	04000	Set UID on execution.
TSGID	02000	Set GID on execution.
TSVTX	01000	On directories, restricted deletion flag.
TUREAD	00400	Read by owner.
TUWRITE	00200	Write by owner special.
TUEXEC	00100	Execute/search by owner.
TGREAD	00040	Read by group.
TGWRITE	00020	Write by group.
TGEXEC	00010	Execute/search by group.
TOREAD	00004	Read by other.
TOWRITE	00002	Write by other.
TOEXEC	00001	Execute/search by other.



14030 **APPLICATION USAGE**

14031 None.

14032 **RATIONALE**

14033 None.

14034 **FUTURE DIRECTIONS**

14035 None.

14036 **SEE ALSO**

14037 XCU *pax*

14038 **CHANGE HISTORY**

14039 First released in Issue 3. Derived from the POSIX.1-1988 standard.

14040 **Issue 6**

14041 The SEE ALSO section is updated to refer to *pax*.

14042 **Issue 7**

14043 This reference page is clarified with respect to macros and symbolic constants.

14044 **NAME**

14045       termios.h — define values for termios

14046 **SYNOPSIS**

14047       #include &lt;termios.h&gt;

14048 **DESCRIPTION**

14049       The <termios.h> header shall contain the definitions used by the terminal I/O interfaces (see  
 14050       Chapter 11 (on page 199) for the structures and names defined).

14051 **The termios Structure**14052       The <termios.h> header shall define the following data types through **typedef**:14053       **cc\_t**               Used for terminal special characters.14054       **speed\_t**          Used for terminal baud rates.14055       **tcflag\_t**         Used for terminal modes.

14056       The above types shall be all unsigned integer types.

14057       The implementation shall support one or more programming environments in which the widths  
 14058       of **cc\_t**, **speed\_t**, and **tcflag\_t** are no greater than the width of type **long**. The names of these  
 14059       programming environments can be obtained using the *confstr()* function or the *getconf* utility.

14060       The <termios.h> header shall define the **termios** structure, which shall include at least the  
 14061       following members:

14062       **tcflag\_t**   **c\_iflag**       Input modes.  
 14063       **tcflag\_t**   **c\_oflag**       Output modes.  
 14064       **tcflag\_t**   **c\_cflag**       Control modes.  
 14065       **tcflag\_t**   **c\_lflag**       Local modes.  
 14066       **cc\_t**        **c\_cc[NCCS]**   Control characters.

14067       The &lt;termios.h&gt; header shall define the following symbolic constant:

14068       **NCCS**           Size of the array **c\_cc** for control characters.

14069       The <termios.h> header shall define the following symbolic constants for use as subscripts for  
 14070       the array **c\_cc**:

Subscript Usage		Description
Canonical Mode	Non-Canonical Mode	
VEOF	VINTR	EOF character.
VEOL		EOL character.
VERASE		ERASE character.
VINTR		INTR character.
VKILL		KILL character.
	VMIN	MIN value.
VQUIT	VQUIT	QUIT character.
VSTART	VSTART	START character.
VSTOP	VSTOP	STOP character.
VSUSP	VSUSP	SUSP character.
	VTIME	TIME value.

14084       The subscript values shall be suitable for use in **#if** preprocessing directives and shall be distinct,  
 14085       except that the VMIN and VTIME subscripts may have the same values as the VEOF and VEOL  
 14086       subscripts, respectively.

14087 **Input Modes**

14088 The <termios.h> header shall define the following symbolic constants for use as flags in the  
14089 *c\_iflag* field. The *c\_iflag* field describes the basic terminal input control.

14090	BRKINT	Signal interrupt on break.
14091	ICRNL	Map CR to NL on input.
14092	IGNBRK	Ignore break condition.
14093	IGNCR	Ignore CR.
14094	IGNPAR	Ignore characters with parity errors.
14095	INLCR	Map NL to CR on input.
14096	INPCK	Enable input parity check.
14097	ISTRIP	Strip character.
14098	IXANY	Enable any character to restart output.
14099	IXOFF	Enable start/stop input control.
14100	IXON	Enable start/stop output control.
14101	PARMRK	Mark parity errors.

14102 **Output Modes**

14103 The <termios.h> header shall define the following symbolic constants for use as flags in the  
14104 *c\_oflag* field. The *c\_oflag* field specifies the system treatment of output.

14105	OPOST	Post-process output.
14106	XSI ONLCR	Map NL to CR-NL on output.
14107	XSI OCRNL	Map CR to NL on output.
14108	XSI ONOCR	No CR output at column 0.
14109	XSI ONLRET	NL performs CR function.
14110	XSI OFDEL	Fill is DEL.
14111	XSI OFILL	Use fill characters for delay.
14112	XSI NLDLY	Select newline delays:
14113		NL0 Newline type 0.
14114		NL1 Newline type 1.
14115	XSI CRDLY	Select carriage-return delays:
14116		CR0 Carriage-return delay type 0.
14117		CR1 Carriage-return delay type 1.
14118		CR2 Carriage-return delay type 2.
14119		CR3 Carriage-return delay type 3.

14120	XSI	<b>TABDLY</b>	Select horizontal-tab delays:
14121		TAB0	Horizontal-tab delay type 0.
14122		TAB1	Horizontal-tab delay type 1.
14123		TAB2	Horizontal-tab delay type 2.
14124		TAB3	Expand tabs to spaces.
14125	XSI	<b>BSDLY</b>	Select backspace delays:
14126		BS0	Backspace-delay type 0.
14127		BS1	Backspace-delay type 1.
14128	XSI	<b>VTDLY</b>	Select vertical-tab delays:
14129		VT0	Vertical-tab delay type 0.
14130		VT1	Vertical-tab delay type 1.
14131	XSI	<b>FFDLY</b>	Select form-feed delays:
14132		FF0	Form-feed delay type 0.
14133		FF1	Form-feed delay type 1.

#### 14134 **Baud Rate Selection**

14135 The <termios.h> header shall define the following symbolic constants for use as values of  
 14136 objects of type **speed\_t**.

14137 The input and output baud rates are stored in the **termios** structure. These are the valid values  
 14138 for objects of type **speed\_t**. Not all baud rates need be supported by the underlying hardware.

14139	B0	Hang up
14140	B50	50 baud
14141	B75	75 baud
14142	B110	110 baud
14143	B134	134.5 baud
14144	B150	150 baud
14145	B200	200 baud
14146	B300	300 baud
14147	B600	600 baud
14148	B1200	1 200 baud
14149	B1800	1 800 baud
14150	B2400	2 400 baud

14151	B4800	4 800 baud
14152	B9600	9 600 baud
14153	B19200	19 200 baud
14154	B38400	38 400 baud

#### 14155 Control Modes

14156 The <termios.h> header shall define the following symbolic constants for use as flags in the  
 14157 *c\_cflag* field. The *c\_cflag* field describes the hardware control of the terminal; not all values  
 14158 specified are required to be supported by the underlying hardware.

14159	CSIZE	Character size:
14160		CS5 5 bits
14161		CS6 6 bits
14162		CS7 7 bits
14163		CS8 8 bits
14164	CSTOPB	Send two stop bits, else one.
14165	CREAD	Enable receiver.
14166	PARENB	Parity enable.
14167	PARODD	Odd parity, else even.
14168	HUPCL	Hang up on last close.
14169	CLOCAL	Ignore modem status lines.

14170 The implementation shall support the functionality associated with the symbols CS7, CS8,  
 14171 CSTOPB, PARODD, and PARENB.

#### 14172 Local Modes

14173 The <termios.h> header shall define the following symbolic constants for use as flags in the  
 14174 *c\_lflag* field. The *c\_lflag* field of the argument structure is used to control various terminal  
 14175 functions.

14176	ECHO	Enable echo.
14177	ECHOE	Echo erase character as error-correcting backspace.
14178	ECHOK	Echo KILL.
14179	ECHONL	Echo NL.
14180	ICANON	Canonical input (erase and kill processing).
14181	IEXTEN	Enable extended input character processing.
14182	ISIG	Enable signals.
14183	NOFLSH	Disable flush after interrupt or quit.
14184	TOSTOP	Send SIGTTOU for background output.

**Attribute Selection**

The <termios.h> header shall define the following symbolic constants for use with *tcsetattr()*:

TCSANOW      Change attributes immediately.  
 TCSADRAIN    Change attributes when output has drained.  
 TCSAFLUSH    Change attributes when output has drained; also flush pending input.

**Line Control**

The <termios.h> header shall define the following symbolic constants for use with *tcflush()*:

TCIFLUSH      Flush pending input.  
 TCIOFLUSH    Flush both pending input and untransmitted output.  
 TCOFLUSH     Flush untransmitted output.

The <termios.h> header shall define the following symbolic constants for use with *tcflow()*:

TCIOFF        Transmit a STOP character, intended to suspend input data.  
 TCION         Transmit a START character, intended to restart input data.  
 TCOOFF        Suspend output.  
 TCOON         Restart output.

The <termios.h> header shall define the **pid\_t** type as described in <sys/types.h>.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
speed_t cfgetispeed(const struct termios *);
speed_t cfgetospeed(const struct termios *);
int      cfsetispeed(struct termios *, speed_t);
int      cfsetospeed(struct termios *, speed_t);
int      tcdrain(int);
int      tcflow(int, int);
int      tcflush(int, int);
int      tcgetattr(int, struct termios *);
pid_t    tcgetsid(int);
int      tcsendbreak(int, int);
int      tcsetattr(int, int, const struct termios *);
```

**APPLICATION USAGE**

The following names are reserved for XSI-conformant systems to use as an extension to the above; therefore strictly conforming applications shall not use them:

CBAUD	EXTB	VDSUSP
DEFECHO	FLUSHO	VLNEXT
ECHOCTL	LOBLK	VREPRINT
ECHOKE	PENDIN	VSTATUS
ECHOPRT	SWTCH	VWERASE
EXTA	VDISCARD	

14223 **RATIONALE**

14224 None.

14225 **FUTURE DIRECTIONS**

14226 None.

14227 **SEE ALSO**

14228 <sys/types.h>

14229 XSH *cfgetispeed()*, *cfgetospeed()*, *cfsetispeed()*, *cfsetospeed()*, *confstr()*, *tcdrain()*, *tcflow()*, *tcflush()*,  
14230 *tcgetattr()*, *tcgetsid()*, *tcsendbreak()*, *tcsetattr()*

14231 XCU Chapter 11 (on page 199), *getconf*

14232 **CHANGE HISTORY**

14233 First released in Issue 3.

14234 Included for alignment with the ISO POSIX-1 standard.

14235 **Issue 6**

14236 The LEGACY symbols IUCLC, OLCUC, and XCASE are removed.

14237 FIPS 151-2 requirements for the symbols CS7, CS8, CSTOPB, PARODD, and PARENB are  
14238 reaffirmed.

14239 IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/19 is applied, changing ECHOK to  
14240 ECHOK in the APPLICATION USAGE section.

14241 **Issue 7**

14242 Austin Group Interpretation 1003.1-2001 #144 is applied, moving functionality relating to the  
14243 IXANY symbol from the XSI option to the Base.

14244 SD5-XBD-ERN-35 is applied, adding the OFDEL output mode.

14245 This reference page is clarified with respect to macros and symbolic constants, and a declaration  
14246 for the **pid\_t** type is added.

14247 **NAME**14248 `tgmath.h` — type-generic macros14249 **SYNOPSIS**14250 `#include <tgmath.h>`14251 **DESCRIPTION**

14252 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 14253 conflict between the requirements described here and the ISO C standard is unintentional. This  
 14254 volume of POSIX.1-2008 defers to the ISO C standard.

14255 The `<tgmath.h>` header shall include the headers `<math.h>` and `<complex.h>` and shall define  
 14256 several type-generic macros.

14257 Of the functions contained within the `<math.h>` and `<complex.h>` headers without an *f* (**float**)  
 14258 or *l* (**long double**) suffix, several have one or more parameters whose corresponding real type is  
 14259 XSI **double**. For each such function, except *modf()*, *j0()*, *j1()*, *jn()*, *y0()*, *y1()*, and *yn()*, there shall  
 14260 be a corresponding type-generic macro. The parameters whose corresponding real type is  
 14261 **double** in the function synopsis are generic parameters. Use of the macro invokes a function  
 14262 whose corresponding real type and type domain are determined by the arguments for the  
 14263 generic parameters.

14264 Use of the macro invokes a function whose generic parameters have the corresponding real type  
 14265 determined as follows:

- 14266 • First, if any argument for generic parameters has type **long double**, the type determined is  
 14267 **long double**.
- 14268 • Otherwise, if any argument for generic parameters has type **double** or is of integer type,  
 14269 the type determined is **double**.
- 14270 • Otherwise, the type determined is **float**.

14271 For each unsuffixed function in the `<math.h>` header for which there is a function in the  
 14272 `<complex.h>` header with the same name except for a *c* prefix, the corresponding type-generic  
 14273 macro (for both functions) has the same name as the function in the `<math.h>` header. The  
 14274 corresponding type-generic macro for *fabs()* and *cabs()* is *fabs()*.

14275	<math.h> Function	<complex.h> Function	Type-Generic Macro
14276	<i>acos()</i>	<i>cacos()</i>	<i>acos()</i>
14277	<i>asin()</i>	<i>casin()</i>	<i>asin()</i>
14278	<i>atan()</i>	<i>catan()</i>	<i>atan()</i>
14279	<i>acosh()</i>	<i>cacosh()</i>	<i>acosh()</i>
14280	<i>asinh()</i>	<i>casinh()</i>	<i>asinh()</i>
14281	<i>atanh()</i>	<i>catanh()</i>	<i>atanh()</i>
14282	<i>cos()</i>	<i>ccos()</i>	<i>cos()</i>
14283	<i>sin()</i>	<i>csin()</i>	<i>sin()</i>
14284	<i>tan()</i>	<i>ctan()</i>	<i>tan()</i>
14285	<i>cosh()</i>	<i>ccosh()</i>	<i>cosh()</i>
14286	<i>sinh()</i>	<i>csinh()</i>	<i>sinh()</i>
14287	<i>tanh()</i>	<i>ctanh()</i>	<i>tanh()</i>
14288	<i>exp()</i>	<i>cexp()</i>	<i>exp()</i>
14289	<i>log()</i>	<i>clog()</i>	<i>log()</i>
14290	<i>pow()</i>	<i>cpow()</i>	<i>pow()</i>
14291	<i>sqrt()</i>	<i>csqrt()</i>	<i>sqrt()</i>
14292	<i>fabs()</i>	<i>cabs()</i>	<i>fabs()</i>



If at least one argument for a generic parameter is complex, then use of the macro invokes a complex function; otherwise, use of the macro invokes a real function.

For each unsuffixed function in the **<math.h>** header without a *c*-prefixed counterpart in the **<complex.h>** header, except for *modf()*, *j0()*, *j1()*, *jn()*, *y0()*, *y1()*, and *yn()*, the corresponding type-generic macro has the same name as the function. These type-generic macros are:

<i>atan2()</i>	<i>fma()</i>	<i>llround()</i>	<i>remainder()</i>
<i>cbrt()</i>	<i>fmax()</i>	<i>log10()</i>	<i>remquo()</i>
<i>ceil()</i>	<i>fmin()</i>	<i>log1p()</i>	<i>rint()</i>
<i>copysign()</i>	<i>fmod()</i>	<i>log2()</i>	<i>round()</i>
<i>erf()</i>	<i>frexp()</i>	<i>logb()</i>	<i>scalbln()</i>
<i>erfc()</i>	<i>hypot()</i>	<i>lrint()</i>	<i>scalbn()</i>
<i>exp2()</i>	<i>ilogb()</i>	<i>lround()</i>	<i>tgamma()</i>
<i>expm1()</i>	<i>ldexp()</i>	<i>nearbyint()</i>	<i>trunc()</i>
<i>fdim()</i>	<i>lgamma()</i>	<i>nextafter()</i>	
<i>floor()</i>	<i>llrint()</i>	<i>nexttoward()</i>	

If all arguments for generic parameters are real, then use of the macro invokes a real function; otherwise, use of the macro results in undefined behavior.

For each unsuffixed function in the **<complex.h>** header that is not a *c*-prefixed counterpart to a function in the **<math.h>** header, the corresponding type-generic macro has the same name as the function. These type-generic macros are:

*carg()*  
*cimag()*  
*conj()*  
*cproj()*  
*creal()*

Use of the macro with any real or complex argument invokes a complex function.

## APPLICATION USAGE

With the declarations:

```
#include <tgmath.h>
int n;
float f;
double d;
long double ld;
float complex fc;
double complex dc;
long double complex ldc;
```

functions invoked by use of type-generic macros are shown in the following table:

Macro	Use Invokes
<i>exp(n)</i>	<i>exp(n)</i> , the function
<i>acosh(f)</i>	<i>acoshf(f)</i>
<i>sin(d)</i>	<i>sin(d)</i> , the function
<i>atan(ld)</i>	<i>atanl(ld)</i>
<i>log(fc)</i>	<i>clogf(fc)</i>
<i>sqrt(dc)</i>	<i>csqrt(dc)</i>

Macro	Use Invokes
<i>pow(ldc,f)</i>	<i>cpowl(ldc,f)</i>
<i>remainder(n,n)</i>	<i>remainder(n,n)</i> , the function
<i>nextafter(d,f)</i>	<i>nextafter(d,f)</i> , the function
<i>nexttoward(f,ld)</i>	<i>nexttowardf(f,ld)</i>
<i>copysign(n,ld)</i>	<i>copysignl(n,ld)</i>
<i>ceil(fc)</i>	Undefined behavior
<i>rint(dc)</i>	Undefined behavior
<i>fmax(ldc,ld)</i>	Undefined behavior
<i>carg(n)</i>	<i>carg(n)</i> , the function
<i>cproj(f)</i>	<i>cprojf(f)</i>
<i>creal(d)</i>	<i>creal(d)</i> , the function
<i>cimag(ld)</i>	<i>cimagl(ld)</i>
<i>cabs(fc)</i>	<i>cabsf(fc)</i>
<i>carg(dc)</i>	<i>carg(dc)</i> , the function
<i>cproj(ldc)</i>	<i>cprojl(ldc)</i>

**RATIONALE**

Type-generic macros allow calling a function whose type is determined by the argument type, as is the case for C operators such as '+' and '\*'. For example, with a type-generic *cos()* macro, the expression *cos((float)x)* will have type **float**. This feature enables writing more portably efficient code and alleviates need for awkward casting and suffixing in the process of porting or adjusting precision. Generic math functions are a widely appreciated feature of Fortran.

The only arguments that affect the type resolution are the arguments corresponding to the parameters that have type **double** in the synopsis. Hence the type of a type-generic call to *nexttoward()*, whose second parameter is **long double** in the synopsis, is determined solely by the type of the first argument.

The term “type-generic” was chosen over the proposed alternatives of intrinsic and overloading. The term is more specific than intrinsic, which already is widely used with a more general meaning, and reflects a closer match to Fortran’s generic functions than to C++ overloading.

The macros are placed in their own header in order not to silently break old programs that include the <math.h> header; for example, with:

```
printf ("%e", sin(x))
```

*modf(double, double \*)* is excluded because no way was seen to make it safe without complicating the type resolution.

The implementation might, as an extension, endow appropriate ones of the macros that POSIX.1-2008 specifies only for real arguments with the ability to invoke the complex functions.

POSIX.1-2008 does not prescribe any particular implementation mechanism for generic macros. It could be implemented simply with built-in macros. The generic macro for *sqrt()*, for example, could be implemented with:

```
#undef sqrt
#define sqrt(x) __BUILTIN_GENERIC_sqrt(x)
```

Generic macros are designed for a useful level of consistency with C++ overloaded math functions.

The great majority of existing C programs are expected to be unaffected when the <tgmath.h> header is included instead of the <math.h> or <complex.h> headers. Generic macros are similar to the ISO/IEC 9899:1999 standard library masking macros, though the semantic types of return

14383 values differ.

14384 The ability to overload on integer as well as floating types would have been useful for some  
14385 functions; for example, *copysign()*. Overloading with different numbers of arguments would  
14386 have allowed reusing names; for example, *remainder()* for *remquo()*. However, these facilities  
14387 would have complicated the specification; and their natural consistent use, such as for a floating  
14388 *abs()* or a two-argument *atan()*, would have introduced further inconsistencies with the  
14389 ISO/IEC 9899:1999 standard for insufficient benefit.

14390 The ISO C standard in no way limits the implementation's options for efficiency, including  
14391 inlining library functions.

#### 14392 FUTURE DIRECTIONS

14393 None.

#### 14394 SEE ALSO

14395 <math.h>, <complex.h>

14396 XSH *cabs()*, *fabs()*, *modf()*

#### 14397 CHANGE HISTORY

14398 First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

#### 14399 Issue 7

14400 Austin Group Interpretation 1003.1-2001 #184 is applied, clarifying the functions for which a  
14401 corresponding type-generic macro exists with the same name as the function.

14402 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0075 [357,427] is applied.

## 14403 NAME

14404 time.h — time types

## 14405 SYNOPSIS

14406 #include &lt;time.h&gt;

## 14407 DESCRIPTION

14408 CX Some of the functionality described on this reference page extends the ISO C standard.  
 14409 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to  
 14410 enable the visibility of these symbols in this header.

14411 The <time.h> header shall define the **clock\_t**, **size\_t**, **time\_t**, types as described in  
 14412 <sys/types.h>.

14413 CX The <time.h> header shall define the **clockid\_t** and **timer\_t** types as described in <sys/types.h>.

14414 The <time.h> header shall define the **locale\_t** type as described in <locale.h>.

14415 CPT The <time.h> header shall define the **pid\_t** type as described in <sys/types.h>.

14416 CX The tag **sigevent** shall be declared as naming an incomplete structure type, the contents of which  
 14417 are described in the <signal.h> header.

14418 The <time.h> header shall declare the **tm** structure, which shall include at least the following  
 14419 members:

14420	int	tm_sec	Seconds [0,60].
14421	int	tm_min	Minutes [0,59].
14422	int	tm_hour	Hour [0,23].
14423	int	tm_mday	Day of month [1,31].
14424	int	tm_mon	Month of year [0,11].
14425	int	tm_year	Years since 1900.
14426	int	tm_wday	Day of week [0,6] (Sunday =0).
14427	int	tm_yday	Day of year [0,365].
14428	int	tm_isdst	Daylight Savings flag.

14429 The value of *tm\_isdst* shall be positive if Daylight Savings Time is in effect, 0 if Daylight Savings  
 14430 Time is not in effect, and negative if the information is not available.

14431 CX The <time.h> header shall declare the **timespec** structure, which shall include at least the  
 14432 following members:

14433	time_t	tv_sec	Seconds.
14434	long	tv_nsec	Nanoseconds.

14435 The <time.h> header shall also declare the **itimerspec** structure, which shall include at least the  
 14436 following members:

14437	struct timespec	it_interval	Timer period.
14438	struct timespec	it_value	Timer expiration.

14439 The <time.h> header shall define the following macros:

14440 NULL As described in <stddef.h>.

14441 CLOCKS\_PER\_SEC A number used to convert the value returned by the *clock()* function into  
 14442 XSI seconds. The value shall be an expression with type **clock\_t**. The value of  
 14443 CLOCKS\_PER\_SEC shall be 1 million on XSI-conformant systems.  
 14444 However, it may be variable on other systems, and it should not be

14445		assumed that <code>CLOCKS_PER_SEC</code> is a compile-time constant.
14446	CX	The <time.h> header shall define the following symbolic constants. The values shall have a type
14447		that is assignment-compatible with <code>clockid_t</code> .
14448	MON	<code>CLOCK_MONOTONIC</code>
14449		The identifier for the system-wide monotonic clock, which is defined as a
14450		clock measuring real time, whose value cannot be set via <code>clock_settime()</code>
14451		and which cannot have negative clock jumps. The maximum possible
14452		clock jump shall be implementation-defined.
14453	CPT	<code>CLOCK_PROCESS_CPUTIME_ID</code>
14454		The identifier of the CPU-time clock associated with the process making a
14455		<code>clock()</code> or <code>timer*()</code> function call.
14456	CX	<code>CLOCK_REALTIME</code> The identifier of the system-wide clock measuring real time.
14457	TCT	<code>CLOCK_THREAD_CPUTIME_ID</code>
14458		The identifier of the CPU-time clock associated with the thread making a
14459		<code>clock()</code> or <code>timer*()</code> function call.
14460	CX	The <time.h> header shall define the following symbolic constant:
14461		<code>TIMER_ABSTIME</code> Flag indicating time is absolute. For functions taking timer objects, this
14462		refers to the clock associated with the timer.
14463	XSI	The <time.h> header shall provide a declaration or definition for <code>getdate_err</code> . The <code>getdate_err</code>
14464		symbol shall expand to an expression of type <code>int</code> . It is unspecified whether <code>getdate_err</code> is a macro
14465		or an identifier declared with external linkage, and whether or not it is a modifiable lvalue. If a
14466		macro definition is suppressed in order to access an actual object, or a program defines an
14467		identifier with the name <code>getdate_err</code> , the behavior is undefined.
14468		The following shall be declared as functions and may also be defined as macros. Function
14469		prototypes shall be provided.
14470	OB	<code>char *asctime(const struct tm *);</code>
14471	OB CX	<code>char *asctime_r(const struct tm *restrict, char *restrict);</code>
14472		<code>clock_t clock(void);</code>
14473	CPT	<code>int clock_getcpuclockid(pid_t, clockid_t *);</code>
14474	CX	<code>int clock_getres(clockid_t, struct timespec *);</code>
14475		<code>int clock_gettime(clockid_t, struct timespec *);</code>
14476		<code>int clock_nanosleep(clockid_t, int, const struct timespec *,</code>
14477		<code>struct timespec *);</code>
14478		<code>int clock_settime(clockid_t, const struct timespec *);</code>
14479	OB	<code>char *ctime(const time_t *);</code>
14480	OB CX	<code>char *ctime_r(const time_t *, char *);</code>
14481		<code>double difftime(time_t, time_t);</code>
14482	XSI	<code>struct tm *getdate(const char *);</code>
14483		<code>struct tm *gmtime(const time_t *);</code>
14484	CX	<code>struct tm *gmtime_r(const time_t *restrict, struct tm *restrict);</code>
14485		<code>struct tm *localtime(const time_t *);</code>
14486	CX	<code>struct tm *localtime_r(const time_t *restrict, struct tm *restrict);</code>
14487		<code>time_t mktime(struct tm *);</code>
14488	CX	<code>int nanosleep(const struct timespec *, struct timespec *);</code>
14489		<code>size_t strftime(char *restrict, size_t, const char *restrict,</code>
14490		<code>const struct tm *restrict);</code>

```

14491 CX      size_t      strftime_l(char *restrict, size_t, const char *restrict,
14492              const struct tm *restrict, locale_t);
14493 XSI      char        *strptime(const char *restrict, const char *restrict,
14494              struct tm *restrict);
14495          time_t      time(time_t *);
14496 CX      int          timer_create(clockid_t, struct sigevent *restrict,
14497              timer_t *restrict);
14498          int          timer_delete(timer_t);
14499          int          timer_getoverrun(timer_t);
14500          int          timer_gettime(timer_t, struct itimerspec *);
14501          int          timer_settime(timer_t, int, const struct itimerspec *restrict,
14502              struct itimerspec *restrict);
14503          void         tzset(void);

```

14504 The <time.h> header shall declare the following as variables:

```

14505 XSI      extern int     daylight;
14506          extern long    timezone;
14507 CX      extern char     *tzname[];

```

14508 CX Inclusion of the <time.h> header may make visible all symbols from the <signal.h> header.

## 14509 APPLICATION USAGE

14510 The range [0,60] for *tm\_sec* allows for the occasional leap second.

14511 *tm\_year* is a signed value; therefore, years before 1900 may be represented.

14512 To obtain the number of clock ticks per second returned by the *times()* function, applications  
14513 should call *sysconf(\_SC\_CLK\_TCK)*.

## 14514 RATIONALE

14515 The range [0,60] seconds allows for positive or negative leap seconds. The formal definition of  
14516 UTC does not permit double leap seconds, so all mention of double leap seconds has been  
14517 removed, and the range shortened from the former [0,61] seconds seen in earlier versions of this  
14518 standard.

## 14519 FUTURE DIRECTIONS

14520 None.

## 14521 SEE ALSO

14522 <locale.h>, <signal.h>, <stddef.h>, <sys/types.h>

14523 XSH Section 2.2 (on page 472), *asctime()*, *clock()*, *clock\_getcpuclockid()*, *clock\_getres()*,  
14524 *clock\_nanosleep()*, *ctime()*, *difftime()*, *getdate()*, *gmtime()*, *localtime()*, *mktime()*, *mq\_receive()*,  
14525 *mq\_send()*, *nanosleep()*, *pthread\_getcpuclockid()*, *pthread\_mutex\_timedlock()*,  
14526 *pthread\_rwlock\_timedrdlock()*, *pthread\_rwlock\_timedwrlock()*, *sem\_timedwait()*, *strptime()*, *strptime()*,  
14527 *sysconf()*, *time()*, *timer\_create()*, *timer\_delete()*, *timer\_getoverrun()*, *tzset()*, *utime()*

## 14528 CHANGE HISTORY

14529 First released in Issue 1. Derived from Issue 1 of the SVID.

## 14530 Issue 5

14531 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX  
14532 Threads Extension.

## Issue 6

The Open Group Corrigendum U035/6 is applied. In the DESCRIPTION, the types **clockid\_t** and **timer\_t** have been described.

The following changes are made for alignment with the ISO POSIX-1: 1996 standard:

- The POSIX timer-related functions are marked as part of the Timers option.

The symbolic name CLK\_TCK is removed. Application usage is added describing how its equivalent functionality can be obtained using *sysconf()*.

The *clock\_getcpuclockid()* function and manifest constants CLOCK\_PROCESS\_CPUTIME\_ID and CLOCK\_THREAD\_CPUTIME\_ID are added for alignment with IEEE Std 1003.1d-1999.

The manifest constant CLOCK\_MONOTONIC and the *clock\_nanosleep()* function are added for alignment with IEEE Std 1003.1j-2000.

The following changes are made for alignment with the ISO/IEC 9899: 1999 standard:

- The range for seconds is changed from [0,61] to [0,60].
- The **restrict** keyword is added to the prototypes for *asctime\_r()*, *gmtime\_r()*, *localtime\_r()*, *strftime()*, *strptime()*, *timer\_create()*, and *timer\_settime()*.

IEEE PASC Interpretation 1003.1 #84 is applied adding the statement that symbols from the <signal.h> header may be made visible when the <time.h> header is included.

Extensions beyond the ISO C standard are marked.

## Issue 7

Austin Group Interpretation 1003.1-2001 #111 is applied.

SD5-XBD-ERN-74 is applied.

The *strptime\_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

Functionality relating to the Timers option is moved to the Base.

This reference page is clarified with respect to macros and symbolic constants, and declarations for the **locale\_t** and **pid\_t** types and the **sigevent** structure are added.

The description of the *getdate\_err* value is expanded.

POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0076 [212] and XBD/TC1-2008/0077 [212] are applied.



14562 **NAME**

14563       trace.h — tracing

14564 **SYNOPSIS**

14565 OB TRC   #include &lt;trace.h&gt;

14566 **DESCRIPTION**14567       The <trace.h> header shall define the **posix\_trace\_event\_info** structure, which shall include at  
14568       least the following members:

```

14569     trace_event_id_t   posix_event_id
14570     pid_t              posix_pid
14571     void               *posix_prog_address
14572     pthread_t          posix_thread_id
14573     struct timespec    posix_timestamp
14574     int                posix_truncation_status

```

14575       The <trace.h> header shall define the **posix\_trace\_status\_info** structure, which shall include at  
14576       least the following members:

```

14577     int                posix_stream_full_status
14578     int                posix_stream_overrun_status
14579     int                posix_stream_status
14580 OB TRL   int          posix_log_full_status
14581         int          posix_log_overrun_status
14582         int          posix_stream_flush_error
14583         int          posix_stream_flush_status

```

14584       The &lt;trace.h&gt; header shall define the following symbolic constants:

```

14585     POSIX_TRACE_ALL_EVENTS
14586 OB TRL   POSIX_TRACE_APPEND
14587 OB TRI   POSIX_TRACE_CLOSE_FOR_CHILD
14588 OB TEF   POSIX_TRACE_FILTER
14589 OB TRL   POSIX_TRACE_FLUSH
14590         POSIX_TRACE_FLUSH_START
14591         POSIX_TRACE_FLUSH_STOP
14592         POSIX_TRACE_FLUSHING
14593         POSIX_TRACE_FULL
14594         POSIX_TRACE_LOOP
14595         POSIX_TRACE_NO_OVERRUN
14596 OB TRL   POSIX_TRACE_NOT_FLUSHING
14597         POSIX_TRACE_NOT_FULL
14598 OB TRI   POSIX_TRACE_INHERITED
14599         POSIX_TRACE_NOT_TRUNCATED
14600         POSIX_TRACE_OVERFLOW
14601         POSIX_TRACE_OVERRUN
14602         POSIX_TRACE_RESUME
14603         POSIX_TRACE_RUNNING
14604         POSIX_TRACE_START
14605         POSIX_TRACE_STOP
14606         POSIX_TRACE_SUSPENDED
14607         POSIX_TRACE_SYSTEM_EVENTS
14608         POSIX_TRACE_TRUNCATED_READ

```



14609 POSIX\_TRACE\_TRUNCATED\_RECORD  
 14610 POSIX\_TRACE\_UNNAMED\_USER\_EVENT  
 14611 POSIX\_TRACE\_UNTIL\_FULL  
 14612 POSIX\_TRACE\_WOPID\_EVENTS

14613 OB TEF The <trace.h> header shall define the **size\_t**, **trace\_attr\_t**, **trace\_event\_id\_t**, **trace\_event\_set\_t**,  
 14614 and **trace\_id\_t** types as described in <sys/types.h>.

14615 The following shall be declared as functions and may also be defined as macros. Function  
 14616 prototypes shall be provided.

```

14617 int  posix_trace_attr_destroy(trace_attr_t *);
14618 int  posix_trace_attr_getclockres(const trace_attr_t *,
14619                                struct timespec *);
14620 int  posix_trace_attr_getcreatetime(const trace_attr_t *,
14621                                struct timespec *);
14622 int  posix_trace_attr_getgenversion(const trace_attr_t *, char *);
14623 TRI  int  posix_trace_attr_getinherited(const trace_attr_t *restrict,
14624                                int *restrict);
14625 TRL  int  posix_trace_attr_getlogfullpolicy(const trace_attr_t *restrict,
14626                                int *restrict);
14627 int  posix_trace_attr_getlogsize(const trace_attr_t *restrict,
14628                                size_t *restrict);
14629 int  posix_trace_attr_getmaxdatasize(const trace_attr_t *restrict,
14630                                size_t *restrict);
14631 int  posix_trace_attr_getmaxsystemeventsize(const trace_attr_t *restrict,
14632                                size_t *restrict);
14633 int  posix_trace_attr_getmaxusereventsize(const trace_attr_t *restrict,
14634                                size_t, size_t *restrict);
14635 int  posix_trace_attr_getname(const trace_attr_t *, char *);
14636 int  posix_trace_attr_getstreamfullpolicy(const trace_attr_t *restrict,
14637                                int *restrict);
14638 int  posix_trace_attr_getstreamsize(const trace_attr_t *restrict,
14639                                size_t *restrict);
14640 int  posix_trace_attr_init(trace_attr_t *);
14641 TRI  int  posix_trace_attr_setinherited(trace_attr_t *, int);
14642 TRL  int  posix_trace_attr_setlogfullpolicy(trace_attr_t *, int);
14643 int  posix_trace_attr_setlogsize(trace_attr_t *, size_t);
14644 int  posix_trace_attr_setmaxdatasize(trace_attr_t *, size_t);
14645 int  posix_trace_attr_setname(trace_attr_t *, const char *);
14646 int  posix_trace_attr_setstreamfullpolicy(trace_attr_t *, int);
14647 int  posix_trace_attr_setstreamsize(trace_attr_t *, size_t);
14648 int  posix_trace_clear(trace_id_t);
14649 TRL  int  posix_trace_close(trace_id_t);
14650 int  posix_trace_create(pid_t, const trace_attr_t *restrict,
14651                                trace_id_t *restrict);
14652 TRL  int  posix_trace_create_withlog(pid_t, const trace_attr_t *restrict,
14653                                int, trace_id_t *restrict);
14654 void posix_trace_event(trace_event_id_t, const void *restrict, size_t);
14655 int  posix_trace_eventid_equal(trace_id_t, trace_event_id_t,
14656                                trace_event_id_t);
14657 int  posix_trace_eventid_get_name(trace_id_t, trace_event_id_t, char *);
14658 int  posix_trace_eventid_open(const char *restrict,
```

```

14659         trace_event_id_t *restrict);
14660 TEF    int  posix_trace_eventset_add(trace_event_id_t, trace_event_set_t *);
14661       int  posix_trace_eventset_del(trace_event_id_t, trace_event_set_t *);
14662       int  posix_trace_eventset_empty(trace_event_set_t *);
14663       int  posix_trace_eventset_fill(trace_event_set_t *, int);
14664       int  posix_trace_eventset_ismember(trace_event_id_t,
14665         const trace_event_set_t *restrict, int *restrict);
14666       int  posix_trace_eventtypelist_getnext_id(trace_id_t,
14667         trace_event_id_t *restrict, int *restrict);
14668       int  posix_trace_eventtypelist_rewind(trace_id_t);
14669 TRL    int  posix_trace_flush(trace_id_t);
14670       int  posix_trace_get_attr(trace_id_t, trace_attr_t *);
14671 TEF    int  posix_trace_get_filter(trace_id_t, trace_event_set_t *);
14672       int  posix_trace_get_status(trace_id_t,
14673         struct posix_trace_status_info *);
14674       int  posix_trace_getnext_event(trace_id_t,
14675         struct posix_trace_event_info *restrict, void *restrict,
14676         size_t, size_t *restrict, int *restrict);
14677 TRL    int  posix_trace_open(int, trace_id_t *);
14678       int  posix_trace_rewind(trace_id_t);
14679 TEF    int  posix_trace_set_filter(trace_id_t, const trace_event_set_t *, int);
14680       int  posix_trace_shutdown(trace_id_t);
14681       int  posix_trace_start(trace_id_t);
14682       int  posix_trace_stop(trace_id_t);
14683       int  posix_trace_timedgetnext_event(trace_id_t,
14684         struct posix_trace_event_info *restrict, void *restrict,
14685         size_t, size_t *restrict, int *restrict,
14686         const struct timespec *restrict);
14687 TEF    int  posix_trace_trid_eventid_open(trace_id_t, const char *restrict,
14688         trace_event_id_t *restrict);
14689       int  posix_trace_trygetnext_event(trace_id_t,
14690         struct posix_trace_event_info *restrict, void *restrict, size_t,
14691         size_t *restrict, int *restrict);

```

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

The &lt;trace.h&gt; header may be removed in a future version.

**SEE ALSO**[<sys/types.h>](#)

XSH Section 2.11 (on page 536), [posix\\_trace\\_attr\\_destroy\(\)](#), [posix\\_trace\\_attr\\_getclockres\(\)](#), [posix\\_trace\\_attr\\_getinherited\(\)](#), [posix\\_trace\\_attr\\_getlogsize\(\)](#), [posix\\_trace\\_clear\(\)](#), [posix\\_trace\\_close\(\)](#), [posix\\_trace\\_create\(\)](#), [posix\\_trace\\_event\(\)](#), [posix\\_trace\\_eventid\\_equal\(\)](#), [posix\\_trace\\_eventset\\_add\(\)](#), [posix\\_trace\\_eventtypelist\\_getnext\\_id\(\)](#), [posix\\_trace\\_get\\_attr\(\)](#), [posix\\_trace\\_get\\_filter\(\)](#), [posix\\_trace\\_getnext\\_event\(\)](#), [posix\\_trace\\_start\(\)](#)

14705 **CHANGE HISTORY**

14706 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

14707 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/40 is applied, adding the TRL margin  
14708 code to the *posix\_trace\_flush()* function, for alignment with the System Interfaces volume of  
14709 POSIX.1-2008.

14710 **Issue 7**

14711 SD5-XBD-ERN-56 is applied, adding a reference to <sys/types.h> for the **size\_t** type.

14712 The <trace.h> header is marked obsolescent.

14713 This reference page is clarified with respect to macros and symbolic constants.

14714 **NAME**

14715       ulimit.h — ulimit commands

14716 **SYNOPSIS**14717 OB XSI `#include <ulimit.h>`14718 **DESCRIPTION**14719       The <ulimit.h> header shall define the symbolic constants used by the *ulimit()* function.

14720       Symbolic constants:

14721       UL\_GETFSIZE    Get maximum file size.

14722       UL\_SETFSIZE    Set maximum file size.

14723       The following shall be declared as a function and may also be defined as a macro. A function  
14724       prototype shall be provided.

14725       long ulimit(int, ...);

14726 **APPLICATION USAGE**14727       See *ulimit()*.14728 **RATIONALE**

14729       None.

14730 **FUTURE DIRECTIONS**

14731       None.

14732 **SEE ALSO**14733       XSH *ulimit()*14734 **CHANGE HISTORY**

14735       First released in Issue 3.

14736 **Issue 7**

14737       The &lt;ulimit.h&gt; header is marked obsolescent.

**NAME**

unistd.h — standard symbolic constants and types

**SYNOPSIS**

```
#include <unistd.h>
```

**DESCRIPTION**

The <unistd.h> header defines miscellaneous symbolic constants and types, and declares miscellaneous functions. The actual values of the constants are unspecified except as shown. The contents of this header are shown below.

**Version Test Macros**

The <unistd.h> header shall define the following symbolic constants. The values shall be suitable for use in **#if** preprocessing directives.

**\_POSIX\_VERSION**

Integer value indicating version of this standard (C-language binding) to which the implementation conforms. For implementations conforming to POSIX.1-2008, the value shall be 200809L.

**\_POSIX2\_VERSION**

Integer value indicating version of the Shell and Utilities volume of POSIX.1 to which the implementation conforms. For implementations conforming to POSIX.1-2008, the value shall be 200809L. For profile implementations that define **\_POSIX\_SUBPROFILE** (see [Section 2.1.5.1](#)) in <unistd.h>, **\_POSIX2\_VERSION** may be left undefined or be defined with the value **-1** to indicate that the Shell and Utilities volume of POSIX.1 is not supported. In this case, a call to *sysconf(\_SC\_2\_VERSION)* shall return either 200809L or **-1** indicating that the Shell and Utilities volume of POSIX.1 is or is not, respectively, supported at runtime.

The <unistd.h> header shall define the following symbolic constant only if the implementation supports the XSI option; see [Section 2.1.4](#) (on page 19). If defined, its value shall be suitable for use in **#if** preprocessing directives.

**\_XOPEN\_VERSION**

Integer value indicating version of the X/Open Portability Guide to which the implementation conforms. The value shall be 700.

**Constants for Options and Option Groups**

The following symbolic constants, if defined in <unistd.h>, shall have a value of **-1**, **0**, or greater, unless otherwise specified below. For profile implementations that define **\_POSIX\_SUBPROFILE** (see [Section 2.1.5.1](#)) in <unistd.h>, constants described below as always having a value greater than zero need not be defined and, if defined, may have a value of **-1**, **0**, or greater. The values shall be suitable for use in **#if** preprocessing directives.

If a symbolic constant is not defined or is defined with the value **-1**, the option is not supported for compilation. If it is defined with a value greater than zero, the option shall always be supported when the application is executed. If it is defined with the value zero, the option shall be supported for compilation and might or might not be supported at runtime. See [Section 2.1.6](#) (on page 26) for further information about the conformance requirements of these three categories of support.

**\_POSIX\_ADVISORY\_INFO**

The implementation supports the Advisory Information option. If this symbol is defined in <unistd.h>, it shall be defined to be **-1**, **0**, or 200809L. The value of this symbol reported by *sysconf()* shall either be **-1** or 200809L.

14783		<b>_POSIX_ASYNCHRONOUS_IO</b>
14784		The implementation supports asynchronous input and output. This symbol shall always be
14785		set to the value 200809L.
14786		<b>_POSIX_BARRIERS</b>
14787		The implementation supports barriers. This symbol shall always be set to the value
14788		200809L.
14789		<b>_POSIX_CHOWN_RESTRICTED</b>
14790		The use of <i>chown()</i> and <i>fchown()</i> is restricted to a process with appropriate privileges, and
14791		to changing the group ID of a file only to the effective group ID of the process or to one of
14792		its supplementary group IDs. This symbol shall be defined with a value other than -1.
14793		<b>_POSIX_CLOCK_SELECTION</b>
14794		The implementation supports clock selection. This symbol shall always be set to the value
14795		200809L.
14796	CPT	<b>_POSIX_CPUTIME</b>
14797		The implementation supports the Process CPU-Time Clocks option. If this symbol is
14798		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14799		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14800	FSC	<b>_POSIX_FSYNC</b>
14801		The implementation supports the File Synchronization option. If this symbol is defined in
14802		<unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
14803		<i>sysconf()</i> shall either be -1 or 200809L.
14804	IP6	<b>_POSIX_IPV6</b>
14805		The implementation supports the IPv6 option. If this symbol is defined in <unistd.h>, it
14806		shall be defined to be -1, 0, or 200809L. The value of this symbol reported by <i>sysconf()</i> shall
14807		either be -1 or 200809L.
14808		<b>_POSIX_JOB_CONTROL</b>
14809		The implementation supports job control. This symbol shall always be set to a value greater
14810		than zero.
14811		<b>_POSIX_MAPPED_FILES</b>
14812		The implementation supports memory mapped Files. This symbol shall always be set to the
14813		value 200809L.
14814	ML	<b>_POSIX_MEMLOCK</b>
14815		The implementation supports the Process Memory Locking option. If this symbol is defined
14816		in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported
14817		by <i>sysconf()</i> shall either be -1 or 200809L.
14818	MLR	<b>_POSIX_MEMLOCK_RANGE</b>
14819		The implementation supports the Range Memory Locking option. If this symbol is defined
14820		in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported
14821		by <i>sysconf()</i> shall either be -1 or 200809L.
14822		<b>_POSIX_MEMORY_PROTECTION</b>
14823		The implementation supports memory protection. This symbol shall always be set to the
14824		value 200809L.
14825	MSG	<b>_POSIX_MESSAGE_PASSING</b>
14826		The implementation supports the Message Passing option. If this symbol is defined in
14827		<unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
14828		<i>sysconf()</i> shall either be -1 or 200809L.

14829	MON	<b>_POSIX_MONOTONIC_CLOCK</b>
14830		The implementation supports the Monotonic Clock option. If this symbol is defined in
14831		<unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
14832		sysconf() shall either be -1 or 200809L.
14833		<b>_POSIX_NO_TRUNC</b>
14834		Pathname components longer than {NAME_MAX} generate an error. This symbol shall be
14835		defined with a value other than -1.
14836	PIO	<b>_POSIX_PRIORITIZED_IO</b>
14837		The implementation supports the Prioritized Input and Output option. If this symbol is
14838		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14839		reported by sysconf() shall either be -1 or 200809L.
14840	PS	<b>_POSIX_PRIORITY_SCHEDULING</b>
14841		The implementation supports the Process Scheduling option. If this symbol is defined in
14842		<unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
14843		sysconf() shall either be -1 or 200809L.
14844	RS	<b>_POSIX_RAW_SOCKETS</b>
14845		The implementation supports the Raw Sockets option. If this symbol is defined in
14846		<unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
14847		sysconf() shall either be -1 or 200809L.
14848		<b>_POSIX_READER_WRITER_LOCKS</b>
14849		The implementation supports read-write locks. This symbol shall always be set to the value
14850		200809L.
14851		<b>_POSIX_REALTIME_SIGNALS</b>
14852		The implementation supports realtime signals. This symbol shall always be set to the value
14853		200809L.
14854		<b>_POSIX_REGEX</b>
14855		The implementation supports the Regular Expression Handling option. This symbol shall
14856		always be set to a value greater than zero.
14857		<b>_POSIX_SAVED_IDS</b>
14858		Each process has a saved set-user-ID and a saved set-group-ID. This symbol shall always be
14859		set to a value greater than zero.
14860		<b>_POSIX_SEMAPHORES</b>
14861		The implementation supports semaphores. This symbol shall always be set to the value
14862		200809L.
14863	SHM	<b>_POSIX_SHARED_MEMORY_OBJECTS</b>
14864		The implementation supports the Shared Memory Objects option. If this symbol is defined
14865		in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported
14866		by sysconf() shall either be -1 or 200809L.
14867		<b>_POSIX_SHELL</b>
14868		The implementation supports the POSIX shell. This symbol shall always be set to a value
14869		greater than zero.
14870	SPN	<b>_POSIX_SPAWN</b>
14871		The implementation supports the Spawn option. If this symbol is defined in <unistd.h>, it
14872		shall be defined to be -1, 0, or 200809L. The value of this symbol reported by sysconf() shall
14873		either be -1 or 200809L.

14874		<b>_POSIX_SPIN_LOCKS</b>
14875		The implementation supports spin locks. This symbol shall always be set to the value
14876		200809L.
14877	SS	<b>_POSIX_SPORADIC_SERVER</b>
14878		The implementation supports the Process Sporadic Server option. If this symbol is defined
14879		in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported
14880		by <i>sysconf()</i> shall either be -1 or 200809L.
14881	SIO	<b>_POSIX_SYNCHRONIZED_IO</b>
14882		The implementation supports the Synchronized Input and Output option. If this symbol is
14883		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14884		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14885	TSA	<b>_POSIX_THREAD_ATTR_STACKADDR</b>
14886		The implementation supports the Thread Stack Address Attribute option. If this symbol is
14887		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14888		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14889	TSS	<b>_POSIX_THREAD_ATTR_STACKSIZE</b>
14890		The implementation supports the Thread Stack Size Attribute option. If this symbol is
14891		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14892		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14893	TCT	<b>_POSIX_THREAD_CPU_TIME</b>
14894		The implementation supports the Thread CPU-Time Clocks option. If this symbol is
14895		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14896		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14897	TPI	<b>_POSIX_THREAD_PRIO_INHERIT</b>
14898		The implementation supports the Non-Robust Mutex Priority Inheritance option. If this
14899		symbol is defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this
14900		symbol reported by <i>sysconf()</i> shall either be -1 or 200809L.
14901	TPP	<b>_POSIX_THREAD_PRIO_PROTECT</b>
14902		The implementation supports the Non-Robust Mutex Priority Protection option. If this
14903		symbol is defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this
14904		symbol reported by <i>sysconf()</i> shall either be -1 or 200809L.
14905	TPS	<b>_POSIX_THREAD_PRIORITY_SCHEDULING</b>
14906		The implementation supports the Thread Execution Scheduling option. If this symbol is
14907		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14908		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14909	TSH	<b>_POSIX_THREAD_PROCESS_SHARED</b>
14910		The implementation supports the Thread Process-Shared Synchronization option. If this
14911		symbol is defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this
14912		symbol reported by <i>sysconf()</i> shall either be -1 or 200809L.
14913	RPI	<b>_POSIX_THREAD_ROBUST_PRIO_INHERIT</b>
14914		The implementation supports the Robust Mutex Priority Inheritance option. If this symbol
14915		is defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14916		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14917	RPP	<b>_POSIX_THREAD_ROBUST_PRIO_PROTECT</b>
14918		The implementation supports the Robust Mutex Priority Protection option. If this symbol is
14919		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol



14920		reported by <i>sysconf()</i> shall either be <code>-1</code> or <code>200809L</code> .
14921		<code>_POSIX_THREAD_SAFE_FUNCTIONS</code>
14922		The implementation supports thread-safe functions. This symbol shall always be set to the
14923		value <code>200809L</code> .
14924	TSP	<code>_POSIX_THREAD_SPORADIC_SERVER</code>
14925		The implementation supports the Thread Sporadic Server option. If this symbol is defined
14926		in <code>&lt;unistd.h&gt;</code> , it shall be defined to be <code>-1</code> , <code>0</code> , or <code>200809L</code> . The value of this symbol reported
14927		by <i>sysconf()</i> shall either be <code>-1</code> or <code>200809L</code> .
14928		<code>_POSIX_THREADS</code>
14929		The implementation supports threads. This symbol shall always be set to the value
14930		<code>200809L</code> .
14931		<code>_POSIX_TIMEOUTS</code>
14932		The implementation supports timeouts. This symbol shall always be set to the value
14933		<code>200809L</code> .
14934		<code>_POSIX_TIMERS</code>
14935		The implementation supports timers. This symbol shall always be set to the value <code>200809L</code> .
14936	OB TRC	<code>_POSIX_TRACE</code>
14937		The implementation supports the Trace option. If this symbol is defined in <code>&lt;unistd.h&gt;</code> , it
14938		shall be defined to be <code>-1</code> , <code>0</code> , or <code>200809L</code> . The value of this symbol reported by <i>sysconf()</i> shall
14939		either be <code>-1</code> or <code>200809L</code> .
14940	OB TEF	<code>_POSIX_TRACE_EVENT_FILTER</code>
14941		The implementation supports the Trace Event Filter option. If this symbol is defined in
14942		<code>&lt;unistd.h&gt;</code> , it shall be defined to be <code>-1</code> , <code>0</code> , or <code>200809L</code> . The value of this symbol reported by
14943		<i>sysconf()</i> shall either be <code>-1</code> or <code>200809L</code> .
14944	OB TRI	<code>_POSIX_TRACE_INHERIT</code>
14945		The implementation supports the Trace Inherit option. If this symbol is defined in
14946		<code>&lt;unistd.h&gt;</code> , it shall be defined to be <code>-1</code> , <code>0</code> , or <code>200809L</code> . The value of this symbol reported by
14947		<i>sysconf()</i> shall either be <code>-1</code> or <code>200809L</code> .
14948	OB TRL	<code>_POSIX_TRACE_LOG</code>
14949		The implementation supports the Trace Log option. If this symbol is defined in <code>&lt;unistd.h&gt;</code> ,
14950		it shall be defined to be <code>-1</code> , <code>0</code> , or <code>200809L</code> . The value of this symbol reported by <i>sysconf()</i>
14951		shall either be <code>-1</code> or <code>200809L</code> .
14952	TYM	<code>_POSIX_TYPED_MEMORY_OBJECTS</code>
14953		The implementation supports the Typed Memory Objects option. If this symbol is defined
14954		in <code>&lt;unistd.h&gt;</code> , it shall be defined to be <code>-1</code> , <code>0</code> , or <code>200809L</code> . The value of this symbol reported
14955		by <i>sysconf()</i> shall either be <code>-1</code> or <code>200809L</code> .
14956	OB	<code>_POSIX_V6_ILP32_OFF32</code>
14957		The implementation provides a C-language compilation environment with 32-bit <b>int</b> , <b>long</b> ,
14958		<b>pointer</b> , and <b>off_t</b> types.
14959	OB	<code>_POSIX_V6_ILP32_OFFBIG</code>
14960		The implementation provides a C-language compilation environment with 32-bit <b>int</b> , <b>long</b> ,
14961		and <b>pointer</b> types and an <b>off_t</b> type using at least 64 bits.
14962	OB	<code>_POSIX_V6_LP64_OFF64</code>
14963		The implementation provides a C-language compilation environment with 32-bit <b>int</b> and
14964		64-bit <b>long</b> , <b>pointer</b> , and <b>off_t</b> types.

14965	OB	<b>_POSIX_V6_LPBIG_OFFBIG</b>
14966		The implementation provides a C-language compilation environment with an <b>int</b> type
14967		using at least 32 bits and <b>long</b> , <b>pointer</b> , and <b>off_t</b> types using at least 64 bits.
14968		<b>_POSIX_V7_ILP32_OFF32</b>
14969		The implementation provides a C-language compilation environment with 32-bit <b>int</b> , <b>long</b> ,
14970		<b>pointer</b> , and <b>off_t</b> types.
14971		<b>_POSIX_V7_ILP32_OFFBIG</b>
14972		The implementation provides a C-language compilation environment with 32-bit <b>int</b> , <b>long</b> ,
14973		and <b>pointer</b> types and an <b>off_t</b> type using at least 64 bits.
14974		<b>_POSIX_V7_LP64_OFF64</b>
14975		The implementation provides a C-language compilation environment with 32-bit <b>int</b> and
14976		64-bit <b>long</b> , <b>pointer</b> , and <b>off_t</b> types.
14977		<b>_POSIX_V7_LPBIG_OFFBIG</b>
14978		The implementation provides a C-language compilation environment with an <b>int</b> type
14979		using at least 32 bits and <b>long</b> , <b>pointer</b> , and <b>off_t</b> types using at least 64 bits.
14980		<b>_POSIX2_C_BIND</b>
14981		The implementation supports the C-Language Binding option. This symbol shall always
14982		have the value 200809L.
14983	CD	<b>_POSIX2_C_DEV</b>
14984		The implementation supports the C-Language Development Utilities option. If this symbol
14985		is defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14986		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14987		<b>_POSIX2_CHAR_TERM</b>
14988		The implementation supports the Terminal Characteristics option. The value of this symbol
14989		reported by <i>sysconf()</i> shall either be -1 or a value greater than zero.
14990	FD	<b>_POSIX2_FORT_DEV</b>
14991		The implementation supports the FORTRAN Development Utilities option. If this symbol
14992		is defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14993		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14994	FR	<b>_POSIX2_FORT_RUN</b>
14995		The implementation supports the FORTRAN Runtime Utilities option. If this symbol is
14996		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
14997		reported by <i>sysconf()</i> shall either be -1 or 200809L.
14998		<b>_POSIX2_LOCALEDEF</b>
14999		The implementation supports the creation of locales by the <i>localedef</i> utility. If this symbol is
15000		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
15001		reported by <i>sysconf()</i> shall either be -1 or 200809L.
15002	OB BE	<b>_POSIX2_PBS</b>
15003		The implementation supports the Batch Environment Services and Utilities option. If this
15004		symbol is defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this
15005		symbol reported by <i>sysconf()</i> shall either be -1 or 200809L.
15006	OB BE	<b>_POSIX2_PBS_ACCOUNTING</b>
15007		The implementation supports the Batch Accounting option. If this symbol is defined in
15008		<unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
15009		<i>sysconf()</i> shall either be -1 or 200809L.

15010	OB BE	<b>_POSIX2_PBS_CHECKPOINT</b>
15011		The implementation supports the Batch Checkpoint/Restart option. If this symbol is
15012		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
15013		reported by <i>sysconf()</i> shall either be -1 or 200809L.
15014	OB BE	<b>_POSIX2_PBS_LOCATE</b>
15015		The implementation supports the Locate Batch Job Request option. If this symbol is defined
15016		in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported
15017		by <i>sysconf()</i> shall either be -1 or 200809L.
15018	OB BE	<b>_POSIX2_PBS_MESSAGE</b>
15019		The implementation supports the Batch Job Message Request option. If this symbol is
15020		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
15021		reported by <i>sysconf()</i> shall either be -1 or 200809L.
15022	OB BE	<b>_POSIX2_PBS_TRACK</b>
15023		The implementation supports the Track Batch Job Request option. If this symbol is defined
15024		in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported
15025		by <i>sysconf()</i> shall either be -1 or 200809L.
15026	SD	<b>_POSIX2_SW_DEV</b>
15027		The implementation supports the Software Development Utilities option. If this symbol is
15028		defined in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol
15029		reported by <i>sysconf()</i> shall either be -1 or 200809L.
15030	UP	<b>_POSIX2_UPE</b>
15031		The implementation supports the User Portability Utilities option. If this symbol is defined
15032		in <unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported
15033		by <i>sysconf()</i> shall either be -1 or 200809L.
15034	XSI	<b>_XOPEN_CRYPT</b>
15035		The implementation supports the X/Open Encryption Option Group.
15036		<b>_XOPEN_ENH_I18N</b>
15037		The implementation supports the Issue 4, Version 2 Enhanced Internationalization Option
15038		Group. This symbol shall always be set to a value other than -1.
15039		<b>_XOPEN_REALTIME</b>
15040		The implementation supports the X/Open Realtime Option Group.
15041		<b>_XOPEN_REALTIME_THREADS</b>
15042		The implementation supports the X/Open Realtime Threads Option Group.
15043		<b>_XOPEN_SHM</b>
15044		The implementation supports the Issue 4, Version 2 Shared Memory Option Group. This
15045		symbol shall always be set to a value other than -1.
15046	OB XSR	<b>_XOPEN_STREAMS</b>
15047		The implementation supports the XSI STREAMS Option Group.
15048	XSI	<b>_XOPEN_UNIX</b>
15049		The implementation supports the XSI option.
15050	UU	<b>_XOPEN_UUCP</b>
15051		The implementation supports the UUCP Utilities option. If this symbol is defined in
15052		<unistd.h>, it shall be defined to be -1, 0, or 200809L. The value of this symbol reported by
15053		<i>sysconf()</i> shall be either -1 or 200809L.

**Execution-Time Symbolic Constants**

If any of the following symbolic constants are not defined in the <unistd.h> header, the value shall vary depending on the file to which it is applied. If defined, they shall have values suitable for use in **#if** preprocessing directives.

If any of the following symbolic constants are defined to have value `-1` in the <unistd.h> header, the implementation shall not provide the option on any file; if any are defined to have a value other than `-1` in the <unistd.h> header, the implementation shall provide the option on all applicable files.

All of the following values, whether defined as symbolic constants in <unistd.h> or not, may be queried with respect to a specific file using the *pathconf()* or *fpathconf()* functions:

`_POSIX_ASYNC_IO`

Asynchronous input or output operations may be performed for the associated file.

`_POSIX_PRIO_IO`

Prioritized input or output operations may be performed for the associated file.

`_POSIX_SYNC_IO`

Synchronized input or output operations may be performed for the associated file.

If the following symbolic constants are defined in the <unistd.h> header, they apply to files and all paths in all file systems on the implementation:

`_POSIX_TIMESTAMP_RESOLUTION`

The resolution in nanoseconds for all file timestamps.

`_POSIX2_SYMLINKS`

Symbolic links can be created.

**Constants for Functions**

The <unistd.h> header shall define `NULL` as described in <stddef.h>.

The <unistd.h> header shall define the following symbolic constants for use with the *access()* function. The values shall be suitable for use in **#if** preprocessing directives.

`F_OK`                Test for existence of file.

`R_OK`                Test for read permission.

`W_OK`                Test for write permission.

`X_OK`                Test for execute (search) permission.

The constants `F_OK`, `R_OK`, `W_OK`, and `X_OK` and the expressions `R_OK | W_OK`, `R_OK | X_OK`, and `R_OK | W_OK | X_OK` shall all have distinct values.

The <unistd.h> header shall define the following symbolic constants for the *confstr()* function:

`_CS_PATH`

This is the value for the *PATH* environment variable that finds all of the standard utilities that are provided in a manner accessible via the *exec* family of functions.

`_CS_POSIX_V7_ILP32_OFF32_CFLAGS`

If *sysconf(\_SC\_V7\_ILP32\_OFF32)* returns `-1`, the meaning of this value is unspecified. Otherwise, this value is the set of initial options to be given to the *c99* utility to build an application using a programming model with 32-bit **int**, **long**, **pointer**, and **off\_t** types.

15094 `_CS_POSIX_V7_ILP32_OFF32_LDFLAGS`  
 15095 If `sysconf(_SC_V7_ILP32_OFF32)` returns `-1`, the meaning of this value is unspecified.  
 15096 Otherwise, this value is the set of final options to be given to the `c99` utility to build an  
 15097 application using a programming model with 32-bit **int**, **long**, **pointer**, and **off\_t** types.

15098 `_CS_POSIX_V7_ILP32_OFF32_LIBS`  
 15099 If `sysconf(_SC_V7_ILP32_OFF32)` returns `-1`, the meaning of this value is unspecified.  
 15100 Otherwise, this value is the set of libraries to be given to the `c99` utility to build an  
 15101 application using a programming model with 32-bit **int**, **long**, **pointer**, and **off\_t** types.

15102 `_CS_POSIX_V7_ILP32_OFFBIG_CFLAGS`  
 15103 If `sysconf(_SC_V7_ILP32_OFFBIG)` returns `-1`, the meaning of this value is unspecified.  
 15104 Otherwise, this value is the set of initial options to be given to the `c99` utility to build an  
 15105 application using a programming model with 32-bit **int**, **long**, and **pointer** types, and an  
 15106 **off\_t** type using at least 64 bits.

15107 `_CS_POSIX_V7_ILP32_OFFBIG_LDFLAGS`  
 15108 If `sysconf(_SC_V7_ILP32_OFFBIG)` returns `-1`, the meaning of this value is unspecified.  
 15109 Otherwise, this value is the set of final options to be given to the `c99` utility to build an  
 15110 application using a programming model with 32-bit **int**, **long**, and **pointer** types, and an  
 15111 **off\_t** type using at least 64 bits.

15112 `_CS_POSIX_V7_ILP32_OFFBIG_LIBS`  
 15113 If `sysconf(_SC_V7_ILP32_OFFBIG)` returns `-1`, the meaning of this value is unspecified.  
 15114 Otherwise, this value is the set of libraries to be given to the `c99` utility to build an  
 15115 application using a programming model with 32-bit **int**, **long**, and **pointer** types, and an  
 15116 **off\_t** type using at least 64 bits.

15117 `_CS_POSIX_V7_LP64_OFF64_CFLAGS`  
 15118 If `sysconf(_SC_V7_LP64_OFF64)` returns `-1`, the meaning of this value is unspecified.  
 15119 Otherwise, this value is the set of initial options to be given to the `c99` utility to build an  
 15120 application using a programming model with 32-bit **int** and 64-bit **long**, **pointer**, and **off\_t**  
 15121 types.

15122 `_CS_POSIX_V7_LP64_OFF64_LDFLAGS`  
 15123 If `sysconf(_SC_V7_LP64_OFF64)` returns `-1`, the meaning of this value is unspecified.  
 15124 Otherwise, this value is the set of final options to be given to the `c99` utility to build an  
 15125 application using a programming model with 32-bit **int** and 64-bit **long**, **pointer**, and **off\_t**  
 15126 types.

15127 `_CS_POSIX_V7_LP64_OFF64_LIBS`  
 15128 If `sysconf(_SC_V7_LP64_OFF64)` returns `-1`, the meaning of this value is unspecified.  
 15129 Otherwise, this value is the set of libraries to be given to the `c99` utility to build an  
 15130 application using a programming model with 32-bit **int** and 64-bit **long**, **pointer**, and **off\_t**  
 15131 types.

15132 `_CS_POSIX_V7_LPBIG_OFFBIG_CFLAGS`  
 15133 If `sysconf(_SC_V7_LPBIG_OFFBIG)` returns `-1`, the meaning of this value is unspecified.  
 15134 Otherwise, this value is the set of initial options to be given to the `c99` utility to build an  
 15135 application using a programming model with an **int** type using at least 32 bits and **long**,  
 15136 **pointer**, and **off\_t** types using at least 64 bits.

15137 `_CS_POSIX_V7_LPBIG_OFFBIG_LDFLAGS`  
 15138 If `sysconf(_SC_V7_LPBIG_OFFBIG)` returns `-1`, the meaning of this value is unspecified.  
 15139 Otherwise, this value is the set of final options to be given to the `c99` utility to build an  
 15140 application using a programming model with an **int** type using at least 32 bits and **long**,

15141 **pointer**, and **off\_t** types using at least 64 bits.

15142 **\_CS\_POSIX\_V7\_LPBIG\_OFFBIG\_LIBS**

15143 If *sysconf*(*\_SC\_V7\_LPBIG\_OFFBIG*) returns -1, the meaning of this value is unspecified.

15144 Otherwise, this value is the set of libraries to be given to the *c99* utility to build an

15145 application using a programming model with an **int** type using at least 32 bits and **long**,

15146 **pointer**, and **off\_t** types using at least 64 bits.

15147 **\_CS\_POSIX\_V7\_THREADS\_CFLAGS**

15148 If *sysconf*(*\_SC\_POSIX\_THREADS*) returns -1, the meaning of this value is unspecified.

15149 Otherwise, this value is the set of initial options to be given to the *c99* utility to build a

15150 multi-threaded application. These flags are in addition to those associated with any of the

15151 other *\_CS\_POSIX\_V7\_\*\_CFLAGS* values used to specify particular type size programming

15152 environments.

15153 **\_CS\_POSIX\_V7\_THREADS\_LDFLAGS**

15154 If *sysconf*(*\_SC\_POSIX\_THREADS*) returns -1, the meaning of this value is unspecified.

15155 Otherwise, this value is the set of final options to be given to the *c99* utility to build a multi-

15156 threaded application. These flags are in addition to those associated with any of the other

15157 *\_CS\_POSIX\_V7\_\*\_LDFLAGS* values used to specify particular type size programming

15158 environments.

15159 **\_CS\_POSIX\_V7\_WIDTH\_RESTRICTED\_ENVS**

15160 This value is a <newline>-separated list of names of programming environments supported

15161 by the implementation in which the widths of the **blksize\_t**, **cc\_t**, **mode\_t**, **nfds\_t**, **pid\_t**,

15162 **ptrdiff\_t**, **size\_t**, **speed\_t**, **ssize\_t**, **suseconds\_t**, **tcflag\_t**, **wchar\_t**, and **wint\_t** types are no

15163 greater than the width of type **long**. The format of each name shall be suitable for use with

15164 the *getconf* -v option.

15165 **\_CS\_V7\_ENV**

15166 This is the value that provides the environment variable information (other than that

15167 provided by *\_CS\_PATH*) that is required by the implementation to create a conforming

15168 environment, as described in the implementation's conformance documentation.

15169 OB The following symbolic constants are reserved for compatibility with Issue 6:

15170 **\_CS\_POSIX\_V6\_ILP32\_OFF32\_CFLAGS**

15171 **\_CS\_POSIX\_V6\_ILP32\_OFF32\_LDFLAGS**

15172 **\_CS\_POSIX\_V6\_ILP32\_OFF32\_LIBS**

15173 **\_CS\_POSIX\_V6\_ILP32\_OFFBIG\_CFLAGS**

15174 **\_CS\_POSIX\_V6\_ILP32\_OFFBIG\_LDFLAGS**

15175 **\_CS\_POSIX\_V6\_ILP32\_OFFBIG\_LIBS**

15176 **\_CS\_POSIX\_V6\_LP64\_OFF64\_CFLAGS**

15177 **\_CS\_POSIX\_V6\_LP64\_OFF64\_LDFLAGS**

15178 **\_CS\_POSIX\_V6\_LP64\_OFF64\_LIBS**

15179 **\_CS\_POSIX\_V6\_LPBIG\_OFFBIG\_CFLAGS**

15180 **\_CS\_POSIX\_V6\_LPBIG\_OFFBIG\_LDFLAGS**

15181 **\_CS\_POSIX\_V6\_LPBIG\_OFFBIG\_LIBS**

15182 **\_CS\_POSIX\_V6\_WIDTH\_RESTRICTED\_ENVS**

15183 **\_CS\_V6\_ENV**

15184 The <unistd.h> header shall define *SEEK\_CUR*, *SEEK\_END*, and *SEEK\_SET* as described in

15185 <stdio.h>.



15186 XSI The <unistd.h> header shall define the following symbolic constants as possible values for the  
15187 *function* argument to the *lockf()* function:

15188	F_LOCK	Lock a section for exclusive use.
15189	F_TEST	Test section for locks by other processes.
15190	F_TLOCK	Test and lock a section for exclusive use.
15191	F_ULOCK	Unlock locked sections.

15192 The <unistd.h> header shall define the following symbolic constants for *pathconf()*:

15193	_PC_2_SYMLINKS
15194	_PC_ALLOC_SIZE_MIN
15195	_PC_ASYNC_IO
15196	_PC_CHOWN_RESTRICTED
15197	_PC_FILESIZEBITS
15198	_PC_LINK_MAX
15199	_PC_MAX_CANON
15200	_PC_MAX_INPUT
15201	_PC_NAME_MAX
15202	_PC_NO_TRUNC
15203	_PC_PATH_MAX
15204	_PC_PIPE_BUF
15205	_PC_PRIO_IO
15206	_PC_REC_INCR_XFER_SIZE
15207	_PC_REC_MAX_XFER_SIZE
15208	_PC_REC_MIN_XFER_SIZE
15209	_PC_REC_XFER_ALIGN
15210	_PC_SYMLINK_MAX
15211	_PC_SYNC_IO
15212	_PC_TIMESTAMP_RESOLUTION
15213	_PC_VDISABLE

15214 The <unistd.h> header shall define the following symbolic constants for *sysconf()*:

15215	_SC_2_C_BIND
15216	_SC_2_C_DEV
15217	_SC_2_CHAR_TERM
15218	_SC_2_FORT_DEV
15219	_SC_2_FORT_RUN
15220	_SC_2_LOCALEDEF
15221	_SC_2_PBS
15222	_SC_2_PBS_ACCOUNTING
15223	_SC_2_PBS_CHECKPOINT
15224	_SC_2_PBS_LOCATE
15225	_SC_2_PBS_MESSAGE
15226	_SC_2_PBS_TRACK
15227	_SC_2_SW_DEV
15228	_SC_2_UPE
15229	_SC_2_VERSION
15230	_SC_ADVISORY_INFO
15231	_SC_AIO_LISTIO_MAX
15232	_SC_AIO_MAX

15233 \_SC\_AIO\_PRIO\_DELTA\_MAX  
15234 \_SC\_ARG\_MAX  
15235 \_SC\_ASYNCHRONOUS\_IO  
15236 \_SC\_ATEXIT\_MAX  
15237 \_SC\_BARRIERS  
15238 \_SC\_BC\_BASE\_MAX  
15239 \_SC\_BC\_DIM\_MAX  
15240 \_SC\_BC\_SCALE\_MAX  
15241 \_SC\_BC\_STRING\_MAX  
15242 \_SC\_CHILD\_MAX  
15243 \_SC\_CLK\_TCK  
15244 \_SC\_CLOCK\_SELECTION  
15245 \_SC\_COLL\_WEIGHTS\_MAX  
15246 \_SC\_CPUTIME  
15247 \_SC\_DELAYTIMER\_MAX  
15248 \_SC\_EXPR\_NEST\_MAX  
15249 \_SC\_FSYNC  
15250 \_SC\_GETGR\_R\_SIZE\_MAX  
15251 \_SC\_GETPW\_R\_SIZE\_MAX  
15252 \_SC\_HOST\_NAME\_MAX  
15253 \_SC\_IOV\_MAX  
15254 \_SC\_IPV6  
15255 \_SC\_JOB\_CONTROL  
15256 \_SC\_LINE\_MAX  
15257 \_SC\_LOGIN\_NAME\_MAX  
15258 \_SC\_MAPPED\_FILES  
15259 \_SC\_MEMLOCK  
15260 \_SC\_MEMLOCK\_RANGE  
15261 \_SC\_MEMORY\_PROTECTION  
15262 \_SC\_MESSAGE\_PASSING  
15263 \_SC\_MONOTONIC\_CLOCK  
15264 \_SC\_MQ\_OPEN\_MAX  
15265 \_SC\_MQ\_PRIO\_MAX  
15266 \_SC\_NGROUPS\_MAX  
15267 \_SC\_OPEN\_MAX  
15268 \_SC\_PAGE\_SIZE  
15269 \_SC\_PAGESIZE  
15270 \_SC\_PRIORITIZED\_IO  
15271 \_SC\_PRIORITY\_SCHEDULING  
15272 \_SC\_RAW\_SOCKETS  
15273 \_SC\_RE\_DUP\_MAX  
15274 \_SC\_READER\_WRITER\_LOCKS  
15275 \_SC\_REALTIME\_SIGNALS  
15276 \_SC\_REGEX  
15277 \_SC\_RTSIG\_MAX  
15278 \_SC\_SAVED\_IDS  
15279 \_SC\_SEM\_NSEMS\_MAX  
15280 \_SC\_SEM\_VALUE\_MAX  
15281 \_SC\_SEMAPHORES  
15282 \_SC\_SHARED\_MEMORY\_OBJECTS  
15283 \_SC\_SHELL  
15284 \_SC\_SIGQUEUE\_MAX



```

15285     _SC_SPAWN
15286     _SC_SPIN_LOCKS
15287     _SC_SPORADIC_SERVER
15288     _SC_SS_REPL_MAX
15289     _SC_STREAM_MAX
15290     _SC_SYMLINK_MAX
15291     _SC_SYNCHRONIZED_IO
15292     _SC_THREAD_ATTR_STACKADDR
15293     _SC_THREAD_ATTR_STACKSIZE
15294     _SC_THREAD_CPU_TIME
15295     _SC_THREAD_DESTRUCTOR_ITERATIONS
15296     _SC_THREAD_KEYS_MAX
15297     _SC_THREAD_PRIO_INHERIT
15298     _SC_THREAD_PRIO_PROTECT
15299     _SC_THREAD_PRIORITY_SCHEDULING
15300     _SC_THREAD_PROCESS_SHARED
15301     _SC_THREAD_ROBUST_PRIO_INHERIT
15302     _SC_THREAD_ROBUST_PRIO_PROTECT
15303     _SC_THREAD_SAFE_FUNCTIONS
15304     _SC_THREAD_SPARADIC_SERVER
15305     _SC_THREAD_STACK_MIN
15306     _SC_THREAD_THREADS_MAX
15307     _SC_THREADS
15308     _SC_TIMEOUTS
15309     _SC_TIMER_MAX
15310     _SC_TIMERS
15311     _SC_TRACE
15312     _SC_TRACE_EVENT_FILTER
15313     _SC_TRACE_EVENT_NAME_MAX
15314     _SC_TRACE_INHERIT
15315     _SC_TRACE_LOG
15316     _SC_TRACE_NAME_MAX
15317     _SC_TRACE_SYS_MAX
15318     _SC_TRACE_USER_EVENT_MAX
15319     _SC_TTY_NAME_MAX
15320     _SC_TYPED_MEMORY_OBJECTS
15321     _SC_TZNAME_MAX
15322     _SC_V7_ILP32_OFF32
15323     _SC_V7_ILP32_OFFBIG
15324     _SC_V7_LP64_OFF64
15325     _SC_V7_LPBIG_OFFBIG
15326 OB  _SC_V6_ILP32_OFF32
15327     _SC_V6_ILP32_OFFBIG
15328     _SC_V6_LP64_OFF64
15329     _SC_V6_LPBIG_OFFBIG
15330     _SC_VERSION
15331     _SC_XOPEN_CRYPT
15332     _SC_XOPEN_ENH_I18N
15333     _SC_XOPEN_REALTIME
15334     _SC_XOPEN_REALTIME_THREADS
15335     _SC_XOPEN_SHM
15336     _SC_XOPEN_STREAMS

```

15337        \_SC\_XOPEN\_UNIX  
 15338        \_SC\_XOPEN\_UUCP  
 15339        \_SC\_XOPEN\_VERSION

15340        The two constants `_SC_PAGESIZE` and `_SC_PAGE_SIZE` may be defined to have the same value.

15341        The <unistd.h> header shall define the following symbolic constants for file streams:

15342        STDERR\_FILENO     File number of *stderr*; 2.  
 15343        STDIN\_FILENO      File number of *stdin*; 0.  
 15344        STDOUT\_FILENO     File number of *stdout*; 1.

15345        The <unistd.h> header shall define the following symbolic constant for terminal special  
 15346        character handling:

15347        \_POSIX\_VDISABLE   This symbol shall be defined to be the value of a character that shall  
 15348                            disable terminal special character handling as described in [Section 11.2.6](#)  
 15349                            (on page 212). This symbol shall always be set to a value other than -1.

## 15350        Type Definitions

15351        The <unistd.h> header shall define the `size_t`, `ssize_t`, `uid_t`, `gid_t`, `off_t`, and `pid_t` types as  
 15352        described in <sys/types.h>.

15353        The <unistd.h> header shall define the `intptr_t` type as described in <stdint.h>.

## 15354        Declarations

15355        The following shall be declared as functions and may also be defined as macros. Function  
 15356        prototypes shall be provided.

15357        int                access(const char \*, int);  
 15358        unsigned          alarm(unsigned);  
 15359        int                chdir(const char \*);  
 15360        int                chown(const char \*, uid\_t, gid\_t);  
 15361        int                close(int);  
 15362        size\_t            confstr(int, char \*, size\_t);  
 15363        XSI char           \*crypt(const char \*, const char \*);  
 15364        int                dup(int);  
 15365        int                dup2(int, int);  
 15366        void              \_exit(int);  
 15367        XSI void          encrypt(char [64], int);  
 15368        int                execl(const char \*, const char \*, ...);  
 15369        int                execle(const char \*, const char \*, ...);  
 15370        int                execlp(const char \*, const char \*, ...);  
 15371        int                execv(const char \*, char \*const []);  
 15372        int                execve(const char \*, char \*const [], char \*const []);  
 15373        int                execvp(const char \*, char \*const []);  
 15374        int                faccessat(int, const char \*, int, int);  
 15375        int                fchdir(int);  
 15376        int                fchown(int, uid\_t, gid\_t);  
 15377        int                fchownat(int, const char \*, uid\_t, gid\_t, int);  
 15378        SIO int           fdatasync(int);  
 15379        int                fexecve(int, char \*const [], char \*const []);  
 15380        pid\_t             fork(void);

```

15381      long      fpathconf(int, int);
15382 FSC      int      fsync(int);
15383      int      ftruncate(int, off_t);
15384      char      *getcwd(char *, size_t);
15385      gid_t     getegid(void);
15386      uid_t     geteuid(void);
15387      gid_t     getgid(void);
15388      int      getgroups(int, gid_t []);
15389 XSI      long      gethostid(void);
15390      int      gethostname(char *, size_t);
15391      char      *getlogin(void);
15392      int      getlogin_r(char *, size_t);
15393      int      getopt(int, char * const [], const char *);
15394      pid_t     getpgid(pid_t);
15395      pid_t     getpgrp(void);
15396      pid_t     getpid(void);
15397      pid_t     getppid(void);
15398      pid_t     getsid(pid_t);
15399      uid_t     getuid(void);
15400      int      isatty(int);
15401      int      lchown(const char *, uid_t, gid_t);
15402      int      link(const char *, const char *);
15403      int      linkat(int, const char *, int, const char *, int);
15404 XSI      int      lockf(int, int, off_t);
15405      off_t     lseek(int, off_t, int);
15406 XSI      int      nice(int);
15407      long      pathconf(const char *, int);
15408      int      pause(void);
15409      int      pipe(int [2]);
15410      ssize_t   pread(int, void *, size_t, off_t);
15411      ssize_t   pwrite(int, const void *, size_t, off_t);
15412      ssize_t   read(int, void *, size_t);
15413      ssize_t   readlink(const char *restrict, char *restrict, size_t);
15414      ssize_t   readlinkat(int, const char *restrict, char *restrict, size_t);
15415      int      rmdir(const char *);
15416      int      setegid(gid_t);
15417      int      seteuid(uid_t);
15418      int      setgid(gid_t);
15419      int      setpgid(pid_t, pid_t);
15420 OB XSI      pid_t   setpgrp(void);
15421 XSI      int      setregid(gid_t, gid_t);
15422      int      setreuid(uid_t, uid_t);
15423      pid_t     setsid(void);
15424      int      setuid(uid_t);
15425      unsigned  sleep(unsigned);
15426 XSI      void     swab(const void *restrict, void *restrict, ssize_t);
15427      int      symlink(const char *, const char *);
15428      int      symlinkat(const char *, int, const char *);
15429 XSI      void     sync(void);
15430      long      sysconf(int);
15431      pid_t     tcgetpgrp(int);
15432      int      tcsetpgrp(int, pid_t);

```

```

15433     int            truncate(const char *, off_t);
15434     char          *ttyname(int);
15435     int           ttyname_r(int, char *, size_t);
15436     int           unlink(const char *);
15437     int           unlinkat(int, const char *, int);
15438     ssize_t       write(int, const void *, size_t);

```

15439 OB Implementations may also include the `pthread_atfork()` prototype as defined in `<pthread.h>`.  
 15440 Implementations may also include the `ctermid()` prototype as defined in `<stdio.h>`.

15441 The `<unistd.h>` header shall declare the following external variables:

```

15442     extern char    *optarg;
15443     extern int     opterr, optind, optopt;

```

15444 Inclusion of the `<unistd.h>` header may make visible all symbols from the headers `<stddef.h>`, +  
 15445 `<stdint.h>`, and `<stdio.h>`.

## 15446 APPLICATION USAGE

15447 POSIX.1-2008 only describes the behavior of systems that claim conformance to it. However,  
 15448 application developers who want to write applications that adapt to other versions of this  
 15449 standard (or to systems that do not conform to any POSIX standard) may find it useful to code  
 15450 them so as to conditionally compile different code depending on the value of  
 15451 `_POSIX_VERSION`, for example:

```

15452     #if _POSIX_VERSION >= 200112L
15453     /* Use the newer function that copes with large files. */
15454     off_t pos=ftello(fp);
15455     #else
15456     /* Either this is an old version of POSIX, or _POSIX_VERSION is
15457        not even defined, so use the traditional function. */
15458     long pos=ftell(fp);
15459     #endif

```

15460 Earlier versions of POSIX.1-2008 and of the Single UNIX Specification can be identified by the  
 15461 following macros:

15462 POSIX.1-1988 standard  
 15463 `_POSIX_VERSION == 198808L`

15464 POSIX.1-1990 standard  
 15465 `_POSIX_VERSION == 199009L`

15466 ISO POSIX-1: 1996 standard  
 15467 `_POSIX_VERSION == 199506L`

15468 Single UNIX Specification, Version 1  
 15469 `_XOPEN_UNIX` and `_XOPEN_VERSION == 4`

15470 Single UNIX Specification, Version 2  
 15471 `_XOPEN_UNIX` and `_XOPEN_VERSION == 500`

15472 ISO POSIX-1: 2001 and Single UNIX Specification, Version 3  
 15473 `_POSIX_VERSION == 200112L`, plus (if the XSI option is supported) `_XOPEN_UNIX` and  
 15474 `_XOPEN_VERSION == 600`

15475 POSIX.1-2008 does not make any attempt to define application binary interaction with the  
 15476 underlying operating system. However, application developers may find it useful to query  
 15477 `_SC_VERSION` at runtime via `sysconf()` to determine whether the current version of the

operating system supports the necessary functionality as in the following program fragment:

```
if (sysconf(_SC_VERSION) < 200809L) {
    fprintf(stderr, "POSIX.1-2008 system required, terminating \n");
    exit(1);
}
```

New applications should not use `_XOPEN_SHM` or `_XOPEN_ENH_I18N`.

## RATIONALE

As POSIX.1-2008 evolved, certain options became sufficiently standardized that it was concluded that simply requiring one of the option choices was simpler than retaining the option. However, for backwards-compatibility, the option flags (with required constant values) are retained.

### Version Test Macros

The standard developers considered altering the definition of `_POSIX_VERSION` and removing `_SC_VERSION` from the specification of `sysconf()` since the utility to an application was deemed by some to be minimal, and since the implementation of the functionality is potentially problematic. However, they recognized that support for existing application binaries is a concern to manufacturers, application developers, and the users of implementations conforming to POSIX.1-2008.

While the example using `_SC_VERSION` in the APPLICATION USAGE section does not provide the greatest degree of imaginable utility to the application developer or user, it is arguably better than a **core** file or some other equally obscure result. (It is also possible for implementations to encode and recognize application binaries compiled in various POSIX.1-conforming environments, and modify the semantics of the underlying system to conform to the expectations of the application.) For the reasons outlined in the preceding paragraphs and in the APPLICATION USAGE section, the standard developers elected to retain the `_POSIX_VERSION` and `_SC_VERSION` functionality.

### Compile-Time Symbolic Constants for System-Wide Options

POSIX.1-2008 includes support in certain areas for the newly adopted policy governing options and stubs.

This policy provides flexibility for implementations in how they support options. It also specifies how conforming applications can adapt to different implementations that support different sets of options. It allows the following:

1. If an implementation has no interest in supporting an option, it does not have to provide anything associated with that option beyond the announcement that it does not support it.
2. An implementation can support a partial or incompatible version of an option (as a non-standard extension) as long as it does not claim to support the option.
3. An application can determine whether the option is supported. A strictly conforming application must check this announcement mechanism before first using anything associated with the option.

There is an important implication of this policy. POSIX.1-2008 cannot dictate the behavior of interfaces associated with an option when the implementation does not claim to support the option. In particular, it cannot require that a function associated with an unsupported option will fail if it does not perform as specified. However, this policy does not prevent a standard

from requiring certain functions to always be present, but that they shall always fail on some implementations. The `setpgid()` function in the POSIX.1-1990 standard, for example, is considered appropriate.

The POSIX standards include various options, and the C-language binding support for an option implies that the implementation must supply data types and function interfaces. An application must be able to discover whether the implementation supports each option.

Any application must consider the following three cases for each option:

1. Option never supported.

The implementation advertises at compile time that the option will never be supported. In this case, it is not necessary for the implementation to supply any of the data types or function interfaces that are provided only as part of the option. The implementation might provide data types and functions that are similar to those defined by POSIX.1-2008, but there is no guarantee for any particular behavior.

2. Option always supported.

The implementation advertises at compile time that the option will always be supported. In this case, all data types and function interfaces shall be available and shall operate as specified.

3. Option might or might not be supported.

Some implementations might not provide a mechanism to specify support of options at compile time. In addition, the implementation might be unable or unwilling to specify support or non-support at compile time. In either case, any application that might use the option at runtime must be able to compile and execute. The implementation must provide, at compile time, all data types and function interfaces that are necessary to allow this. In this situation, there must be a mechanism that allows the application to query, at runtime, whether the option is supported. If the application attempts to use the option when it is not supported, the result is unspecified unless explicitly specified otherwise in POSIX.1-2008.

## FUTURE DIRECTIONS

None.

## SEE ALSO

<limits.h>, <stddef.h>, <stdint.h>, <stdio.h>, <sys/socket.h>, <sys/types.h>, <termios.h>, <wctype.h>

XSH `access()`, `alarm()`, `chown()`, `close()`, `confstr()`, `crypt()`, `ctermid()`, `dup()`, `_Exit()`, `encrypt()`, `exec`, `fchdir()`, `fchown()`, `fdatasync()`, `fork()`, `fpathconf()`, `fsync()`, `ftruncate()`, `getcwd()`, `getegid()`, `geteuid()`, `getgid()`, `getgroups()`, `gethostid()`, `gethostname()`, `getlogin()`, `getopt()`, `getpgid()`, `getpgrp()`, `getpid()`, `getppid()`, `getsid()`, `getuid()`, `isatty()`, `lchown()`, `link()`, `lockf()`, `lseek()`, `nice()`, `pause()`, `pipe()`, `read()`, `readlink()`, `rmdir()`, `setegid()`, `seteuid()`, `setgid()`, `setpgid()`, `setpgrp()`, `setregid()`, `setreuid()`, `setsid()`, `setuid()`, `sleep()`, `swab()`, `symlink()`, `sync()`, `sysconf()`, `tcgetpgrp()`, `tcsetpgrp()`, `truncate()`, `ttyname()`, `unlink()`, `write()`

## CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

## Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

The symbolic constants `_XOPEN_REALTIME` and `_XOPEN_REALTIME_THREADS` are added.

\_POSIX2\_C\_BIND, \_XOPEN\_ENH\_I18N, and \_XOPEN\_SHM must now be set to a value other than -1 by a conforming implementation.

Large File System extensions are added.

The type of the argument to *sbrk()* is changed from **int** to **intptr\_t**.

\_XBS\_ constants are added to the list of constants for Options and Option Groups, to the list of constants for the *confstr()* function, and to the list of constants to the *sysconf()* function. These are all marked EX.

## Issue 6

\_POSIX2\_C\_VERSION is removed.

The Open Group Corrigendum U026/4 is applied, adding the prototype for *fdatasync()*.

The Open Group Corrigendum U026/1 is applied, adding the symbols \_SC\_XOPEN\_LEGACY, \_SC\_XOPEN\_REALTIME, and \_SC\_XOPEN\_REALTIME\_THREADS.

The symbols \_XOPEN\_STREAMS and \_SC\_XOPEN\_STREAMS are added to support the XSI STREAMS Option Group.

Text in the DESCRIPTION relating to conformance requirements is moved elsewhere in IEEE Std 1003.1-2001.

The LEGACY symbol \_SC\_PASS\_MAX is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The \_CS\_POSIX\_\* and \_CS\_XBS5\_\* constants are added for the *confstr()* function.
- The \_SC\_XBS5\_\* constants are added for the *sysconf()* function.
- The symbolic constants F\_ULOCK, F\_LOCK, F\_TLOCK, and F\_TEST are added.
- The **uid\_t**, **gid\_t**, **off\_t**, **pid\_t**, and **useconds\_t** types are mandated.

The *gethostname()* prototype is added for sockets.

A new section is added for System-Wide Options.

Function prototypes for *setegid()* and *seteuid()* are added.

Option symbolic constants are added for \_POSIX\_ADVISORY\_INFO, \_POSIX\_CPUTIME, \_POSIX\_SPAWN, \_POSIX\_SPORADIC\_SERVER, \_POSIX\_THREAD\_CPUTIME, \_POSIX\_THREAD\_SPORADIC\_SERVER, and \_POSIX\_TIMEOUTS, and *pathconf()* variables are added for \_PC\_ALLOC\_SIZE\_MIN, \_PC\_REC\_INCR\_XFER\_SIZE, \_PC\_REC\_MAX\_XFER\_SIZE, \_PC\_REC\_MIN\_XFER\_SIZE, and \_PC\_REC\_XFER\_ALIGN for alignment with IEEE Std 1003.1d-1999.

The following are added for alignment with IEEE Std 1003.1j-2000:

- Option symbolic constants \_POSIX\_BARRIERS, \_POSIX\_CLOCK\_SELECTION, \_POSIX\_MONOTONIC\_CLOCK, \_POSIX\_READER\_WRITER\_LOCKS, \_POSIX\_SPIN\_LOCKS, and \_POSIX\_TYPED\_MEMORY\_OBJECTS
- *sysconf()* variables \_SC\_BARRIERS, \_SC\_CLOCK\_SELECTION, \_SC\_MONOTONIC\_CLOCK, \_SC\_READER\_WRITER\_LOCKS, \_SC\_SPIN\_LOCKS, and \_SC\_TYPED\_MEMORY\_OBJECTS

The \_SC\_XBS5 macros associated with the ISO/IEC 9899:1990 standard are marked LEGACY, and new equivalent \_SC\_V6 macros associated with the ISO/IEC 9899:1999 standard are



introduced.

The `getwd()` function is marked LEGACY.

The **restrict** keyword is added to the prototypes for `readlink()` and `swab()`.

Constants for options are now harmonized, so when supported they take the year of approval of IEEE Std 1003.1-2001 as the value.

The following are added for alignment with IEEE Std 1003.1q-2000:

- Optional symbolic constants `_POSIX_TRACE`, `_POSIX_TRACE_EVENT_FILTER`, `_POSIX_TRACE_LOG`, and `_POSIX_TRACE_INHERIT`
- The `sysconf()` symbolic constants `_SC_TRACE`, `_SC_TRACE_EVENT_FILTER`, `_SC_TRACE_LOG`, and `_SC_TRACE_INHERIT`

The `brk()` and `sbrk()` LEGACY functions are removed.

The Open Group Base Resolution bwg2001-006 is applied, which reworks the XSI versioning information.

The Open Group Base Resolution bwg2001-008 is applied, changing the `namelen` parameter for `gethostname()` from `socklen_t` to `size_t`.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/2 is applied, changing “Thread Stack Address Size” to “Thread Stack Size Attribute”.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/20 is applied, adding the `_POSIX_IPV6`, `_SC_V6`, and `_SC_RAW_SOCKETS` symbols.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/21 is applied, correcting the description in “Constants for Functions” for the `_CS_POSIX_V6_LP64_OFF64_CFLAGS`, `_CS_POSIX_V6_LP64_OFF64_LDFLAGS`, and `_CS_POSIX_V6_LP64_OFF64_LIBS` symbols.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/22 is applied, removing the shading for the `_PC*` and `_SC*` constants, since these are mandatory on all implementations.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/23 is applied, adding the `_PC_SYMLINK_MAX` and `_SC_SYMLINK_MAX` constants.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/24 is applied, correcting the shading and margin code for the `fsync()` function.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/25 is applied, adding the following text to the APPLICATION USAGE section: “New applications should not use `_XOPEN_SHM` or `_XOPEN_ENH_I18N`.”.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/29 is applied, clarifying the requirements for when constants for Options and Option Groups can be defined or undefined.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/30 is applied, changing the `_V6_ILP32_OFF32`, `_V6_ILP32_OFFBIG`, `_V6_LP64_OFF64`, and `_V6_LP64_OFFBIG` symbols to `_POSIX_V6_ILP32_OFF32`, `_POSIX_V6_ILP32_OFFBIG`, `_POSIX_V6_LP64_OFF64`, and `_POSIX_V6_LP64_OFFBIG`, respectively. This is for consistency with the `sysconf()` and `c99` reference pages.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/31 is applied, adding that the format of names of programming environments can be obtained using the `getconf -v` option.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/32 is applied, deleting the `_SC_FILE_LOCKING`, `_SC_2_C_VERSION`, and `_SC_XOPEN_XCU_VERSION` constants.



15650 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/33 is applied, adding  
 15651 `_SC_SS_REPL_MAX`, `_SC_TRACE_EVENT_NAME_MAX`, `_SC_TRACE_NAME_MAX`,  
 15652 `_SC_TRACE_SYS_MAX`, and `_SC_TRACE_USER_EVENT_MAX` to the list of symbolic constants  
 15653 for `sysconf()`.

15654 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/34 is applied, updating the prototype for  
 15655 the `symlink()` function to match that in the System Interfaces volume of IEEE Std 1003.1-2001.

15656 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/35 is applied, adding `_PC_2_SYMLINKS`  
 15657 to the symbolic constants list for `pathconf()`. This corresponds to the definition of  
 15658 `POSIX2_SYMLINKS` in the Shell and Utilities volume of IEEE Std 1003.1-2001.

15659 **Issue 7**

15660 Austin Group Interpretations 1003.1-2001 #026 and #047 are applied.

15661 Austin Group Interpretation 1003.1-2001 #166 is applied to permit an additional compiler flag to  
 15662 enable threads.

15663 Austin Group Interpretation 1003.1-2001 #178 is applied, clarifying the values allowed for  
 15664 `_POSIX2_CHAR_TERM`.

15665 SD5-XBD-ERN-41 is applied, adding the `_POSIX2_SYMLINKS` constant.

15666 SD5-XBD-ERN-76 and SD5-XBD-ERN-77 are applied.

15667 Symbols to support the UUCP Utilities option are added.

15668 The variables for the supported programming environments are updated to be V7.

15669 The LEGACY and obsolescent symbols are removed.

15670 The `faccessat()`, `fchownat()`, `fexecve()`, `linkat()`, `readlinkat()`, `symlinkat()`, and `unlinkat()` functions  
 15671 are added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

15672 The `_POSIX_TRACE*` constants from the Trace option are marked obsolescent.

15673 The `_POSIX2_PBS*` constants from the Batch Environment Services and Utilities option are  
 15674 marked obsolescent.

15675 Functionality relating to the Asynchronous Input and Output, Barriers, Clock Selection, Memory  
 15676 Mapped Files, Memory Protection, Realtime Signals Extension, Semaphores, Spin Locks,  
 15677 Threads, Timeouts, and Timers options is moved to the Base.

15678 Functionality relating to the Thread Priority Protection and Thread Priority Inheritance options  
 15679 is changed to be Non-Robust Mutex or Robust Mutex Priority Protection and Non-Robust Mutex  
 15680 or Robust Mutex Priority Inheritance, respectively.

15681 This reference page is clarified with respect to macros and symbolic constants.

15682 Changes are made related to support for finegrained timestamps and the  
 15683 `_POSIX_TIMESTAMP_RESOLUTION` constant is added.

15684 The `_SC_THREAD_ROBUST_PRIO_INHERIT` and `_SC_THREAD_ROBUST_PRIO_PROTECT`  
 15685 symbolic constants are added.

15686 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0078 [311], XBD/TC1-2008/0079 [209],  
 15687 and XBD/TC1-2008/0080 [360] are applied.

15688 **NAME**

15689 utime.h — access and modification times structure

15690 **SYNOPSIS**15691 OB `#include <utime.h>`15692 **DESCRIPTION**15693 The <utime.h> header shall declare the **utimbuf** structure, which shall include the following  
15694 members:15695 `time_t` `actime` Access time.  
15696 `time_t` `modtime` Modification time.

15697 The times shall be measured in seconds since the Epoch.

15698 The <utime.h> header shall define the **time\_t** type as described in <sys/types.h>.15699 The following shall be declared as a function and may also be defined as a macro. A function  
15700 prototype shall be provided.15701 `int utime(const char *, const struct utimbuf *);`15702 **APPLICATION USAGE**15703 The *utime()* function only allows setting file timestamps to the nearest second. Applications  
15704 should use the *utimensat()* function instead. See <sys/stat.h>.15705 **RATIONALE**

15706 None.

15707 **FUTURE DIRECTIONS**

15708 The &lt;utime.h&gt; header may be removed in a future version.

15709 **SEE ALSO**15710 [<sys/stat.h>](#), [<sys/types.h>](#)15711 XSH *futimens()*, *utime()*15712 **CHANGE HISTORY**

15713 First released in Issue 3.

15714 **Issue 6**15715 The following new requirements on POSIX implementations derive from alignment with the  
15716 Single UNIX Specification:

- 15717
- The **time\_t** type is defined.

15718 **Issue 7**

15719 The &lt;utime.h&gt; header is marked obsolescent.

15720 **NAME**

15721 utmpx.h — user accounting database definitions

15722 **SYNOPSIS**15723 XSI `#include <utmpx.h>`15724 **DESCRIPTION**15725 The <utmpx.h> header shall define the **utmpx** structure that shall include at least the following  
15726 members:

15727	char	ut_user[]	User login name.
15728	char	ut_id[]	Unspecified initialization process identifier.
15729	char	ut_line[]	Device name.
15730	pid_t	ut_pid	Process ID.
15731	short	ut_type	Type of entry.
15732	struct timeval	ut_tv	Time entry was made.

15733 The <utmpx.h> header shall define the **pid\_t** type through **typedef**, as described in  
15734 <sys/types.h>.15735 The <utmpx.h> header shall define the **timeval** structure as described in <sys/time.h>.

15736 Inclusion of the &lt;utmpx.h&gt; header may also make visible all symbols from &lt;sys/time.h&gt;.

15737 The <utmpx.h> header shall define the following symbolic constants as possible values for the  
15738 *ut\_type* member of the **utmpx** structure:

15739	EMPTY	No valid user accounting information.
15740	BOOT_TIME	Identifies time of system boot.
15741	OLD_TIME	Identifies time when system clock changed.
15742	NEW_TIME	Identifies time after system clock changed.
15743	USER_PROCESS	Identifies a process.
15744	INIT_PROCESS	Identifies a process spawned by the init process.
15745	LOGIN_PROCESS	Identifies the session leader of a logged-in user.
15746	DEAD_PROCESS	Identifies a session leader who has exited.

15747 The following shall be declared as functions and may also be defined as macros. Function  
15748 prototypes shall be provided.

```

15749 void          endutxent(void);
15750 struct utmpx *getutxent(void);
15751 struct utmpx *getutxid(const struct utmpx *);
15752 struct utmpx *getutxline(const struct utmpx *);
15753 struct utmpx *pututxline(const struct utmpx *);
15754 void          setutxent(void);

```

**15755 APPLICATION USAGE**

15756       None.

**15757 RATIONALE**

15758       None.

**15759 FUTURE DIRECTIONS**

15760       None.

**15761 SEE ALSO**

15762       [\*\*<sys/time.h>\*\*](#), [\*\*<sys/types.h>\*\*](#)

15763       XSH *endutxent()*

**15764 CHANGE HISTORY**

15765       First released in Issue 4, Version 2.

## 15766 NAME

15767       wchar.h — wide-character handling

## 15768 SYNOPSIS

15769       #include &lt;wchar.h&gt;

## 15770 DESCRIPTION

15771 CX       Some of the functionality described on this reference page extends the ISO C standard.  
 15772       Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to  
 15773       enable the visibility of these symbols in this header.

15774       The &lt;wchar.h&gt; header shall define the following types:

15775 CX       **FILE**           As described in <stdio.h>.15776 CX       **locale\_t**       As described in <locale.h>.

15777       **mbstate\_t**       An object type other than an array type that can hold the conversion state  
 15778       information necessary to convert between sequences of (possibly multi-byte)  
 15779 CX       characters and wide characters. If a codeset is being used such that an  
 15780       **mbstate\_t** needs to preserve more than two levels of reserved state, the results  
 15781       are unspecified.

15782       **size\_t**           As described in <stddef.h>.15783 CX       **va\_list**       As described in <stdarg.h>.15784       **wchar\_t**       As described in <stddef.h>.

15785 OB XSI   **wctype\_t**       A scalar type of a data object that can hold values which represent locale-  
 15786       specific character classification.

15787       **wint\_t**       An integer type capable of storing any valid value of **wchar\_t** or WEOF.

15788       The tag **tm** shall be declared as naming an incomplete structure type, the contents of which are  
 15789       described in the <time.h> header.

15790       The implementation shall support one or more programming environments in which the width  
 15791       of **wint\_t** is no greater than the width of type **long**. The names of these programming  
 15792       environments can be obtained using the *confstr()* function or the *getconf* utility.

15793       The &lt;wchar.h&gt; header shall define the following macros:

15794       **WCHAR\_MAX**   As described in <stdint.h>.15795       **WCHAR\_MIN**   As described in <stdint.h>.

15796       **WEOF**           Constant expression of type **wint\_t** that is returned by several WP functions to  
 15797       indicate end-of-file.

15798       **NULL**           As described in <stddef.h>.

15799 CX       Inclusion of the <wchar.h> header may make visible all symbols from the headers <ctype.h>,  
 15800       <string.h>, <stdarg.h>, <stddef.h>, <stdio.h>, <stdlib.h>, and <time.h>.

15801       The following shall be declared as functions and may also be defined as macros. Function  
 15802       prototypes shall be provided for use with ISO C standard compilers. Arguments to functions in  
 15803       this list can point to arrays containing **wchar\_t** values that do not correspond to members of the  
 15804       character set of the current locale. Such values shall be processed according to the specified  
 15805       semantics, unless otherwise stated.

15806       **wint\_t**        btowc(int);15807       **wint\_t**        fgetwc(FILE \*);

```

15808     wchar_t      *fgetws(wchar_t *restrict, int, FILE *restrict);
15809     wint_t      fputwc(wchar_t, FILE *);
15810     int         fputws(const wchar_t *restrict, FILE *restrict);
15811     int         fwide(FILE *, int);
15812     int         fwprintf(FILE *restrict, const wchar_t *restrict, ...);
15813     int         fwscanf(FILE *restrict, const wchar_t *restrict, ...);
15814     wint_t      getwc(FILE *);
15815     wint_t      getwchar(void);
15816     OB XSI     int     iswalnum(wint_t);
15817               int     iswalpha(wint_t);
15818               int     iswcntrl(wint_t);
15819               int     iswctype(wint_t, wctype_t);
15820               int     iswdigit(wint_t);
15821               int     iswgraph(wint_t);
15822               int     iswlower(wint_t);
15823               int     iswprint(wint_t);
15824               int     iswpunct(wint_t);
15825               int     iswspace(wint_t);
15826               int     iswupper(wint_t);
15827               int     iswxdigit(wint_t);
15828     size_t      mbrlen(const char *restrict, size_t, mbstate_t *restrict);
15829     size_t      mbrtowc(wchar_t *restrict, const char *restrict, size_t,
15830                        mbstate_t *restrict);
15831     int         mbsinit(const mbstate_t *);
15832     CX         size_t  mbsnrtowcs(wchar_t *restrict, const char **restrict,
15833                        size_t, size_t, mbstate_t *restrict);
15834     size_t      mbsrtowcs(wchar_t *restrict, const char **restrict, size_t,
15835                        mbstate_t *restrict);
15836     CX         FILE    *open_wmemstream(wchar_t **, size_t *);
15837     wint_t      putwc(wchar_t, FILE *);
15838     wint_t      putwchar(wchar_t);
15839     int         swprintf(wchar_t *restrict, size_t,
15840                        const wchar_t *restrict, ...);
15841     int         swscanf(const wchar_t *restrict,
15842                        const wchar_t *restrict, ...);
15843     OB XSI     wint_t  tolower(wint_t);
15844               wint_t  toupper(wint_t);
15845     wint_t      ungetwc(wint_t, FILE *);
15846     int         vfwprintf(FILE *restrict, const wchar_t *restrict, va_list);
15847     int         vfwscanf(FILE *restrict, const wchar_t *restrict, va_list);
15848     int         vswprintf(wchar_t *restrict, size_t,
15849                        const wchar_t *restrict, va_list);
15850     int         vswscanf(const wchar_t *restrict, const wchar_t *restrict,
15851                        va_list);
15852     int         vwprintf(const wchar_t *restrict, va_list);
15853     int         vwscanf(const wchar_t *restrict, va_list);
15854     CX         wchar_t *wcpcpy(wchar_t *restrict, const wchar_t *restrict);
15855     wchar_t     *wcpncpy(wchar_t *restrict, const wchar_t *restrict, size_t);
15856     size_t      wcrtoomb(char *restrict, wchar_t, mbstate_t *restrict);
15857     CX         int     wcscasecmp(const wchar_t *, const wchar_t *);
15858               int     wcscasecmp_l(const wchar_t *, const wchar_t *, locale_t);
15859     wchar_t     *wcscat(wchar_t *restrict, const wchar_t *restrict);

```

```

15860 wchar_t      *wcschr(const wchar_t *, wchar_t);
15861 int           wscmp(const wchar_t *, const wchar_t *);
15862 int           wscoll(const wchar_t *, const wchar_t *);
15863 CX int        wscoll_l(const wchar_t *, const wchar_t *, locale_t);
15864 wchar_t      *wcscpy(wchar_t *restrict, const wchar_t *restrict);
15865 size_t       wcsncpy(const wchar_t *, const wchar_t *);
15866 CX wchar_t    *wcsdup(const wchar_t *);
15867 size_t       wcsftime(wchar_t *restrict, size_t,
15868                       const wchar_t *restrict, const struct tm *restrict);
15869 size_t       wcslen(const wchar_t *);
15870 CX int        wcsncasecmp(const wchar_t *, const wchar_t *, size_t);
15871 int          wcsncasecmp_l(const wchar_t *, const wchar_t *, size_t,
15872                             locale_t);
15873 wchar_t      *wcsncat(wchar_t *restrict, const wchar_t *restrict, size_t);
15874 int           wcsncmp(const wchar_t *, const wchar_t *, size_t);
15875 wchar_t      *wcsncpy(wchar_t *restrict, const wchar_t *restrict, size_t);
15876 CX size_t     wcsnlen(const wchar_t *, size_t);
15877 size_t       wcsnrtombs(char *restrict, const wchar_t **restrict, size_t,
15878                          size_t, mbstate_t *restrict);
15879 wchar_t      *wcpbrk(const wchar_t *, const wchar_t *);
15880 wchar_t      *wcsrchr(const wchar_t *, wchar_t);
15881 size_t       wcsrtoombs(char *restrict, const wchar_t **restrict,
15882                          size_t, mbstate_t *restrict);
15883 size_t       wcsspncpy(const wchar_t *, const wchar_t *);
15884 wchar_t      *wcstrncpy(const wchar_t *restrict, const wchar_t *restrict);
15885 double       wcstod(const wchar_t *restrict, wchar_t **restrict);
15886 float        wcstof(const wchar_t *restrict, wchar_t **restrict);
15887 wchar_t      *wcstok(wchar_t *restrict, const wchar_t *restrict,
15888                      wchar_t **restrict);
15889 long         wcstol(const wchar_t *restrict, wchar_t **restrict, int);
15890 long double  wcstold(const wchar_t *restrict, wchar_t **restrict);
15891 long long    wcstoll(const wchar_t *restrict, wchar_t **restrict, int);
15892 unsigned long wcstoul(const wchar_t *restrict, wchar_t **restrict, int);
15893 unsigned long long
15894 wcstoull(const wchar_t *restrict, wchar_t **restrict, int);
15895 XSI int       wcswidth(const wchar_t *, size_t);
15896 size_t       wcsxfrm(wchar_t *restrict, const wchar_t *restrict, size_t);
15897 CX size_t     wcsxfrm_l(wchar_t *restrict, const wchar_t *restrict,
15898                          size_t, locale_t);
15899 int          wctob(wint_t);
15900 OB XSI wctype_t wctype(const char *);
15901 XSI int       wcwidth(wchar_t);
15902 wchar_t      *wmemchr(const wchar_t *, wchar_t, size_t);
15903 int          wmemcmp(const wchar_t *, const wchar_t *, size_t);
15904 wchar_t      *wmemcpy(wchar_t *restrict, const wchar_t *restrict, size_t);
15905 wchar_t      *wmemmove(wchar_t *, const wchar_t *, size_t);
15906 wchar_t      *wmemset(wchar_t *, wchar_t, size_t);
15907 int          wprintf(const wchar_t *restrict, ...);
15908 int          wscanf(const wchar_t *restrict, ...);

```

**APPLICATION USAGE**

The `iswblank()` function was a late addition to the ISO C standard and was introduced at the same time as the ISO C standard introduced **<wctype.h>**, which contains all of the `isw*()` functions. The Open Group Base Specifications had previously aligned with the MSE working draft and had introduced the rest of the `isw*()` functions into **<wchar.h>**. For backwards-compatibility, the original set of `isw*()` functions, without `iswblank()`, are permitted (as part of the XSI option) in **<wchar.h>**. For maximum portability, applications should include **<wctype.h>** in order to obtain declarations for the `isw*()` functions. This compatibility has been made obsolescent.

**RATIONALE**

In the ISO C standard, the symbols referenced as XSI extensions are in **<wctype.h>**. Their presence here is thus an extension.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

**<ctype.h>**, **<locale.h>**, **<stdarg.h>**, **<stddef.h>**, **<stdint.h>**, **<stdio.h>**, **<stdlib.h>**, **<string.h>**, **<time.h>**, **<wctype.h>**

XSH Section 2.2 (on page 472), `btowc()`, `confstr()`, `fgetwc()`, `fgetws()`, `fputwc()`, `fputws()`, `fwide()`, `fwprintf()`, `fwscanf()`, `getwc()`, `getwchar()`, `iswalnum()`, `iswalpunct()`, `iswalpha()`, `iswcntrl()`, `iswctype()`, `iswdigit()`, `iswgraph()`, `iswlower()`, `iswprint()`, `iswpunct()`, `iswspace()`, `iswupper()`, `iswxdigit()`, `mbrlen()`, `mbrtowc()`, `mbsinit()`, `mbsrtowcs()`, `open_memstream()`, `putwc()`, `putwchar()`, `towlower()`, `towupper()`, `ungetwc()`, `vfwprintf()`, `vfwscanf()`, `wcrtomb()`, `wcscasecmp()`, `wcscat()`, `wcschr()`, `wcscmp()`, `wcscoll()`, `wcscpy()`, `wcscspn()`, `wcsdup()`, `wcsftime()`, `wcslen()`, `wcsncat()`, `wcsncmp()`, `wcsncpy()`, `wcspbrk()`, `wcsrchr()`, `wcsrtombs()`, `wcsspn()`, `wcsstr()`, `wcstod()`, `wcstok()`, `wcstol()`, `wcstoul()`, `wcswidth()`, `wcsxfrm()`, `wctob()`, `wctype()`, `wcwidth()`, `wmemchr()`, `wmemcmp()`, `wmemncpy()`, `wmemmove()`, `wmemset()`

XCU `getconf`

**CHANGE HISTORY**

First released in Issue 4.

**Issue 5**

Aligned with the ISO/IEC 9899:1990/Amendment 1:1995 (E).

**Issue 6**

The Open Group Corrigendum U021/10 is applied. The prototypes for `wcswidth()` and `wcwidth()` are marked as extensions.

The Open Group Corrigendum U028/5 is applied, correcting the prototype for the `mbsinit()` function.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- Various function prototypes are updated to add the **restrict** keyword.
- The functions `vfwscanf()`, `vwscanf()`, `wcstof()`, `wcstold()`, `wcstoll()`, and `wcstoull()` are added.

The type **wctype\_t**, the `isw*()`, `to*()`, and `wctype()` functions are marked as XSI extensions.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/26 is applied, adding the APPLICATION USAGE section.



15952 **Issue 7**

15953 The *mbsnrtowcs()*, *open\_wmemstream()*, *wcpcpy()*, *wcpncpy()*, *wscasecmp()*, *wcsdup()*,  
15954 *wcsncasecmp()*, *wcsnlen()*, and *wcsnrtombs()* functions are added from The Open Group  
15955 Technical Standard, 2006, Extended API Set Part 1.

15956 The *wscasecmp\_l()*, *wcsncasecmp\_l()*, *wscoll\_l()*, and *wcsxfrm\_l()* functions are added from The  
15957 Open Group Technical Standard, 2006, Extended API Set Part 4.

15958 The **wctype\_t** type, and the *isw\*()*, *towlower()*, and *towupper()* functions are marked obsolescent  
15959 in <wchar.h> since the ISO C standard requires the declarations to be in <wctype.h>.

15960 This reference page is clarified with respect to macros and symbolic constants, and a declaration  
15961 for the **locale\_t** type is added.

15962 POSIX.1-2008, Technical Corrigendum 1, XBD/TC1-2008/0081 [380] is applied.

## 15963 NAME

15964 wctype.h — wide-character classification and mapping utilities

## 15965 SYNOPSIS

15966 #include &lt;wctype.h&gt;

## 15967 DESCRIPTION

15968 CX Some of the functionality described on this reference page extends the ISO C standard.  
 15969 Applications shall define the appropriate feature test macro (see XSH [Section 2.2](#), on page 472) to  
 15970 enable the visibility of these symbols in this header.

15971 The &lt;wctype.h&gt; header shall define the following types:

15972 **wint\_t** As described in <wchar.h>.

15973 **wctrans\_t** A scalar type that can hold values which represent locale-specific character  
 15974 mappings.

15975 **wctype\_t** As described in <wchar.h>.15976 CX The <wctype.h> header shall define the **locale\_t** type as described in <locale.h>.

15977 The &lt;wctype.h&gt; header shall define the following macro:

15978 **WEOF** As described in <wchar.h>.

15979 For all functions described in this header that accept an argument of type **wint\_t**, the value is  
 15980 representable as a **wchar\_t** or equals the value of WEOF. If this argument has any other value,  
 15981 the behavior is undefined.

15982 The behavior of these functions shall be affected by the *LC\_CTYPE* category of the current locale.

15983 CX Inclusion of the <wctype.h> header may make visible all symbols from the headers <ctype.h>,  
 15984 <stdarg.h>, <stddef.h>, <stdio.h>, <stdlib.h>, <string.h>, <time.h>, and <wchar.h>.

15985 The following shall be declared as functions and may also be defined as macros. Function  
 15986 prototypes shall be provided for use with ISO C standard compilers.

```

15987 int      iswalnum(wint_t);
15988 CX int    iswalnum_l(wint_t, locale_t);
15989 int      iswalpha(wint_t);
15990 CX int    iswalpha_l(wint_t, locale_t);
15991 int      iswblank(wint_t);
15992 CX int    iswblank_l(wint_t, locale_t);
15993 int      iswcntrl(wint_t);
15994 CX int    iswcntrl_l(wint_t, locale_t);
15995 int      iswctype(wint_t, wctype_t);
15996 CX int    iswctype_l(wint_t, wctype_t, locale_t);
15997 int      iswdigit(wint_t);
15998 CX int    iswdigit_l(wint_t, locale_t);
15999 int      iswgraph(wint_t);
16000 CX int    iswgraph_l(wint_t, locale_t);
16001 int      iswlower(wint_t);
16002 CX int    iswlower_l(wint_t, locale_t);
16003 int      iswprint(wint_t);
16004 CX int    iswprint_l(wint_t, locale_t);
16005 int      iswpunct(wint_t);
16006 CX int    iswpunct_l(wint_t, locale_t);
16007 int      iswspace(wint_t);
  
```

```

16008 CX      int      iswspace_l(wint_t, locale_t);
16009          int      iswupper(wint_t);
16010 CX      int      iswupper_l(wint_t, locale_t);
16011          int      iswxdigit(wint_t);
16012 CX      int      iswxdigit_l(wint_t, locale_t);
16013          wint_t    towctrans(wint_t, wctrans_t);
16014 CX      wint_t    towctrans_l(wint_t, wctrans_t, locale_t);
16015          wint_t    towlower(wint_t);
16016 CX      wint_t    towlower_l(wint_t, locale_t);
16017          wint_t    towupper(wint_t);
16018 CX      wint_t    towupper_l(wint_t, locale_t);
16019          wctrans_t wctrans(const char *);
16020 CX      wctrans_t wctrans_l(const char *, locale_t);
16021          wctype_t  wctype(const char *);
16022 CX      wctype_t  wctype_l(const char *, locale_t);

```

## 16023 APPLICATION USAGE

16024 None.

## 16025 RATIONALE

16026 None.

## 16027 FUTURE DIRECTIONS

16028 None.

## 16029 SEE ALSO

16030 <ctype.h>, <locale.h>, <stdarg.h>, <stddef.h>, <stdio.h>, <stdlib.h>, <string.h>, <time.h>,  
 16031 <wchar.h>

16032 XSH Section 2.2 (on page 472), *iswalnum()*, *iswalpha()*, *iswblank()*, *iswcntrl()*, *iswctype()*,  
 16033 *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*, *iswspace()*, *iswupper()*, *iswxdigit()*,  
 16034 *setlocale()*, *towctrans()*, *towlower()*, *towupper()*, *wctrans()*, *wctype()*

## 16035 CHANGE HISTORY

16036 First released in Issue 5. Derived from the ISO/IEC 9899:1990/Amendment 1:1995 (E).

### 16037 Issue 6

16038 The *iswblank()* function is added for alignment with the ISO/IEC 9899:1999 standard.

### 16039 Issue 7

16040 SD5-XBD-ERN-6 is applied.

16041 The *\*\_l()* functions are added from The Open Group Technical Standard, 2006, Extended API Set  
 16042 Part 4.

16043 This reference page is clarified with respect to macros and symbolic constants.

## 16044 NAME

16045 wordexp.h — word-expansion types

## 16046 SYNOPSIS

16047 #include &lt;wordexp.h&gt;

## 16048 DESCRIPTION

16049 The <wordexp.h> header shall define the structures and symbolic constants used by the  
16050 *wordexp()* and *wordfree()* functions.16051 The <wordexp.h> header shall define the **wordexp\_t** structure type, which shall include at least  
16052 the following members:

16053 `size_t we_wordc` Count of words matched by *words*.  
 16054 `char **we_wordv` Pointer to list of expanded words.  
 16055 `size_t we_offs` Slots to reserve at the beginning of *we\_wordv*.

16056 The <wordexp.h> header shall define the following symbolic constants for use as flags for the  
16057 *wordexp()* function:

16058 **WRDE\_APPEND** Append words to those previously generated.  
 16059 **WRDE\_DOOFFS** Number of null pointers to prepend to *we\_wordv*.  
 16060 **WRDE\_NOCMD** Fail if command substitution is requested.  
 16061 **WRDE\_REUSE** The *pwordexp* argument was passed to a previous successful call to  
 16062 *wordexp()*, and has not been passed to *wordfree()*. The result is the same  
 16063 as if the application had called *wordfree()* and then called *wordexp()*  
 16064 without **WRDE\_REUSE**.

16065 **WRDE\_SHOWERR** Do not redirect *stderr* to **/dev/null**.16066 **WRDE\_UNDEF** Report error on an attempt to expand an undefined shell variable.

16067 The &lt;wordexp.h&gt; header shall define the following symbolic constants as error return values:

16068 **WRDE\_BADCHAR** One of the unquoted characters—<newline>, ' | ', '&', ' ; ', '<', '>',  
 16069 ' ( ', ' ) ', ' { ', ' } '—appears in *words* in an inappropriate context.  
 16070 **WRDE\_BADVAL** Reference to undefined shell variable when **WRDE\_UNDEF** is set in *flags*.  
 16071 **WRDE\_CMDSUB** Command substitution requested when **WRDE\_NOCMD** was set in *flags*.  
 16072 **WRDE\_NOSPACE** Attempt to allocate memory failed.  
 16073 **WRDE\_SYNTAX** Shell syntax error, such as unbalanced parentheses or unterminated  
 16074 string.

16075 The <wordexp.h> header shall define the **size\_t** type as described in <stddef.h>.16076 The following shall be declared as functions and may also be defined as macros. Function  
16077 prototypes shall be provided.

16078 `int wordexp(const char *restrict, wordexp_t *restrict, int);`  
 16079 `void wordfree(wordexp_t *);`

16080 **APPLICATION USAGE**

16081 None.

16082 **RATIONALE**

16083 None.

16084 **FUTURE DIRECTIONS**

16085 None.

16086 **SEE ALSO**

16087 <stddef.h>

16088 XSH Section 2.6

16089 **CHANGE HISTORY**

16090 First released in Issue 4. Derived from the ISO POSIX-2 standard.

16091 **Issue 6**

16092 The **restrict** keyword is added to the prototype for *wordexp()*.

16093 The WRDE\_NOSYS constant is marked obsolescent.

16094 **Issue 7**

16095 The obsolescent WRDE\_NOSYS constant is removed.

16096 This reference page is clarified with respect to macros and symbolic constants.



16097



16098

**Vol. 2:**

16099

**System Interfaces, Issue 7**

16100

*The Open Group*

16101

*The Institute of Electrical and Electronics Engineers, Inc.*





# Introduction

The System Interfaces volume of POSIX.1-2008 describes the interfaces offered to application programs by POSIX-conformant systems.

## 1.1 Relationship to Other Formal Standards

Great care has been taken to ensure that this volume of POSIX.1-2008 is fully aligned with the following standards:

ISO C (1999)

ISO/IEC 9899:1999, Programming Languages — C, including ISO/IEC 9899:1999/Cor.1:2001(E), ISO/IEC 9899:1999/Cor.2:2004(E), and ISO/IEC 9899:1999/Cor.3.

Parts of the ISO/IEC 9899:1999 standard (hereinafter referred to as the ISO C standard) are referenced to describe requirements also mandated by this volume of POSIX.1-2008. Some functions and headers included within this volume of POSIX.1-2008 have a version in the ISO C standard; in this case CX markings are added as appropriate to show where the ISO C standard has been extended (see [Section 1.7.1](#), on page 7). Any conflict between this volume of POSIX.1-2008 and the ISO C standard is unintentional.

This volume of POSIX.1-2008 also allows, but does not require, mathematics functions to support IEEE Std 754-1985 and IEEE Std 854-1987.

## 1.2 Format of Entries

The entries in [Chapter 3](#) are based on a common format as follows. The only sections relating to conformance are the SYNOPSIS, DESCRIPTION, RETURN VALUE, and ERRORS sections.

### NAME

This section gives the name or names of the entry and briefly states its purpose.

### SYNOPSIS

This section summarizes the use of the entry being described. If it is necessary to include a header to use this function, the names of such headers are shown, for example:

```
#include <stdio.h>
```

### DESCRIPTION

This section describes the functionality of the function or header.

### RETURN VALUE

This section indicates the possible return values, if any.

If the implementation can detect errors, “successful completion” means that no error

16136 has been detected during execution of the function. If the implementation does detect  
16137 an error, the error is indicated.

16138 For functions where no errors are defined, “successful completion” means that if the  
16139 implementation checks for errors, no error has been detected. If the implementation can  
16140 detect errors, and an error is detected, the indicated return value is returned and *errno*  
16141 may be set.

## 16142 **ERRORS**

16143 This section gives the symbolic names of the error values returned by a function or  
16144 stored into a variable accessed through the symbol *errno* if an error occurs.

16145 “No errors are defined” means that error values returned by a function or stored into a  
16146 variable accessed through the symbol *errno*, if any, depend on the implementation.

## 16147 **EXAMPLES**

16148 This section is informative.

16149 This section gives examples of usage, where appropriate. In the event of conflict  
16150 between an example and a normative part of this volume of POSIX.1-2008, the  
16151 normative material is to be taken as correct.

## 16152 **APPLICATION USAGE**

16153 This section is informative.

16154 This section gives warnings and advice to application developers about the entry. In the  
16155 event of conflict between warnings and advice and a normative part of this volume of  
16156 POSIX.1-2008, the normative material is to be taken as correct.

## 16157 **RATIONALE**

16158 This section is informative.

16159 This section contains historical information concerning the contents of this volume of  
16160 POSIX.1-2008 and why features were included or discarded by the standard  
16161 developers.

## 16162 **FUTURE DIRECTIONS**

16163 This section is informative.

16164 This section provides comments which should be used as a guide to current thinking;  
16165 there is not necessarily a commitment to adopt these future directions.

## 16166 **SEE ALSO**

16167 This section is informative.

16168 This section gives references to related information.

## 16169 **CHANGE HISTORY**

16170 This section is informative.

16171 This section shows the derivation of the entry and any significant changes that have  
16172 been made to it.

16173

Chapter 2

16174

# General Information

16175

This chapter covers information that is relevant to all the functions specified in [Chapter 3](#) and XBD [Chapter 13](#) (on page 219).

16176

16177

## 2.1 Use and Implementation of Interfaces

16178

### 2.1.1 Use and Implementation of Functions

16179

Each of the following statements shall apply to all functions unless explicitly stated otherwise in the detailed descriptions that follow:

16180

16181

1. If an argument to a function has an invalid value (such as a value outside the domain of the function, or a pointer outside the address space of the program, or a null pointer), the behavior is undefined.

16182

16183

16184

2. Any function declared in a header may also be implemented as a macro defined in the header, so a function should not be declared explicitly if its header is included. Any macro definition of a function can be suppressed locally by enclosing the name of the function in parentheses, because the name is then not followed by the <left-parenthesis> that indicates expansion of a macro function name. For the same syntactic reason, it is permitted to take the address of a function even if it is also defined as a macro. The use of the C-language **#undef** construct to remove any such macro definition shall also ensure that an actual function is referred to.

16185

16186

16187

16188

16189

16190

16191

16192

3. Any invocation of a function that is implemented as a macro shall expand to code that evaluates each of its arguments exactly once, fully protected by parentheses where necessary, so it is generally safe to use arbitrary expressions as arguments.

16193

16194

16195

4. Provided that a function can be declared without reference to any type defined in a header, it is also permissible to declare the function explicitly and use it without including its associated header.

16196

16197

16198

5. If a function that accepts a variable number of arguments is not declared (explicitly or by including its associated header), the behavior is undefined.

16199

## 2.1.2 Use and Implementation of Macros

Each of the following statements shall apply to all macros unless explicitly stated otherwise:

1. Any definition of an object-like macro in a header shall expand to code that is fully protected by parentheses where necessary, so that it groups in an arbitrary expression as if it were a single identifier.
2. All object-like macros listed as expanding to integer constant expressions shall additionally be suitable for use in **#if** preprocessing directives.
3. Any definition of a function-like macro in a header shall expand to code that evaluates each of its arguments exactly once, fully protected by parentheses where necessary, so that it is generally safe to use arbitrary expressions as arguments.
4. Any definition of a function-like macro in a header can be invoked in an expression anywhere a function with a compatible return type could be called.

## 2.2 The Compilation Environment

### 2.2.1 POSIX.1 Symbols

Certain symbols in this volume of POSIX.1-2008 are defined in headers (see XBD [Chapter 13](#), on page 219). Some of those headers could also define symbols other than those defined by POSIX.1-2008, potentially conflicting with symbols used by the application. Also, POSIX.1-2008 defines symbols that are not permitted by other standards to appear in those headers without some control on the visibility of those symbols.

Symbols called “feature test macros” are used to control the visibility of symbols that might be included in a header. Implementations, future versions of this standard, and other standards may define additional feature test macros.

In the compilation of an application that **#defines** a feature test macro specified by POSIX.1-2008, no header defined by POSIX.1-2008 shall be included prior to the definition of the feature test macro. This restriction also applies to any implementation-provided header in which these feature test macros are used. If the definition of the macro does not precede the **#include**, the result is undefined.

Feature test macros shall begin with the <underscore> character (‘\_’).

#### 2.2.1.1 The `_POSIX_C_SOURCE` Feature Test Macro

A POSIX-conforming application shall ensure that the feature test macro `_POSIX_C_SOURCE` is defined before inclusion of any header.

When an application includes a header described by POSIX.1-2008, and when this feature test macro is defined to have the value 200809L:

1. All symbols required by POSIX.1-2008 to appear when the header is included shall be made visible.

2. Symbols that are explicitly permitted, but not required, by POSIX.1-2008 to appear in that header (including those in reserved name spaces) may be made visible.

3. Additional symbols not required or explicitly permitted by POSIX.1-2008 to be in that header shall not be made visible, except when enabled by another feature test macro.

Identifiers in POSIX.1-2008 may only be undefined using the **#undef** directive as described in [Section 2.1](#) (on page 471) or [Section 2.2.2](#). These **#undef** directives shall follow all **#include** directives of any header in POSIX.1-2008.

**Note:** The POSIX.1-1990 standard specified a macro called `_POSIX_SOURCE`. This has been superseded by `_POSIX_C_SOURCE`.

#### 2.2.1.2 The `_XOPEN_SOURCE` Feature Test Macro

XSI An XSI-conforming application shall ensure that the feature test macro `_XOPEN_SOURCE` is defined with the value 700 before inclusion of any header. This is needed to enable the functionality described in [Section 2.2.1.1](#) (on page 472) and to ensure that the XSI option is enabled.

Since this volume of POSIX.1-2008 is aligned with the ISO C standard, and since all functionality enabled by `_POSIX_C_SOURCE` set equal to 200809L is enabled by `_XOPEN_SOURCE` set equal to 700, there should be no need to define `_POSIX_C_SOURCE` if `_XOPEN_SOURCE` is so defined. Therefore, if `_XOPEN_SOURCE` is set equal to 700 and `_POSIX_C_SOURCE` is set equal to 200809L, the behavior is the same as if only `_XOPEN_SOURCE` is defined and set equal to 700. However, should `_POSIX_C_SOURCE` be set to a value greater than 200809L, the behavior is unspecified.

If `_XOPEN_SOURCE` is defined with the value 700 and `_POSIX_C_SOURCE` is undefined before inclusion of any header, then the header may define the `_POSIX_C_SOURCE` macro with the value 200809L.

### 2.2.2 The Name Space

XSI All identifiers in this volume of POSIX.1-2008, except *environ*, are defined in at least one of the headers, as shown in XBD [Chapter 13](#) (on page 219). When `_XOPEN_SOURCE` or `_POSIX_C_SOURCE` is defined, each header defines or declares some identifiers, potentially conflicting with identifiers used by the application. The set of identifiers visible to the application consists of precisely those identifiers from the header pages of the included headers, as well as additional identifiers reserved for the implementation. In addition, some headers may make visible identifiers from other headers as indicated on the relevant header pages.

Implementations may also add members to a structure or union without controlling the visibility of those members with a feature test macro, as long as a user-defined macro with the same name cannot interfere with the correct interpretation of the program. The identifiers reserved for use by the implementation are described below:

1. Each identifier with external linkage described in the header section is reserved for use as an identifier with external linkage if the header is included.
2. Each macro described in the header section is reserved for any use if the header is included.
3. Each identifier with file scope described in the header section is reserved for use as an identifier with file scope in the same name space if the header is included.

The prefixes `posix_`, `POSIX_`, and `_POSIX_` are reserved for use by POSIX.1-2008 and other

16278 POSIX standards. Implementations may add symbols to the headers shown in the following  
16279 table, provided the identifiers for those symbols either:

- 16280 1. Begin with the corresponding reserved prefixes in the table, or
- 16281 2. Have one of the corresponding complete names in the table, or
- 16282 3. End in the string indicated as a reserved suffix in the table and do not use the reserved  
16283 prefixes `posix_`, `POSIX_`, or `_POSIX_`, as long as the reserved suffix is in that part of the  
16284 name considered significant by the implementation.

16285 Symbols that use the reserved prefix `_POSIX_` may be made visible by implementations in any  
16286 header defined by POSIX.1-2008.

		Header	Prefix	Suffix	Complete Name
16287		<aio.h>	aio_, lio_, AIO_, LIO_		
16288		<arpa/inet.h>	inet_		
16289		<ctype.h>	to[a-z], is[a-z]		
16290		<dlfcn.h>	RTLD_		
16291		<dirent.h>	d_		
16292		<fcntl.h>	l_		
16293		<fmtmsg.h>	MM_		
16294	XSI	<fnmatch.h>	FNM_		
16295		<ftw.h>	FTW		
16296	XSI	<glob.h>	gl_, GLOB_		
16297		<grp.h>	gr_		
16298		<limits.h>		_MAX, _MIN	
16299		<math.h>	M_		
16300	XSI	<mqueue.h>	mq_, MQ_		
16301	MSG	<ndbm.h>	dbm_, DBM_		
16302	XSI	<netdb.h>	ai_, h_, n_, p_, s_		
16303		<net/if.h>	if_, IF_		
16304		<netinet/in.h>	in_, ip_, s_, sin_, INADDR_, IPPROTO_		
16305			in6_, s6_, sin6_, IPV6_		
16306	IP6	<netinet/tcp.h>	TCP_		
16307		<nl_types.h>	NL_		
16308		<poll.h>	pd_, ph_, ps_, POLL		
16309		<pthread.h>	pthread_, PTHREAD_		
16310		<pwd.h>	pw_		
16311		<regex.h>	re_, rm_, REG_		
16312		<sched.h>	sched_, SCHED_		
16313		<semaphore.h>	sem_, SEM_		
16314		<signal.h>	sa_, si_, sigev_, sival_, uc_, BUS_, CLD_, FPE_, ILL_, SA_, SEGV_, SI_, SIGEV_, ss_, sv_, SS_, TRAP_, POLL_		
16315	CX	<stropts.h>	bi_, ic_, l_, sl_, str_, FLUSH[A-Z], I_, S_, SND[A-Z]		
16316	XSI	<stdint.h>			
16317		<stdlib.h>	str[a-z]		
16318		<string.h>	str[a-z], mem[a-z], wcs[a-z]		
16319	XSI	<sys/ipc.h>	ipc_, IPC_		key, pad, seq
16320		<sys/mman.h>	shm_, MAP_, MCL_, MS_, PROT_		
16321	XSI	<sys/msg.h>	msg, MSG_[A-Z]		msg
16322	XSI	<sys/resource.h>	rlim_, ru_, PRIO_, RLIMIT_, RUSAGE_		
16323		<sys/select.h>	fd_, fds_, FD_		

		Header	Prefix	Suffix	Complete Name
16331					
16332					
16333	XSI	<sys/sem.h>	sem, SEM_		sem
16334	XSI	<sys/shm.h>	shm, SHM[A-Z], SHM_[A-Z]		
16335		<sys/socket.h>	cmsg_, if_, ifc_, ifra_, ifru_, infu_, l_, msg_, sa_, ss_, AF_, MSG_, PF_, SCM_, SHUT_, SO		
16336					
16337	XSI				
16338					
16339		<sys/stat.h>	st_		
16340		<sys/statvfs.h>	f_, ST_		
16341	XSI	<sys/time.h>	it_, tv_, ITIMER_		
16342		<sys/times.h>	tms_		
16343	XSI	<sys/uio.h>	iov_		UIO_MAXIOV
16344		<sys/un.h>	sun_		
16345		<sys/utsname.h>	uts_		
16346		<sys/wait.h>	P_, W[A-Z]		
16347	XSI	<syslog.h>	LOG_		
16348		<termios.h>	c_, B[0-9], TC		
16349	CX	<time.h>	tm_ clock_, it_, timer_, tv_, CLOCK_, TIMER_		
16350					
16351					
16352	XSI	<ulimit.h>	UL_		
16353	OB	<utime.h>	utim_		
16354	XSI	<utmpx.h>	ut_	_LVL, _PROCESS, _TIME	
16355					
16356		<wchar.h>	wcs[a-z]		
16357		<wctype.h>	is[a-z], to[a-z]		
16358		<wordexp.h>	we_, WRDE_		
16359	CX	ANY header		_t	

**Note:** The notation [0–9] indicates any digit. The notation [A–Z] indicates any uppercase letter in the portable character set. The notation [a–z] indicates any lowercase letter in the portable character set. Commas and spaces in the lists of prefixes and complete names in the above table are not part of any prefix or complete name. +

Implementations may also add symbols to the **<complex.h>** header with the following complete names or the same names suffixed with 'f' or 'l': +

16366	cerf	+	cerfc	+	cexp2	+	++
16367	cexpm1	+	clog10	+	clog1p	+	++
16368	clog2	+	clgamma	+	ctgamma	+	++



If any header in the following table is included, macros with the prefixes shown may be defined. After the last inclusion of a given header, an application may use identifiers with the corresponding prefixes for its own purpose, provided their use is preceded by a **#undef** of the corresponding macro.

Header	Prefix
<b>&lt;errno.h&gt;</b>	E[0-9], E[A-Z]
<b>&lt;fcntl.h&gt;</b>	F_, O_
<b>&lt;fenv.h&gt;</b>	FE_[A-Z]
<b>&lt;inttypes.h&gt;</b>	PRI[Xa-z], SCN[Xa-z]
<b>&lt;locale.h&gt;</b>	LC_[A-Z]
<b>&lt;math.h&gt;</b>	FP_[A-Z]
<b>&lt;netinet/in.h&gt;</b>	IMPLINK_, IN_, IP_, IPPORT_, SOCK_,
	IN6_
<b>&lt;signal.h&gt;</b>	SIG_, SIG[A-Z],
	SV_
<b>&lt;stdio.h&gt;</b>	SEEK_
<b>&lt;stropts.h&gt;</b>	M_, MUXID_R[A-Z], STR
<b>&lt;sys/resource.h&gt;</b>	RLIM_
<b>&lt;sys/socket.h&gt;</b>	CMSG_
<b>&lt;sys/stat.h&gt;</b>	S_
<b>&lt;sys/uio.h&gt;</b>	IOV_
<b>&lt;termios.h&gt;</b>	I, O, V (See below.)
<b>&lt;unistd.h&gt;</b>	SEEK_

The following are used to reserve complete names for the **<stdint.h>** header:

INT[0-9A-Za-z-]\*\_MIN  
 INT[0-9A-Za-z-]\*\_MAX  
 INT[0-9A-Za-z-]\*\_C  
 UINT[0-9A-Za-z-]\*\_MIN  
 UINT[0-9A-Za-z-]\*\_MAX  
 UINT[0-9A-Za-z-]\*\_C

**Note:** The notation [0-9] indicates any digit. The notation [A-Z] indicates any uppercase letter in the portable character set. The notation [Xa-z] indicates the character 'x' or any lowercase letter in the portable character set. The notation [0-9A-Za-z-]\* indicates zero or more occurrences of any of the following: a digit, an uppercase or lowercase letter in the portable character set, or an underscore.

The following reserved names are used as exact matches for **<termios.h>**:

CBAUD	EXTB	VDSUSP
DEFECHO	FLUSHO	VLNEXT
ECHOCTL	LOBLK	VREPRINT
ECHOKE	PENDIN	VSTATUS
ECHOPRT	SWTCH	VWERASE
EXTA	VDISCARD	

- 16411 The following identifiers are reserved regardless of the inclusion of headers:
- 16412 1. With the exception of identifiers beginning with the prefix `_POSIX_`, all identifiers that
- 16413 begin with an `<underscore>` and either an uppercase letter or another `<underscore>` are
- 16414 always reserved for any use by the implementation.
- 16415 2. All identifiers that begin with an `<underscore>` are always reserved for use as identifiers
- 16416 with file scope in both the ordinary identifier and tag name spaces.
- 16417 3. All identifiers in the table below are reserved for use as identifiers with external linkage.
- 16418 Some of these identifiers do not appear in this volume of POSIX.1-2008, but are reserved for
- 16419 future use by the ISO C standard.
- 16420 4. All functions and external identifiers defined in XBD [Chapter 13](#) (on page 219) are reserved
- 16421 for use as identifiers with external linkage.
- 16422 5. All the identifiers defined in this volume of POSIX.1-2008 that have external linkage are
- 16423 always reserved for use as identifiers with external linkage.
- 16424 **Note:** The notation `[a-z]` indicates any lowercase letter in the portable character set. The notation `' * '`
- 16425 indicates any combination of digits, letters in the portable character set, or `<underscore>`.
- 16426 No other identifiers are reserved.

16427	_Exit	catan	clogf	exit	fopen
16428	abort	catanf	clogl	exp	fprintf
16429	abs	catanh	conj	exp2	fputc
16430	acos	catanhf	conjf	exp2f	fputs
16431	acosh	catanhl	conjl	exp2l	fputwc
16432	acoshf	catanl	copysign	expf	fputws
16433	acoshf	cbrt	copysignf	expl	fread
16434	acoshl	cbrtf	copysignl	expm1	free
16435	acosl	cbrtl	cos	expm1f	freopen
16436	asctime	ccos	cosf	expm1l	frexp
16437	asin	ccosf	cosh	fabs	frexpf
16438	asinf	ccosh	coshf	fabsf	frexpl
16439	asinh	ccoshf	coshl	fabsl	fscanf
16440	asinhf	ccoshl	cosl	fclose	fseek
16441	asinhf	ccosl	cpow	fdim	fsetpos
16442	asinl	ceil	cpowf	fdimf	ftell
16443	atan	ceilf	cpowl	fdiml	fwide
16444	atan2	ceilf	cproj	feclearexcept	fwprintf
16445	atan2f	cerf	cprojf	fegetenv	fwrite
16446	atan2l	cerfc	cprojl	fegetexceptflag	fwscanf
16447	atanf	cerfcf	creal	fegetround	getc
16448	atanh	cerfcl	crealf	feholdexcept	getchar
16449	atanhf	cerff	creall	feof	getenv
16450	atanhl	cerfl	csin	feraiseexcept	gets
16451	atanl	cexpm1	csinf	ferror	getwc
16452	atexit	cexpm1f	csinh	fesetenv	getwchar
16453	atof	cexpm1l	csinhf	fesetexceptflag	gmtime
16454	atoi	cexp	csinhf	fesetround	hypot
16455	atol	cexp2	csinl	fetestexcept	hypotf
16456	atoll	cexp2f	csqrt	feupdateenv	hypotl
16457	bsearch	cexp2l	csqrtf	fflush	ilogb
16458	btowc	cexpf	csqrtl	fgetc	ilogbf
16459	cabs	cexpl	ctan	fgetpos	ilogbl
16460	cabsf	cimag	ctanf	fgets	imaxabs
16461	cabsl	cimagf	ctanh	fgetwc	imaxdiv
16462	cacos	cimagl	ctanhf	fgetws	is[a-z]*
16463	cacosf	clearerr	ctanhl	floor	labs
16464	cacosh	clgamma	ctanl	floorf	ldexp
16465	cacoshf	clgammaf	ctgamma	floorl	ldexpf
16466	cacoshl	clgammaf	ctgammaf	fma	ldexpl
16467	cacosl	clock	ctgammaf	fmaf	ldiv
16468	calloc	clog	ctime	fmal	lgamma
16469	carg	clog10	difftime	fmax	lgammaf
16470	cargf	clog10f	div	fmaxf	lgammal
16471	cargl	clog10l	erf	fmaxl	llabs
16472	casin	clog1p	erfc	fmin	lldiv
16473	casinf	clog1pf	erfcf	fminf	llrint
16474	casinh	clog1pl	erfcl	fminl	llrintf
16475	casinhf	clog2	erff	fmod	llrintl
16476	casinhl	clog2f	erfl	fmodf	llround
16477	casinl	clog2l	errno	fmodl	llroundf

16478	llroundl	mbtowc	remainderf	sprintf	vfwscanf
16479	localeconv	mem[a-z]*	remainderl	sqrt	vprintf
16480	localtime	mktime	remove	sqrtd	vscanf
16481	log	modf	remquo	sqrtd	vsnprintf
16482	log10	modff	remquof	srand	vsprintf
16483	log10f	modfl	remquol	sscanf	vsscanf
16484	log10l	nan	rename	str[a-z]*	vswprintf
16485	log1p	nanf	rewind	swprintf	vswscanf
16486	log1pf	nanl	rint	swscanf	vwprintf
16487	log1pl	nearbyint	rintf	system	vwscanf
16488	log2	nearbyintf	rintl	tan	wcrtomb
16489	log2f	nearbyintl	round	tanf	wcs[a-z]*
16490	log2l	nextafter	roundf	tanh	wctob
16491	logb	nextafterf	roundl	tanhf	wctomb
16492	logbf	nextafterl	scalbln	tanhl	wctrans
16493	logbl	nexttoward	scalblnf	tanl	wctype
16494	logf	nexttowardf	scalblnl	tgamma	wmemchr
16495	logl	nexttowardl	scalbn	tgammaf	wmemcmp
16496	longjmp	perror	scalbnf	tgammal	wmemcpy
16497	lrint	pow	scalbnl	time	wmemmove
16498	lrintf	powf	scanf	tmpfile	wmemset
16499	lrintl	powl	setbuf	tmpnam	wprintf
16500	lround	printf	setjmp	to[a-z]*	wscanf
16501	lroundf	putc	setlocale	trunc	
16502	lroundl	putchar	setvbuf	truncf	
16503	malloc	puts	signal	truncl	
16504	math_errhandling	putwc	sin	ungetc	
16505	mblen	putwchar	sinf	ungetwc	
16506	mbrlen	qsort	sinh	va_copy	
16507	mbrtowc	raise	sinhf	va_end	
16508	mbsinit	rand	sinhl	vfprintf	
16509	mbsrtowcs	realloc	sinl	vfscanf	
16510	mbstowcs	remainder	snprintf	vfwprintf	

**Note:** The notation [a-z] indicates any lowercase letter in the portable character set. The notation ' \* ' indicates any sequence of zero or more characters that are valid in identifiers with external linkage.

Applications shall not declare or define identifiers with the same name as an identifier reserved in the same context. Since macro names are replaced whenever found, independent of scope and name space, macro names matching any of the reserved identifier names shall not be defined by an application if any associated header is included.

Except that the effect of each inclusion of **<assert.h>** depends on the definition of **NDEBUG**, headers may be included in any order, and each may be included more than once in a given scope, with no difference in effect from that of being included only once.

If used, the application shall ensure that a header is included outside of any external declaration or definition, and it shall be first included before the first reference to any type or macro it defines, or to any function or object it declares. However, if an identifier is declared or defined in more than one header, the second and subsequent associated headers may be included after the initial reference to the identifier. Prior to the inclusion of a header, the application shall not define any macros with names lexically identical to symbols defined by that header.

## 2.3 Error Numbers

Most functions can provide an error number. The means by which each function provides its error numbers is specified in its description.

Some functions provide the error number in a variable accessed through the symbol *errno*, defined by including the `<errno.h>` header. The value of *errno* should only be examined when it is indicated to be valid by a function's return value. No function in this volume of POSIX.1-2008 shall set *errno* to zero. For each thread of a process, the value of *errno* shall not be affected by function calls or assignments to *errno* by other threads.

Some functions return an error number directly as the function value. These functions return a value of zero to indicate success.

If more than one error occurs in processing a function call, any one of the possible errors may be returned, as the order of detection is undefined.

Implementations may support additional errors not included in this list, may generate errors included in this list under circumstances other than those described here, or may contain extensions or limitations that prevent some errors from occurring.

The ERRORS section on each reference page specifies which error conditions shall be detected by all implementations ("shall fail") and which may be optionally detected by an implementation ("may fail"). If no error condition is detected, the action requested shall be successful. If an error condition is detected, the action requested may have been partially performed, unless otherwise stated.

Implementations may generate error numbers listed here under circumstances other than those described, if and only if all those error conditions can always be treated identically to the error conditions as described in this volume of POSIX.1-2008. Implementations shall not generate a different error number from one required by this volume of POSIX.1-2008 for an error condition described in this volume of POSIX.1-2008, but may generate additional errors unless explicitly disallowed for a particular function.

Each implementation shall document, in the conformance document, situations in which each of the optional conditions defined in POSIX.1-2008 is detected. The conformance document may also contain statements that one or more of the optional error conditions are not detected.

Certain threads-related functions are not allowed to return an error code of [EINTR]. Where this applies it is stated in the ERRORS section on the individual function pages.

The following macro names identify the possible error numbers, in the context of the functions specifically defined in this volume of POSIX.1-2008; these general descriptions are more precisely defined in the ERRORS sections of the functions that return them. Only these macro names should be used in programs, since the actual value of the error number is unspecified. All values listed in this section shall be unique, except as noted below. The values for all these macros shall be found in the `<errno.h>` header defined in the Base Definitions volume of POSIX.1-2008. The actual values are unspecified by this volume of POSIX.1-2008.

[E2BIG]

Argument list too long. The sum of the number of bytes used by the new process image's argument list and environment list is greater than the system-imposed limit of {ARG\_MAX} bytes.

or:

Lack of space in an output buffer.

or:

16572		Argument is greater than the system-imposed maximum.
16573		[EACCES]
16574		Permission denied. An attempt was made to access a file in a way forbidden by its file
16575		access permissions.
16576		[EADDRINUSE]
16577		Address in use. The specified address is in use.
16578		[EADDRNOTAVAIL]
16579		Address not available. The specified address is not available from the local system.
16580		[EAFNOSUPPORT]
16581		Address family not supported. The implementation does not support the specified address
16582		family, or the specified address is not a valid address for the address family of the specified
16583		socket.
16584		[EAGAIN]
16585		Resource temporarily unavailable. This is a temporary condition and later calls to the same
16586		routine may complete normally.
16587		[EALREADY]
16588		Connection already in progress. A connection request is already in progress for the specified
16589		socket.
16590		[EBADF]
16591		Bad file descriptor. A file descriptor argument is out of range, refers to no open file, or a
16592		read (write) request is made to a file that is only open for writing (reading).
16593		[EBADMSG]
16594	OB XSR	Bad message. During a <i>read()</i> , <i>getmsg()</i> , <i>getpmsg()</i> , or <i>ioctl()</i> I_RECVFD request to a
16595		STREAMS device, a message arrived at the head of the STREAM that is inappropriate for
16596		the function receiving the message.
16597		<i>read()</i> Message waiting to be read on a STREAM is not a data message.
16598		<i>getmsg()</i> or <i>getpmsg()</i>
16599		A file descriptor was received instead of a control message.
16600		<i>ioctl()</i> Control or data information was received instead of a file descriptor when
16601		I_RECVFD was specified.
16602		or:
16603		Bad Message. The implementation has detected a corrupted message.
16604		[EBUSY]
16605		Resource busy. An attempt was made to make use of a system resource that is not currently
16606		available, as it is being used by another process in a manner that would have conflicted
16607		with the request being made by this process.
16608		[ECANCELED]
16609		Operation canceled. The associated asynchronous operation was canceled before
16610		completion.
16611		[ECHILD]
16612		No child process. A <i>wait()</i> , <i>waitid()</i> , or <i>waitpid()</i> function was executed by a process that
16613		had no existing or unwaited-for child process.

16614	[ECONNABORTED]
16615	Connection aborted. The connection has been aborted.
16616	[ECONNREFUSED]
16617	Connection refused. An attempt to connect to a socket was refused because there was no
16618	process listening or because the queue of connection requests was full and the underlying
16619	protocol does not support retransmissions.
16620	[ECONNRESET]
16621	Connection reset. The connection was forcibly closed by the peer.
16622	[EDEADLK]
16623	Resource deadlock would occur. An attempt was made to lock a system resource that would
16624	have resulted in a deadlock situation.
16625	[EDESTADDRREQ]
16626	Destination address required. No bind address was established.
16627	[EDOM]
16628	Domain error. An input argument is outside the defined domain of the mathematical
16629	function (defined in the ISO C standard).
16630	[EDQUOT]
16631	Reserved.
16632	[EEXIST]
16633	File exists. An existing file was mentioned in an inappropriate context; for example, as a
16634	new link name in the <i>link()</i> function.
16635	[EFAULT]
16636	Bad address. The system detected an invalid address in attempting to use an argument of a
16637	call. The reliable detection of this error cannot be guaranteed, and when not detected may
16638	result in the generation of a signal, indicating an address violation, which is sent to the
16639	process.
16640	[EFBIG]
16641	File too large. The size of a file would exceed the maximum file size of an implementation
16642	or offset maximum established in the corresponding file description.
16643	[EHOSTUNREACH]
16644	Host is unreachable. The destination host cannot be reached (probably because the host is
16645	down or a remote router cannot reach it).
16646	[EIDRM]
16647	Identifier removed. Returned during XSI interprocess communication if an identifier has
16648	been removed from the system.
16649	[EILSEQ]
16650	Illegal byte sequence. A wide-character code has been detected that does not correspond to
16651	a valid character, or a byte sequence does not form a valid wide-character code (defined in
16652	the ISO C standard).
16653	[EINPROGRESS]
16654	Operation in progress. This code is used to indicate that an asynchronous operation has not
16655	yet completed.
16656	or:
16657	O_NONBLOCK is set for the socket file descriptor and the connection cannot be
16658	immediately established.



16659	[EINTR]
16660	Interrupted function call. An asynchronous signal was caught by the process during the
16661	execution of an interruptible function. If the signal handler performs a normal return, the
16662	interrupted function call may return this condition (see the Base Definitions volume of
16663	POSIX.1-2008, <signal.h>).
16664	[EINVAL]
16665	Invalid argument. Some invalid argument was supplied; for example, specifying an
16666	undefined signal in a <i>signal()</i> function or a <i>kill()</i> function.
16667	[EIO]
16668	Input/output error. Some physical input or output error has occurred. This error may be
16669	reported on a subsequent operation on the same file descriptor. Any other error-causing
16670	operation on the same file descriptor may cause the [EIO] error indication to be lost.
16671	[EISCONN]
16672	Socket is connected. The specified socket is already connected.
16673	[EISDIR]
16674	Is a directory. An attempt was made to open a directory with write mode specified.
16675	[ELOOP]
16676	Symbolic link loop. A loop exists in symbolic links encountered during pathname
16677	resolution. This error may also be returned if more than {SYMLOOP_MAX} symbolic links
16678	are encountered during pathname resolution.
16679	[EMFILE]
16680	File descriptor value too large or too many open streams. An attempt was made to open a
16681	XSI file descriptor with a value greater than or equal to {OPEN_MAX}, or greater than or equal
16682	to the soft limit RLIMIT_NOFILE for the process (if smaller than {OPEN_MAX}); or an
16683	attempt was made to open more than the maximum number of streams allowed in the
16684	process.
16685	[EMLINK]
16686	Too many links. An attempt was made to have the link count of a single file exceed
16687	{LINK_MAX}.
16688	[EMSGSIZE]
16689	Message too large. A message sent on a transport provider was larger than an internal
16690	message buffer or some other network limit.
16691	or:
16692	Inappropriate message buffer length.
16693	[EMULTIHOP]
16694	Reserved.
16695	[ENAMETOOLONG]
16696	Filename too long. The length of a pathname exceeds {PATH_MAX} and the
16697	implementation considers this to be an error, or a pathname component is longer than
16698	{NAME_MAX}. This error may also occur when pathname substitution, as a result of
16699	encountering a symbolic link during pathname resolution, results in a pathname string the
16700	size of which exceeds {PATH_MAX}.
16701	[ENETDOWN]
16702	Network is down. The local network interface used to reach the destination is down.



16703		[ENETRESET]
16704		The connection was aborted by the network.
16705		[ENETUNREACH]
16706		Network unreachable. No route to the network is present.
16707		[ENFILE]
16708		Too many files open in system. Too many files are currently open in the system. The system
16709		has reached its predefined limit for simultaneously open files and temporarily cannot
16710		accept requests to open another one.
16711		[ENOBUFFS]
16712		No buffer space available. Insufficient buffer resources were available in the system to
16713		perform the socket operation.
16714	OB XSR	[ENODATA]
16715		No message available. No message is available on the STREAM head read queue.
16716		[ENODEV]
16717		No such device. An attempt was made to apply an inappropriate function to a device; for
16718		example, trying to read a write-only device such as a printer.
16719		[ENOENT]
16720		No such file or directory. A component of a specified pathname does not exist, or the
16721		pathname is an empty string.
16722		[ENOEXEC]
16723		Executable file format error. A request is made to execute a file that, although it has
16724		appropriate privileges, is not in the format required by the implementation for executable
16725		files.
16726		[ENOLCK]
16727		No locks available. A system-imposed limit on the number of simultaneous file and record
16728		locks has been reached and no more are currently available.
16729		[ENOLINK]
16730		Reserved.
16731		[ENOMEM]
16732		Not enough space. The new process image requires more memory than is allowed by the
16733		hardware or system-imposed memory management constraints.
16734		[ENOMSG]
16735		No message of the desired type. The message queue does not contain a message of the
16736		required type during XSI interprocess communication.
16737		[ENOPROTOPT]
16738		Protocol not available. The protocol option specified to <i>setsockopt()</i> is not supported by the
16739		implementation.
16740		[ENOSPC]
16741		No space left on a device. During the <i>write()</i> function on a regular file or when extending a
16742		directory, there is no free space left on the device.
16743	OB XSR	[ENOSR]
16744		No STREAM resources. Insufficient STREAMS memory resources are available to perform a
16745		STREAMS-related function. This is a temporary condition; it may be recovered from if other
16746		processes release resources.

16747	OB XSR	[ENOSTR]
16748		Not a STREAM. A STREAM function was attempted on a file descriptor that was not
16749		associated with a STREAMS device.
16750		[ENOSYS]
16751		Functionality not supported. An attempt was made to use optional functionality that is not
16752		supported in this implementation.
16753		[ENOTCONN]
16754		Socket not connected. The socket is not connected.
16755		[ENOTDIR]
16756		Not a directory. A component of the specified pathname exists, but it is not a directory,
16757		when a directory was expected; or an attempt was made to create a non-directory file, and
16758		the specified pathname contains at least one non- <code>&lt;slash&gt;</code> character and ends with one or
16759		more trailing <code>&lt;slash&gt;</code> characters.
16760		[ENOTEMPTY]
16761		Directory not empty. A directory other than an empty directory was supplied when an
16762		empty directory was expected.
16763		[ENOTRECOVERABLE]
16764		State not recoverable. The state protected by a robust mutex is not recoverable.
16765		[ENOTSOCK]
16766		Not a socket. The file descriptor does not refer to a socket.
16767		[ENOTSUP]
16768		Not supported. The implementation does not support the requested feature or value.
16769		[ENOTTY]
16770		Inappropriate I/O control operation. A control function has been attempted for a file or
16771		special file for which the operation is inappropriate.
16772		[ENXIO]
16773		No such device or address. Input or output on a special file refers to a device that does not
16774		exist, or makes a request beyond the capabilities of the device. It may also occur when, for
16775		example, a tape drive is not on-line.
16776		[EOPNOTSUPP]
16777		Operation not supported on socket. The type of socket (address family or protocol) does not
16778		support the requested operation. A conforming implementation may assign the same values
16779		for [EOPNOTSUPP] and [ENOTSUP].
16780		[EOVERFLOW]
16781		Value too large to be stored in data type. An operation was attempted which would
16782		generate a value that is outside the range of values that can be represented in the relevant
16783		data type or that are allowed for a given data item.
16784		[EOWNERDEAD]
16785		Previous owner died. The owner of a robust mutex terminated while holding the mutex
16786		lock.
16787		[EPERM]
16788		Operation not permitted. An attempt was made to perform an operation limited to
16789		processes with appropriate privileges or to the owner of a file or other resource.
16790		[EPIPE]
16791		Broken pipe. A write was attempted on a socket, pipe, or FIFO for which there is no process
16792		to read the data.

16793	[EPROTO]
16794	Protocol error. Some protocol error occurred. This error is device-specific, but is generally
16795	not related to a hardware failure.
16796	[EPROTONOSUPPORT]
16797	Protocol not supported. The protocol is not supported by the address family, or the protocol
16798	is not supported by the implementation.
16799	[EPROTOTYPE]
16800	Protocol wrong type for socket. The socket type is not supported by the protocol.
16801	[ERANGE]
16802	Result too large or too small. The result of the function is too large (overflow) or too small
16803	(underflow) to be represented in the available space (defined in the ISO C standard).
16804	[EROFS]
16805	Read-only file system. An attempt was made to modify a file or directory on a file system
16806	that is read-only.
16807	[ESPIPE]
16808	Invalid seek. An attempt was made to access the file offset associated with a pipe or FIFO.
16809	[ESRCH]
16810	No such process. No process can be found corresponding to that specified by the given
16811	process ID.
16812	[ESTALE]
16813	Reserved.
16814	OB XSR [ETIME]
16815	STREAM <i>ioctl()</i> timeout. The timer set for a STREAMS <i>ioctl()</i> call has expired. The cause of
16816	this error is device-specific and could indicate either a hardware or software failure, or a
16817	timeout value that is too short for the specific operation. The status of the <i>ioctl()</i> operation is
16818	unspecified.
16819	[ETIMEDOUT]
16820	Connection timed out. The connection to a remote machine has timed out. If the connection
16821	timed out during execution of the function that reported this error (as opposed to timing
16822	out prior to the function being called), it is unspecified whether the function has completed
16823	some or all of the documented behavior associated with a successful completion of the
16824	function.
16825	or:
16826	Operation timed out. The time limit associated with the operation was exceeded before the
16827	operation completed.
16828	[ETXTBSY]
16829	Text file busy. An attempt was made to execute a pure-procedure program that is currently
16830	open for writing, or an attempt has been made to open for writing a pure-procedure
16831	program that is being executed.
16832	[EWOULDBLOCK]
16833	Operation would block. An operation on a socket marked as non-blocking has encountered
16834	a situation such as no data available that otherwise would have caused the function to
16835	suspend execution.
16836	A conforming implementation may assign the same values for [EWOULDBLOCK] and
16837	[EAGAIN].

[EXDEV]

Improper link. A link to a file on another file system was attempted.

### 2.3.1 Additional Error Numbers

Additional implementation-defined error numbers may be defined in `<errno.h>`.

## 2.4 Signal Concepts

### 2.4.1 Signal Generation and Delivery

A signal is said to be “generated” for (or sent to) a process or thread when the event that causes the signal first occurs. Examples of such events include detection of hardware faults, timer expiration, signals generated via the **sigevent** structure and terminal activity, as well as invocations of the `kill()` and `sigqueue()` functions. In some circumstances, the same event generates signals for multiple processes.

At the time of generation, a determination shall be made whether the signal has been generated for the process or for a specific thread within the process. Signals which are generated by some action attributable to a particular thread, such as a hardware fault, shall be generated for the thread that caused the signal to be generated. Signals that are generated in association with a process ID or process group ID or an asynchronous event, such as terminal activity, shall be generated for the process.

Each process has an action to be taken in response to each signal defined by the system (see [Section 2.4.3](#)). A signal is said to be “delivered” to a process when the appropriate action for the process and signal is taken. A signal is said to be “accepted” by a process when the signal is selected and returned by one of the `sigwait()` functions.

During the time between the generation of a signal and its delivery or acceptance, the signal is said to be “pending”. Ordinarily, this interval cannot be detected by an application. However, a signal can be “blocked” from delivery to a thread. If the action associated with a blocked signal is anything other than to ignore the signal, and if that signal is generated for the thread, the signal shall remain pending until it is unblocked, it is accepted when it is selected and returned by a call to the `sigwait()` function, or the action associated with it is set to ignore the signal. Signals generated for the process shall be delivered to exactly one of those threads within the process which is in a call to a `sigwait()` function selecting that signal or has not blocked delivery of the signal. If there are no threads in a call to a `sigwait()` function selecting that signal, and if all threads within the process block delivery of the signal, the signal shall remain pending on the process until a thread calls a `sigwait()` function selecting that signal, a thread unblocks delivery of the signal, or the action associated with the signal is set to ignore the signal. If the action associated with a blocked signal is to ignore the signal and if that signal is generated for the process, it is unspecified whether the signal is discarded immediately upon generation or remains pending.

Each thread has a “signal mask” that defines the set of signals currently blocked from delivery to it. The signal mask for a thread shall be initialized from that of its parent or creating thread, or from the corresponding thread in the parent process if the thread was created as the result of a call to `fork()`. The `pthread_sigmask()`, `sigaction()`, `sigprocmask()`, and `sigsuspend()` functions control the manipulation of the signal mask.

The determination of which action is taken in response to a signal is made at the time the signal is delivered, allowing for any changes since the time of generation. This determination is independent of the means by which the signal was originally generated. If a subsequent occurrence of a pending signal is generated, it is implementation-defined as to whether the signal is delivered or accepted more than once in circumstances other than those in which queuing is required. The order in which multiple, simultaneously pending signals outside the range SIGRTMIN to SIGRTMAX are delivered to or accepted by a process is unspecified.

When any stop signal (SIGSTOP, SIGTSTP, SIGTTIN, SIGTTOU) is generated for a process or thread, all pending SIGCONT signals for that process or any of the threads within that process shall be discarded. Conversely, when SIGCONT is generated for a process or thread, all pending stop signals for that process or any of the threads within that process shall be discarded. When SIGCONT is generated for a process that is stopped, the process shall be continued, even if the SIGCONT signal is ignored by the process or is blocked by all threads within the process and there are no threads in a call to a *sigwait()* function selecting SIGCONT. If SIGCONT is blocked by all threads within the process, there are no threads in a call to a *sigwait()* function selecting SIGCONT, and SIGCONT is not ignored by the process, the SIGCONT signal shall remain pending on the process until it is either unblocked by a thread or a thread calls a *sigwait()* function selecting SIGCONT, or a stop signal is generated for the process or any of the threads within the process.

An implementation shall document any condition not specified by this volume of POSIX.1-2008 under which the implementation generates signals.

## 2.4.2 Realtime Signal Generation and Delivery

This section describes functionality to support realtime signal generation and delivery.

Some signal-generating functions, such as high-resolution timer expiration, asynchronous I/O completion, interprocess message arrival, and the *sigqueue()* function, support the specification of an application-defined value, either explicitly as a parameter to the function or in a **sigevent** structure parameter. The **sigevent** structure is defined in **<signal.h>** and contains at least the following members:

Member Type	Member Name	Description
<b>int</b>	<i>sigev_notify</i>	Notification type.
<b>int</b>	<i>sigev_signo</i>	Signal number.
<b>union signal</b>	<i>sigev_value</i>	Signal value.
<b>void(*) (union signal)</b>	<i>sigev_notify_function</i>	Notification function.
<b>(pthread_attr_t*)</b>	<i>sigev_notify_attributes</i>	Notification attributes.

The *sigev\_notify* member specifies the notification mechanism to use when an asynchronous event occurs. This volume of POSIX.1-2008 defines the following values for the *sigev\_notify* member:

**SIGEV\_NONE** No asynchronous notification shall be delivered when the event of interest occurs.

**SIGEV\_SIGNAL** The signal specified in *sigev\_signo* shall be generated for the process when the event of interest occurs. If the implementation supports the Realtime Signals Extension option and if the SA\_SIGINFO flag is set for that signal number, then the signal shall be queued to the process and the value specified in *sigev\_value* shall be the *si\_value* component of the generated signal. If SA\_SIGINFO is not set for that signal number, it is unspecified whether the signal is queued and what value, if any, is sent.

SIGEV\_THREAD A notification function shall be called to perform notification.

An implementation may define additional notification mechanisms.

The *sigeval\_sigeno* member specifies the signal to be generated. The *sigeval\_value* member is the application-defined value to be passed to the signal-catching function at the time of the signal delivery or to be returned at signal acceptance as the *si\_value* member of the **siginfo\_t** structure.

The **sigval** union is defined in **<signal.h>** and contains at least the following members:

Member Type	Member Name	Description
<b>int</b>	<i>sival_int</i>	Integer signal value.
<b>void*</b>	<i>sival_ptr</i>	Pointer signal value.

The *sival\_int* member shall be used when the application-defined value is of type **int**; the *sival\_ptr* member shall be used when the application-defined value is a pointer.

When a signal is generated by the *sigqueue()* function or any signal-generating function that supports the specification of an application-defined value, the signal shall be marked pending and, if the SA\_SIGINFO flag is set for that signal, the signal shall be queued to the process along with the application-specified signal value. Multiple occurrences of signals so generated are queued in FIFO order. It is unspecified whether signals so generated are queued when the SA\_SIGINFO flag is not set for that signal.

Signals generated by the *kill()* function or other events that cause signals to occur, such as detection of hardware faults, *alarm()* timer expiration, or terminal activity, and for which the implementation does not support queuing, shall have no effect on signals already queued for the same signal number.

When multiple unblocked signals, all in the range SIGRTMIN to SIGRTMAX, are pending, the behavior shall be as if the implementation delivers the pending unblocked signal with the lowest signal number within that range. No other ordering of signal delivery is specified.

If, when a pending signal is delivered, there are additional signals queued to that signal number, the signal shall remain pending. Otherwise, the pending indication shall be reset.

Multi-threaded programs can use an alternate event notification mechanism. When a notification is processed, and the *sigeval\_notify* member of the **sigevent** structure has the value SIGEV\_THREAD, the function *sigeval\_notify\_function* is called with parameter *sigeval\_value*.

The function shall be executed in an environment as if it were the *start\_routine* for a newly created thread with thread attributes specified by *sigeval\_notify\_attributes*. If *sigeval\_notify\_attributes* is NULL, the behavior shall be as if the thread were created with the *detachstate* attribute set to PTHREAD\_CREATE\_DETACHED. Supplying an attributes structure with a *detachstate* attribute of PTHREAD\_CREATE\_JOINABLE results in undefined behavior. The signal mask of this thread is implementation-defined.

### 2.4.3 Signal Actions

There are three types of action that can be associated with a signal: SIG\_DFL, SIG\_IGN, or a pointer to a function. Initially, all signals shall be set to SIG\_DFL or SIG\_IGN prior to entry of the *main()* routine (see the *exec* functions). The actions prescribed by these values are as follows.



**SIG\_DFL**

Signal-specific default action.

The default actions for the signals defined in this volume of POSIX.1-2008 are specified under **<signal.h>**. The default actions for the realtime signals in the range SIGRTMIN to SIGRTMAX shall be to terminate the process abnormally.

If the default action is to terminate the process abnormally, the process is terminated as if by a call to `_exit()`, except that the status made available to `wait()`, `waitid()`, and `waitpid()` indicates abnormal termination by the signal. If the default action is to terminate the process abnormally with additional actions, implementation-defined abnormal termination actions, such as creation of a core file, may also occur.

If the default action is to stop the process, the execution of that process is temporarily suspended. When a process stops, a SIGCHLD signal shall be generated for its parent process, unless the parent process has set the SA\_NOCLDSTOP flag. While a process is stopped, any additional signals that are sent to the process shall not be delivered until the process is continued, except SIGKILL which always terminates the receiving process. A process that is a member of an orphaned process group shall not be allowed to stop in response to the SIGTSTP, SIGTTIN, or SIGTTOU signals. In cases where delivery of one of these signals would stop such a process, the signal shall be discarded.

If the default action is to ignore the signal, delivery of the signal shall have no effect on the process.

Setting a signal action to SIG\_DFL for a signal that is pending, and whose default action is to ignore the signal (for example, SIGCHLD), shall cause the pending signal to be discarded, whether or not it is blocked. Any queued values pending shall be discarded and the resources used to queue them shall be released and returned to the system for other use.

The default action for SIGCONT is to resume execution at the point where the process was stopped, after first handling any pending unblocked signals.

When a stopped process is continued, a SIGCHLD signal may be generated for its parent process, unless the parent process has set the SA\_NOCLDSTOP flag.

**SIG\_IGN**

Ignore signal.

Delivery of the signal shall have no effect on the process. The behavior of a process is undefined after it ignores a SIGFPE, SIGILL, SIGSEGV, or SIGBUS signal that was not generated by `kill()`, `sigqueue()`, or `raise()`.

The system shall not allow the action for the signals SIGKILL or SIGSTOP to be set to SIG\_IGN.

Setting a signal action to SIG\_IGN for a signal that is pending shall cause the pending signal to be discarded, whether or not it is blocked.

If a process sets the action for the SIGCHLD signal to SIG\_IGN, the behavior is unspecified, except as specified under “Consequences of Process Termination” in the description of the `_Exit()` function (see XSH `_Exit()`, on page 553).

Any queued values pending shall be discarded and the resources used to queue them shall be released and made available to queue other signals.

**Pointer to a Function**

Catch signal.

On delivery of the signal, the receiving process is to execute the signal-catching function at the specified address. After returning from the signal-catching function, the receiving process shall resume execution at the point at which it was interrupted.

If the SA\_SIGINFO flag for the signal is cleared, the signal-catching function shall be entered as a C-language function call as follows:

```
void func(int signo);
```

If the SA\_SIGINFO flag for the signal is set, the signal-catching function shall be entered as a C-language function call as follows:

```
void func(int signo, siginfo_t *info, void *context);
```

where *func* is the specified signal-catching function, *signo* is the signal number of the signal being delivered, and *info* is a pointer to a **siginfo\_t** structure defined in **<signal.h>** containing at least the following members:

Member Type	Member Name	Description
<b>int</b>	<i>si_signo</i>	Signal number.
<b>int</b>	<i>si_code</i>	Cause of the signal.
<b>pid_t</b>	<i>si_pid</i>	Sending process ID.
<b>uid_t</b>	<i>si_uid</i>	Real user ID of sending process.
<b>void *</b>	<i>si_addr</i>	Address of faulting instruction.
<b>int</b>	<i>si_status</i>	Exit value or signal.
<b>union signal</b>	<i>si_value</i>	Signal value.

The *si\_signo* member shall contain the signal number. This shall be the same as the *signo* parameter. The *si\_code* member shall contain a code identifying the cause of the signal. The following non-signal-specific values are defined for *si\_code*:

**SI\_USER**      The signal was sent by the *kill()* function. The implementation may set *si\_code* to **SI\_USER** if the signal was sent by the *raise()* or *abort()* functions or any similar functions provided as implementation extensions.

**SI\_QUEUE**      The signal was sent by the *sigqueue()* function.

**SI\_TIMER**      The signal was generated by the expiration of a timer set by *timer\_settime()*.

**SI\_ASYNCIO**    The signal was generated by the completion of an asynchronous I/O request.

**MSG**      **SI\_MSGQ**      The signal was generated by the arrival of a message on an empty message queue.

Signal-specific values for *si\_code* are also defined, as described in XBD **<signal.h>**.

If the signal was not generated by one of the functions or events listed above, *si\_code* shall be set either to one of the signal-specific values described in XBD **<signal.h>**, or to an implementation-defined value that is not equal to any of the values defined above.

**XSI**      If *si\_code* is **SI\_USER** or **SI\_QUEUE**, or any value less than or equal to 0, then the signal was generated by a process and *si\_pid* and *si\_uid* shall be set to the process ID and the real user ID of the sender, respectively.

In addition, *si\_addr*, *si\_pid*, *si\_status*, and *si\_uid* shall be set for certain signal-specific values of *si\_code*, as described in XBD **<signal.h>**.

If *si\_code* is one of **SI\_QUEUE**, **SI\_TIMER**, **SI\_ASYNCIO**, or **SI\_MSGQ**, then *si\_value* shall



17048 contain the application-specified signal value. Otherwise, the contents of *si\_value* are undefined.

17049 The behavior of a process is undefined after it returns normally from a signal-catching function  
 17050 for a SIGBUS, SIGFPE, SIGILL, or SIGSEGV signal that was not generated by *kill()*, *sigqueue()*,  
 17051 or *raise()*.

17052 The system shall not allow a process to catch the signals SIGKILL and SIGSTOP.

17053 If a process establishes a signal-catching function for the SIGCHLD signal while it has a  
 17054 terminated child process for which it has not waited, it is unspecified whether a SIGCHLD  
 17055 signal is generated to indicate that child process.

17056 If the process is multi-threaded, or if the process is single-threaded and a signal handler is  
 17057 executed other than as the result of:

- 17058 • The process calling *abort()*, *raise()*, *kill()*, *pthread\_kill()*, or *sigqueue()* to generate a signal  
 17059 that is not blocked
- 17060 • A pending signal being unblocked and being delivered before the call that unblocked it  
 17061 returns

17062 the behavior is undefined if the signal handler refers to any object other than *errno* with static  
 17063 storage duration other than by assigning a value to an object declared as **volatile sig\_atomic\_t**,  
 17064 or if the signal handler calls any function defined in this standard other than one of the functions  
 17065 listed in the following table.

17066 The following table defines a set of functions that shall be async-signal-safe. Therefore, |  
 17067 applications can call them, without restriction, from signal-catching functions. Note that, |  
 17068 although there is no restriction on the calls themselves, for certain functions there are restrictions |  
 17069 on subsequent behavior after the function is called from a signal-catching function (see XSH |  
 17070 *longjmp()*, on page 1275).

17071	<code>_Exit()</code>	<code>getppid()</code>	<code>sendmsg()</code>	<code>tcgetpgrp()</code>
17072	<code>_exit()</code>	<code>getsockname()</code>	<code>sendto()</code>	<code>tcsendbreak()</code>
17073	<code>abort()</code>	<code>getsockopt()</code>	<code>setgid()</code>	<code>tcsetattr()</code>
17074	<code>accept()</code>	<code>getuid()</code>	<code>setpgid()</code>	<code>tcsetpgrp()</code>
17075	<code>access()</code>	<code>htonl()</code>	<code>setsid()</code>	<code>time()</code>
17076	<code>aio_error()</code>	<code>htons()</code>	<code>setsockopt()</code>	<code>timer_getoverrun()</code>
17077	<code>aio_return()</code>	<code>kill()</code>	<code>setuid()</code>	<code>timer_gettime()</code>
17078	<code>aio_suspend()</code>	<code>link()</code>	<code>shutdown()</code>	<code>timer_settime()</code>
17079	<code>alarm()</code>	<code>linkat()</code>	<code>sigaction()</code>	<code>times()</code>
17080	<code>bind()</code>	<code>listen()</code>	<code>sigaddset()</code>	<code>umask()</code>
17081	<code>cfgetispeed()</code>	<code>longjmp()</code>	<code>sigdelset()</code>	<code>uname()</code>
17082	<code>cfgetospeed()</code>	<code>lseek()</code>	<code>sigemptyset()</code>	<code>unlink()</code>
17083	<code>cfsetispeed()</code>	<code>lstat()</code>	<code>sigfillset()</code>	<code>unlinkat()</code>
17084	<code>cfsetospeed()</code>	<code>memccpy()</code>	<code>sigismember()</code>	<code>utime()</code>
17085	<code>chdir()</code>	<code>memchr()</code>	<code>siglongjmp()</code>	<code>utimensat()</code>
17086	<code>chmod()</code>	<code>memcmp()</code>	<code>signal()</code>	<code>utimes()</code>
17087	<code>chown()</code>	<code>memcpy()</code>	<code>sigpause()</code>	<code>wait()</code>
17088	<code>clock_gettime()</code>	<code>memmove()</code>	<code>sigpending()</code>	<code>waitpid()</code>
17089	<code>close()</code>	<code>memset()</code>	<code>sigprocmask()</code>	<code>wcpcpy()</code>
17090	<code>connect()</code>	<code>mkdir()</code>	<code>sigqueue()</code>	<code>wcpncpy()</code>
17091	<code>creat()</code>	<code>mkdirat()</code>	<code>sigset()</code>	<code>wscat()</code>
17092	<code>dup()</code>	<code>mkfifo()</code>	<code>sigsuspend()</code>	<code>wcschr()</code>
17093	<code>dup2()</code>	<code>mkfifoat()</code>	<code>sleep()</code>	<code>wscpy()</code>
17094	<code>execl()</code>	<code>mknod()</code>	<code>socketatmark()</code>	<code>wscspn()</code>
17095	<code>execle()</code>	<code>mknodat()</code>	<code>socket()</code>	<code>wcslen()</code>
17096	<code>execv()</code>	<code>ntohl()</code>	<code>socketpair()</code>	<code>wcsncat()</code>
17097	<code>execve()</code>	<code>ntohs()</code>	<code>stat()</code>	<code>wcsncmp()</code>
17098	<code>faccessat()</code>	<code>open()</code>	<code>stpncpy()</code>	<code>wcsncpy()</code>
17099	<code>fchdir()</code>	<code>openat()</code>	<code>stpncpy()</code>	<code>wcsnlen()</code>
17100	<code>fchmod()</code>	<code>pause()</code>	<code>strcat()</code>	<code>wcsnbrk()</code>
17101	<code>fchmodat()</code>	<code>pipe()</code>	<code>strchr()</code>	<code>wcsrchr()</code>
17102	<code>fchown()</code>	<code>poll()</code>	<code>strcmp()</code>	<code>wcsspn()</code>
17103	<code>fchownat()</code>	<code>posix_trace_event()</code>	<code>strcpy()</code>	<code>wcsstr()</code>
17104	<code>fcntl()</code>	<code>pselect()</code>	<code>strcspn()</code>	<code>wcstok()</code>
17105	<code>fdatasync()</code>	<code>pthread_kill()</code>	<code>strlen()</code>	<code>wmemchr()</code>
17106	<code>fexecve()</code>	<code>pthread_self()</code>	<code>strncat()</code>	<code>wmemcmp()</code>
17107	<code>ffs()</code>	<code>pthread_sigmask()</code>	<code>strncmp()</code>	<code>wmemcpy()</code>
17108	<code>fork()</code>	<code>raise()</code>	<code>strncpy()</code>	<code>wmemmove()</code>
17109	<code>fstat()</code>	<code>read()</code>	<code>strlen()</code>	<code>wmemset()</code>
17110	<code>fstatat()</code>	<code>readlink()</code>	<code>strpbrk()</code>	<code>write()</code>
17111	<code>fsync()</code>	<code>readlinkat()</code>	<code>strrchr()</code>	
17112	<code>ftruncate()</code>	<code>recv()</code>	<code>strspn()</code>	
17113	<code>futimens()</code>	<code>recvfrom()</code>	<code>strstr()</code>	
17114	<code>getegid()</code>	<code>recvmsg()</code>	<code>strtok_r()</code>	
17115	<code>geteuid()</code>	<code>rename()</code>	<code>symlink()</code>	
17116	<code>getgid()</code>	<code>renameat()</code>	<code>symlinkat()</code>	
17117	<code>getgroups()</code>	<code>rmdir()</code>	<code>tcdrain()</code>	
17118	<code>getpeername()</code>	<code>select()</code>	<code>tcflow()</code>	
17119	<code>getpgrp()</code>	<code>sem_post()</code>	<code>tcflush()</code>	
17120	<code>getpid()</code>	<code>send()</code>	<code>tcgetattr()</code>	

All functions not in the above table may be unsafe with respect to signals. Implementations may make other interfaces async-signal-safe. In the presence of signals, all functions defined by this

volume of POSIX.1-2008 shall behave as defined when called from or interrupted by a signal-catching function, with the exception that when a signal interrupts an unsafe function or equivalent (such as the processing equivalent to *exit()* performed after a return from the initial call to *main()*) and the signal-catching function calls an unsafe function, the behavior is undefined. Additional exceptions are specified in the descriptions of individual functions such as *longjmp()*.

Operations which obtain the value of *errno* and operations which assign a value to *errno* shall be async-signal-safe, provided that the signal-catching function saves the value of *errno* upon entry and restores it before it returns.

When a signal is delivered to a thread, if the action of that signal specifies termination, stop, or continue, the entire process shall be terminated, stopped, or continued, respectively.

#### 2.4.4 Signal Effects on Other Functions

Signals affect the behavior of certain functions defined by this volume of POSIX.1-2008 if delivered to a process while it is executing such a function. If the action of the signal is to terminate the process, the process shall be terminated and the function shall not return. If the action of the signal is to stop the process, the process shall stop until continued or terminated. Generation of a SIGCONT signal for the process shall cause the process to be continued, and the original function shall continue at the point the process was stopped. If the action of the signal is to invoke a signal-catching function, the signal-catching function shall be invoked; in this case the original function is said to be “interrupted” by the signal. If the signal-catching function executes a **return** statement, the behavior of the interrupted function shall be as described individually for that function, except as noted for unsafe functions. After returning from a signal-catching function, the value of *errno* is unspecified if the signal-catching function or any function it called assigned a value to *errno* and the signal-catching function did not save and restore the original value of *errno*. Signals that are ignored shall not affect the behavior of any function; signals that are blocked shall not affect the behavior of any function until they are unblocked and then delivered, except as specified for the *sigpending()* and *sigwait()* functions.

### 2.5 Standard I/O Streams

A stream is associated with an external file (which may be a physical device) or memory buffer by “opening” a file or buffer. This may involve “creating” a new file. Creating an existing file causes its former contents to be discarded if necessary. If a file can support positioning requests (such as a disk file, as opposed to a terminal), then a “file position indicator” associated with the stream is positioned at the start (byte number 0) of the file, unless the file is opened with append mode, in which case it is implementation-defined whether the file position indicator is initially positioned at the beginning or end of the file. The file position indicator is maintained by subsequent reads, writes, and positioning requests, to facilitate an orderly progression through the file.

The wide-character input functions shall read characters from the stream and convert them to wide characters as if they were read by successive calls to the *fgetwc()* function. Each conversion shall occur as if by a call to the *mbrtowc()* function, with the conversion state described by the stream’s own **mbstate\_t** object (see Section 2.5.2, on page 498). The byte input functions shall read characters from the stream as if by successive calls to the *fgetc()* function.

The wide-character output functions shall convert wide characters to characters and write them to the stream as if they were written by successive calls to the *fputwc()* function. Each conversion

shall occur as if by a call to the *wcrtomb()* function, with the conversion state described by the stream's own **mbstate\_t** object (see [Section 2.5.2](#), on page 498). The byte output functions shall write characters to the stream as if by successive calls to the *fputc()* function.

The *perror()*, *psiginfo()*, and *psignal()* functions shall behave as described above for the byte output functions if the stream is already byte-oriented, and shall behave as described above for the wide-character output functions if the stream is already wide-oriented. If the stream has no orientation, they shall behave as described for the byte output functions except that they shall not change the orientation of the stream.

Functions other than *perror()*, *psiginfo()*, and *psignal()* that write to streams but are neither wide-character output nor byte output functions (*getopt()* and *wordexp()*), shall behave as described above for the byte output functions, except that if the stream has no orientation, it is unspecified whether they set the stream to byte orientation or leave it with no orientation.

When a stream is “unbuffered”, bytes are intended to appear from the source or at the destination as soon as possible; otherwise, bytes may be accumulated and transmitted as a block. When a stream is “fully buffered”, bytes are intended to be transmitted as a block when a buffer is filled. When a stream is “line buffered”, bytes are intended to be transmitted as a block when a <newline> byte is encountered. Furthermore, bytes are intended to be transmitted as a block when a buffer is filled, when input is requested on an unbuffered stream, or when input is requested on a line-buffered stream that requires the transmission of bytes. Support for these characteristics is implementation-defined, and may be affected via *setbuf()* and *setvbuf()*.

A file may be disassociated from a controlling stream by “closing” the file. Output streams are flushed (any unwritten buffer contents are transmitted) before the stream is disassociated from the file. The value of a pointer to a **FILE** object is unspecified after the associated file is closed (including the standard streams).

A file may be subsequently reopened, by the same or another program execution, and its contents reclaimed or modified (if it can be repositioned at its start). If the *main()* function returns to its original caller, or if the *exit()* function is called, all open files are closed (hence all output streams are flushed) before program termination. Other paths to program termination, such as calling *abort()*, need not close all files properly.

The address of the **FILE** object used to control a stream may be significant; a copy of a **FILE** object need not necessarily serve in place of the original.

At program start-up, three streams are predefined and need not be opened explicitly: *standard input* (for reading conventional input), *standard output* (for writing conventional output), and *standard error* (for writing diagnostic output). When opened, the standard error stream is not fully buffered; the standard input and standard output streams are fully buffered if and only if the stream can be determined not to refer to an interactive device.

CX A stream associated with a memory buffer shall have the same operations for text files that a stream associated with an external file would have. In addition, the stream orientation shall be determined in exactly the same fashion.

Input and output operations on a stream associated with a memory buffer by a call to *fmemopen()* shall be constrained by the implementation to take place within the bounds of the memory buffer. In the case of a stream opened by *open\_memstream()* or *open\_wmemstream()*, the memory area shall grow dynamically to accommodate write operations as necessary. For output, data is moved from the buffer provided by *setvbuf()* to the memory stream during a flush or close operation.

## 2.5.1 Interaction of File Descriptors and Standard I/O Streams

CX This section describes the interaction of file descriptors and standard I/O streams. The functionality described in this section is an extension to the ISO C standard (and the rest of this section is not further CX shaded).

An open file description may be accessed through a file descriptor, which is created using functions such as *open()* or *pipe()*, or through a stream, which is created using functions such as *fopen()* or *popen()*. Either a file descriptor or a stream is called a “handle” on the open file description to which it refers; an open file description may have several handles.

Handles can be created or destroyed by explicit user action, without affecting the underlying open file description. Some of the ways to create them include *fcntl()*, *dup()*, *fdopen()*, *fileno()*, and *fork()*. They can be destroyed by at least *fclose()*, *close()*, and the *exec* functions.

A file descriptor that is never used in an operation that could affect the file offset (for example, *read()*, *write()*, or *lseek()*) is not considered a handle for this discussion, but could give rise to one (for example, as a consequence of *fdopen()*, *dup()*, or *fork()*). This exception does not include the file descriptor underlying a stream, whether created with *fopen()* or *fdopen()*, so long as it is not used directly by the application to affect the file offset. The *read()* and *write()* functions implicitly affect the file offset; *lseek()* explicitly affects it.

The result of function calls involving any one handle (the “active handle”) is defined elsewhere in this volume of POSIX.1-2008, but if two or more handles are used, and any one of them is a stream, the application shall ensure that their actions are coordinated as described below. If this is not done, the result is undefined.

A handle which is a stream is considered to be closed when either an *fclose()*, or *freopen()* with non-full filename, is executed on it (for *freopen()* with a null filename, it is implementation-defined whether a new handle is created or the existing one reused), or when the process owning that stream terminates with *exit()*, *abort()*, or due to a signal. A file descriptor is closed by *close()*, *\_exit()*, or the *exec* functions when *FD\_CLOEXEC* is set on that file descriptor.

For a handle to become the active handle, the application shall ensure that the actions below are performed between the last use of the handle (the current active handle) and the first use of the second handle (the future active handle). The second handle then becomes the active handle. All activity by the application affecting the file offset on the first handle shall be suspended until it again becomes the active file handle. (If a stream function has as an underlying function one that affects the file offset, the stream function shall be considered to affect the file offset.)

The handles need not be in the same process for these rules to apply.

Note that after a *fork()*, two handles exist where one existed before. The application shall ensure that, if both handles can ever be accessed, they are both in a state where the other could become the active handle first. The application shall prepare for a *fork()* exactly as if it were a change of active handle. (If the only action performed by one of the processes is one of the *exec* functions or *\_exit()* (not *exit()*), the handle is never accessed in that process.)

For the first handle, the first applicable condition below applies. After the actions required below are taken, if the handle is still open, the application can close it.

- If it is a file descriptor, no action is required.
- If the only further action to be performed on any handle to this open file descriptor is to close it, no action need be taken.
- If it is a stream which is unbuffered, no action need be taken.

- If it is a stream which is line buffered, and the last byte written to the stream was a <newline> (that is, as if a:

`putc( '\n' )`

was the most recent operation on that stream), no action need be taken.

- If it is a stream which is open for writing or appending (but not also open for reading), the application shall either perform an *fflush()*, or the stream shall be closed.
- If the stream is open for reading and it is at the end of the file (*feof()* is true), no action need be taken.
- If the stream is open with a mode that allows reading and the underlying open file description refers to a device that is capable of seeking, the application shall either perform an *fflush()*, or the stream shall be closed.

For the second handle:

- If any previous active handle has been used by a function that explicitly changed the file offset, except as required above for the first handle, the application shall perform an *lseek()* or *fseek()* (as appropriate to the type of handle) to an appropriate location.

If the active handle ceases to be accessible before the requirements on the first handle, above, have been met, the state of the open file description becomes undefined. This might occur during functions such as a *fork()* or *\_exit()*.

The *exec* functions make inaccessible all streams that are open at the time they are called, independent of which streams or file descriptors may be available to the new process image.

When these rules are followed, regardless of the sequence of handles used, implementations shall ensure that an application, even one consisting of several processes, shall yield correct results: no data shall be lost or duplicated when writing, and all data shall be written in order, except as requested by seeks. It is implementation-defined whether, and under what conditions, all input is seen exactly once.

Each function that operates on a stream is said to have zero or more “underlying functions”. This means that the stream function shares certain traits with the underlying functions, but does not require that there be any relation between the implementations of the stream function and its underlying functions.

## 2.5.2 Stream Orientation and Encoding Rules

For conformance to the ISO/IEC 9899:1999 standard, the definition of a stream includes an “orientation”. After a stream is associated with an external file, but before any operations are performed on it, the stream is without orientation. Once a wide-character input/output function has been applied to a stream without orientation, the stream shall become “wide-oriented”. Similarly, once a byte input/output function has been applied to a stream without orientation, the stream shall become “byte-oriented”. Only a call to the *freopen()* function or the *fwide()* function can otherwise alter the orientation of a stream.

A successful call to *freopen()* shall remove any orientation. The three predefined streams *standard input*, *standard output*, and *standard error* shall be unoriented at program start-up.

Byte input/output functions cannot be applied to a wide-oriented stream, and wide-character input/output functions cannot be applied to a byte-oriented stream. The remaining stream operations shall not affect and shall not be affected by a stream’s orientation, except for the following additional restriction:



- For wide-oriented streams, after a successful call to a file-positioning function that leaves the file position indicator prior to the end-of-file, a wide-character output function can overwrite a partial character; any file contents beyond the byte(s) written are henceforth undefined.

Each wide-oriented stream has an associated **mbstate\_t** object that stores the current parse state of the stream. A successful call to *fgetpos()* shall store a representation of the value of this **mbstate\_t** object as part of the value of the **fpos\_t** object. A later successful call to *fsetpos()* using the same stored **fpos\_t** value shall restore the value of the associated **mbstate\_t** object as well as the position within the controlled stream.

Implementations that support multiple encoding rules associate an encoding rule with the stream. The encoding rule shall be determined by the setting of the *LC\_CTYPE* category in the current locale at the time when the stream becomes wide-oriented. As with the stream's orientation, the encoding rule associated with a stream cannot be changed once it has been set, except by a successful call to *freopen()* which clears the encoding rule and resets the orientation to unoriented.

Although wide-oriented streams are conceptually sequences of wide characters, the external file associated with a wide-oriented stream is a sequence of (possibly multi-byte) characters generalized as follows:

- Multi-byte encodings within files may contain embedded null bytes (unlike multi-byte encodings valid for use internal to the program).
- A file need not begin nor end in the initial shift state.

Moreover, the encodings used for characters may differ among files. Both the nature and choice of such encodings are implementation-defined.

The wide-character input functions read characters from the stream and convert them to wide characters as if they were read by successive calls to the *fgetwc()* function. Each conversion shall occur as if by a call to the *mbrtowc()* function, with the conversion state described by the stream's own **mbstate\_t** object, except the encoding rule associated with the stream is used instead of the encoding rule implied by the *LC\_CTYPE* category of the current locale.

The wide-character output functions convert wide characters to (possibly multi-byte) characters and write them to the stream as if they were written by successive calls to the *fputwc()* function. Each conversion shall occur as if by a call to the *wcrtomb()* function, with the conversion state described by the stream's own **mbstate\_t** object, except the encoding rule associated with the stream is used instead of the encoding rule implied by the *LC\_CTYPE* category of the current locale.

An "encoding error" shall occur if the character sequence presented to the underlying *mbrtowc()* function does not form a valid (generalized) character, or if the code value passed to the underlying *wcrtomb()* function does not correspond to a valid (generalized) character. The wide-character input/output functions and the byte input/output functions store the value of the macro `[EILSEQ]` in *errno* if and only if an encoding error occurs.

## 2.6 STREAMS

OB XSR STREAMS functionality is provided on implementations supporting the XSI STREAMS Option Group. The functionality described in this section is dependent on support of the XSI STREAMS option (and the rest of this section is not further shaded for this option).

STREAMS provides a uniform mechanism for implementing networking services and other character-based I/O. The STREAMS function provides direct access to protocol modules. STREAMS modules are unspecified objects. Access to STREAMS modules is provided by interfaces in POSIX.1-2008. Creation of STREAMS modules is outside the scope of POSIX.1-2008.

A STREAM is typically a full-duplex connection between a process and an open device or pseudo-device. However, since pipes may be STREAMS-based, a STREAM can be a full-duplex connection between two processes. The STREAM itself exists entirely within the implementation and provides a general character I/O function for processes. It optionally includes one or more intermediate processing modules that are interposed between the process end of the STREAM (STREAM head) and a device driver at the end of the STREAM (STREAM end).

STREAMS I/O is based on messages. There are three types of message:

- *Data messages* containing actual data for input or output
- *Control data* containing instructions for the STREAMS modules and underlying implementation
- Other messages, which include file descriptors

The interface between the STREAM and the rest of the implementation is provided by a set of functions at the STREAM head. When a process calls *write()*, *writew()*, *putmsg()*, *putpmsg()*, or *ioctl()*, messages are sent down the STREAM, and *read()*, *readv()*, *getmsg()*, or *getpmsg()* accepts data from the STREAM and passes it to a process. Data intended for the device at the downstream end of the STREAM is packaged into messages and sent downstream, while data and signals from the device are composed into messages by the device driver and sent upstream to the STREAM head.

When a STREAMS-based device is opened, a STREAM shall be created that contains the STREAM head and the STREAM end (driver). If pipes are STREAMS-based in an implementation, when a pipe is created, two STREAMS shall be created, each containing a STREAM head. Other modules are added to the STREAM using *ioctl()*. New modules are “pushed” onto the STREAM one at a time in last-in, first-out (LIFO) style, as though the STREAM was a push-down stack.

### Priority

Message types are classified according to their queuing priority and may be *normal* (non-priority), *priority*, or *high-priority* messages. A message belongs to a particular priority band that determines its ordering when placed on a queue. Normal messages have a priority band of 0 and shall always be placed at the end of the queue following all other messages in the queue. High-priority messages are always placed at the head of a queue, but shall be discarded if there is already a high-priority message in the queue. Their priority band shall be ignored; they are high-priority by virtue of their type. Priority messages have a priority band greater than 0. Priority messages are always placed after any messages of the same or higher priority. High-priority and priority messages are used to send control and data information outside the normal flow of control. By convention, high-priority messages shall not be affected by flow control. Normal and priority messages have separate flow controls.



## Message Parts

A process may access STREAMS messages that contain a data part, control part, or both. The data part is that information which is transmitted over the communication medium and the control information is used by the local STREAMS modules. The other types of messages are used between modules and are not accessible to processes. Messages containing only a data part are accessible via *putmsg()*, *putpmsg()*, *getmsg()*, *getpmsg()*, *read()*, *readv()*, *write()*, or *writv()*. Messages containing a control part with or without a data part are accessible via calls to *putmsg()*, *putpmsg()*, *getmsg()*, or *getpmsg()*.

### 2.6.1 Accessing STREAMS

A process accesses STREAMS-based files using the standard functions *close()*, *ioctl()*, *getmsg()*, *getpmsg()*, *open()*, *pipe()*, *poll()*, *putmsg()*, *putpmsg()*, *read()*, or *write()*. Refer to the applicable function definitions for general properties and errors.

Calls to *ioctl()* shall perform control functions on the STREAM associated with the file descriptor *fildev*. The control functions may be performed by the STREAM head, a STREAMS module, or the STREAMS driver for the STREAM.

STREAMS modules and drivers can detect errors, sending an error message to the STREAM head, thus causing subsequent functions to fail and set *errno* to the value specified in the message. In addition, STREAMS modules and drivers can elect to fail a particular *ioctl()* request alone by sending a negative acknowledgement message to the STREAM head. This shall cause just the pending *ioctl()* request to fail and set *errno* to the value specified in the message.

## 2.7 XSI Interprocess Communication

XSI This section describes extensions to support interprocess communication. The functionality described in this section shall be provided on implementations that support the XSI option (and the rest of this section is not further shaded).

The following message passing, semaphore, and shared memory services form an XSI interprocess communication facility. Certain aspects of their operation are common, and are defined as follows.

IPC Functions		
<i>msgctl()</i>	<i>semctl()</i>	<i>shmat()</i>
<i>msgget()</i>	<i>semget()</i>	<i>shmctl()</i>
<i>msgrcv()</i>	<i>semop()</i>	<i>shmdt()</i>
<i>msgsnd()</i>		<i>shmget()</i>

Another interprocess communication facility is provided by functions in the Realtime Option Group; see [Section 2.8](#) (on page 503).

## 2.7.1 IPC General Description

Each individual shared memory segment, message queue, and semaphore set shall be identified by a unique positive integer, called, respectively, a shared memory identifier, *shmid*, a semaphore identifier, *semid*, and a message queue identifier, *msqid*. The identifiers shall be returned by calls to *shmget()*, *semget()*, and *msgget()*, respectively.

Associated with each identifier is a data structure which contains data related to the operations which may be or may have been performed; see the Base Definitions volume of POSIX.1-2008, **<sys/shm.h>**, **<sys/sem.h>**, and **<sys/msg.h>** for their descriptions.

Each of the data structures contains both ownership information and an **ipc\_perm** structure (see the Base Definitions volume of POSIX.1-2008, **<sys/ipc.h>**) which are used in conjunction to determine whether or not read/write (read/alter for semaphores) permissions should be granted to processes using the IPC facilities. The *mode* member of the **ipc\_perm** structure acts as a bit field which determines the permissions.

The values of the bits are given below in octal notation.

Bit	Meaning
0400	Read by user.
0200	Write by user.
0040	Read by group.
0020	Write by group.
0004	Read by others.
0002	Write by others.

The name of the **ipc\_perm** structure is *shm\_perm*, *sem\_perm*, or *msg\_perm*, depending on which service is being used. In each case, read and write/alter permissions shall be granted to a process if one or more of the following are true ("xxx" is replaced by *shm*, *sem*, or *msg*, as appropriate):

- The process has appropriate privileges.
- The effective user ID of the process matches *xxx\_perm.cuid* or *xxx\_perm.uid* in the data structure associated with the IPC identifier, and the appropriate bit of the *user* field in *xxx\_perm.mode* is set.
- The effective user ID of the process does not match *xxx\_perm.cuid* or *xxx\_perm.uid* but the effective group ID of the process matches *xxx\_perm.cgid* or *xxx\_perm.gid* in the data structure associated with the IPC identifier, and the appropriate bit of the *group* field in *xxx\_perm.mode* is set.
- The effective user ID of the process does not match *xxx\_perm.cuid* or *xxx\_perm.uid* and the effective group ID of the process does not match *xxx\_perm.cgid* or *xxx\_perm.gid* in the data structure associated with the IPC identifier, but the appropriate bit of the *other* field in *xxx\_perm.mode* is set.

Otherwise, the permission shall be denied.

In addition to the **ipc\_perm** structure, each associated data structure includes several **time\_t** fields for recording timestamps of particular operations. When an operation is described as setting a timestamp to the current time, that particular timestamp member of the associated data structure shall be set to the largest **time\_t** value which is not greater than the current time.

## 2.8 Realtime

This section defines functions to support the source portability of applications with realtime requirements. The presence of some of these functions is dependent on support for implementation options described in the text.

The specific functional areas included in this section and their scope include the following. Full definitions of these terms can be found in XBD [Chapter 3](#) (on page 33).

- Semaphores
- Process Memory Locking
- Memory Mapped Files and Shared Memory Objects
- Priority Scheduling
- Realtime Signal Extension
- Timers
- Interprocess Communication
- Synchronized Input and Output
- Asynchronous Input and Output

All the realtime functions defined in this volume of POSIX.1-2008 are portable, although some of the numeric parameters used by an implementation may have hardware dependencies.

### 2.8.1 Realtime Signals

See [Section 2.4.2](#) (on page 489).

### 2.8.2 Asynchronous I/O

An asynchronous I/O control block structure **aio** is used in many asynchronous I/O functions. It is defined in the Base Definitions volume of POSIX.1-2008, **<aio.h>** and has at least the following members:

Member Type	Member Name	Description
<b>int</b>	<i>aio_fildes</i>	File descriptor.
<b>off_t</b>	<i>aio_offset</i>	File offset.
<b>volatile void*</b>	<i>aio_buf</i>	Location of buffer.
<b>size_t</b>	<i>aio_nbytes</i>	Length of transfer.
<b>int</b>	<i>aio_reqprio</i>	Request priority offset.
<b>struct sigevent</b>	<i>aio_sigevent</i>	Signal number and value.
<b>int</b>	<i>aio_lio_opcode</i>	Operation to be performed.

The *aio\_fildes* element is the file descriptor on which the asynchronous operation is performed.

If **O\_APPEND** is not set for the file descriptor *aio\_fildes* and if *aio\_fildes* is associated with a device that is capable of seeking, then the requested operation takes place at the absolute position in the file as given by *aio\_offset*, as if *lseek()* were called immediately prior to the operation with an *offset* argument equal to *aio\_offset* and a *whence* argument equal to **SEEK\_SET**. If **O\_APPEND** is set for the file descriptor, or if *aio\_fildes* is associated with a device that is incapable of seeking, write operations append to the file in the same order as the calls were made, with the following exception: under implementation-defined circumstances, such as

operation on a multi-processor or when requests of differing priorities are submitted at the same time, the ordering restriction may be relaxed. Since there is no way for a strictly conforming application to determine whether this relaxation applies, all strictly conforming applications which rely on ordering of output shall be written in such a way that they will operate correctly if the relaxation applies. After a successful call to enqueue an asynchronous I/O operation, the value of the file offset for the file is unspecified. The *aio\_nbytes* and *aio\_buf* elements are the same as the *nbyte* and *buf* arguments defined by *read()* and *write()*, respectively.

If `_POSIX_PRIORITIZED_IO` and `_POSIX_PRIORITY_SCHEDULING` are defined, then asynchronous I/O is queued in priority order, with the priority of each asynchronous operation based on the current scheduling priority of the calling process. The *aio\_reqprio* member can be used to lower (but not raise) the asynchronous I/O operation priority and is within the range zero through `{AIO_PRIO_DELTA_MAX}`, inclusive. Unless both `_POSIX_PRIORITIZED_IO` and `_POSIX_PRIORITY_SCHEDULING` are defined, the order of processing asynchronous I/O requests is unspecified. When both `_POSIX_PRIORITIZED_IO` and `_POSIX_PRIORITY_SCHEDULING` are defined, the order of processing of requests submitted by processes whose schedulers are not `SCHED_FIFO`, `SCHED_RR`, or `SCHED_SPORADIC` is unspecified. The priority of an asynchronous request is computed as (process scheduling priority) minus *aio\_reqprio*. The priority assigned to each asynchronous I/O request is an indication of the desired order of execution of the request relative to other asynchronous I/O requests for this file. If `_POSIX_PRIORITIZED_IO` is defined, requests issued with the same priority to a character special file are processed by the underlying device in FIFO order; the order of processing of requests of the same priority issued to files that are not character special files is unspecified. Numerically higher priority values indicate requests of higher priority. The value of *aio\_reqprio* has no effect on process scheduling priority. When prioritized asynchronous I/O requests to the same file are blocked waiting for a resource required for that I/O operation, the higher-priority I/O requests shall be granted the resource before lower-priority I/O requests are granted the resource. The relative priority of asynchronous I/O and synchronous I/O is implementation-defined. If `_POSIX_PRIORITIZED_IO` is defined, the implementation shall define for which files I/O prioritization is supported.

The *aio\_sigevent* determines how the calling process shall be notified upon I/O completion, as specified in [Section 2.4.1](#) (on page 488). If *aio\_sigevent.sigev\_notify* is `SIGEV_NONE`, then no signal shall be posted upon I/O completion, but the error status for the operation and the return status for the operation shall be set appropriately.

The *aio\_lio\_opcode* field is used only by the *lio\_listio()* call. The *lio\_listio()* call allows multiple asynchronous I/O operations to be submitted at a single time. The function takes as an argument an array of pointers to **aiocb** structures. Each **aiocb** structure indicates the operation to be performed (read or write) via the *aio\_lio\_opcode* field.

The address of the **aiocb** structure is used as a handle for retrieving the error status and return status of the asynchronous operation while it is in progress.

The **aiocb** structure and the data buffers associated with the asynchronous I/O operation are being used by the system for asynchronous I/O while, and only while, the error status of the asynchronous operation is equal to `[EINPROGRESS]`. Applications shall not modify the **aiocb** structure while the structure is being used by the system for asynchronous I/O.

The return status of the asynchronous operation is the number of bytes transferred by the I/O operation. If the error status is set to indicate an error completion, then the return status is set to the return value that the corresponding *read()*, *write()*, or *fsync()* call would have returned. When the error status is not equal to `[EINPROGRESS]`, the return status shall reflect the return status of the corresponding synchronous operation.

## 17546 2.8.3 Memory Management

### 17547 2.8.3.1 Memory Locking

17548 MLR Range memory locking operations are defined in terms of pages. Implementations may restrict  
17549 the size and alignment of range lockings to be on page-size boundaries. The page size, in bytes,  
17550 is the value of the configurable system variable {PAGESIZE}. If an implementation has no  
17551 restrictions on size or alignment, it may specify a 1-byte page size.

17552 ML|MLR Memory locking guarantees the residence of portions of the address space. It is implementation-  
17553 defined whether locking memory guarantees fixed translation between virtual addresses (as  
17554 seen by the process) and physical addresses. Per-process memory locks are not inherited across a  
17555 *fork()*, and all memory locks owned by a process are unlocked upon *exec* or process termination.  
17556 Unmapping of an address range removes any memory locks established on that address range  
17557 by this process.

### 17558 2.8.3.2 Memory Mapped Files

17559 Range memory mapping operations are defined in terms of pages. Implementations may  
17560 restrict the size and alignment of range mappings to be on page-size boundaries. The page size,  
17561 in bytes, is the value of the configurable system variable {PAGESIZE}. If an implementation has  
17562 no restrictions on size or alignment, it may specify a 1-byte page size.

17563 Memory mapped files provide a mechanism that allows a process to access files by directly  
17564 incorporating file data into its address space. Once a file is mapped into a process address space,  
17565 the data can be manipulated as memory. If more than one process maps a file, its contents are  
17566 shared among them. If the mappings allow shared write access, then data written into the  
17567 memory object through the address space of one process appears in the address spaces of all  
17568 processes that similarly map the same portion of the memory object.

17569 SHM Shared memory objects are named regions of storage that may be independent of the file system  
17570 and can be mapped into the address space of one or more processes to allow them to share the  
17571 associated memory.

17572 SHM An *unlink()* of a file or *shm\_unlink()* of a shared memory object, while causing the removal of  
17573 the name, does not unmap any mappings established for the object. Once the name has been  
17574 removed, the contents of the memory object are preserved as long as it is referenced. The  
17575 memory object remains referenced as long as a process has the memory object open or has some  
17576 area of the memory object mapped.

### 17577 2.8.3.3 Memory Protection

17578 When an object is mapped, various application accesses to the mapped region may result in  
17579 signals. In this context, SIGBUS is used to indicate an error using the mapped object, and  
17580 SIGSEGV is used to indicate a protection violation or misuse of an address:

- 17581 • A mapping may be restricted to disallow some types of access.
- 17582 • Write attempts to memory that was mapped without write access, or any access to  
17583 memory mapped PROT\_NONE, shall result in a SIGSEGV signal.
- 17584 • References to unmapped addresses shall result in a SIGSEGV signal.

- 17585 • Reference to whole pages within the mapping, but beyond the current length of the object,
- 17586 shall result in a SIGBUS signal.
- 17587 • The size of the object is unaffected by access beyond the end of the object (even if a
- 17588 SIGBUS is not generated).

#### 17589 2.8.3.4 Typed Memory Objects

17590 TYM The functionality described in this section shall be provided on implementations that support  
 17591 the Typed Memory Objects option (and the rest of this section is not further shaded for this  
 17592 option).

17593 Implementations may support the Typed Memory Objects option independently of support for  
 17594 memory mapped files or shared memory objects. Typed memory objects are implementation-  
 17595 configurable named storage pools accessible from one or more processors in a system, each via  
 17596 one or more ports, such as backplane buses, LANs, I/O channels, and so on. Each valid  
 17597 combination of a storage pool and a port is identified through a name that is defined at system  
 17598 configuration time, in an implementation-defined manner; the name may be independent of the  
 17599 file system. Using this name, a typed memory object can be opened and mapped into process  
 17600 address space. For a given storage pool and port, it is necessary to support both dynamic  
 17601 allocation from the pool as well as mapping at an application-supplied offset within the pool;  
 17602 when dynamic allocation has been performed, subsequent deallocation must be supported.  
 17603 Lastly, accessing typed memory objects from different ports requires a method for obtaining the  
 17604 offset and length of contiguous storage of a region of typed memory (dynamically allocated or  
 17605 not); this allows typed memory to be shared among processes and/or processors while being  
 17606 accessed from the desired port.

### 17607 2.8.4 Process Scheduling

17608 PS The functionality described in this section shall be provided on implementations that support  
 17609 the Process Scheduling option (and the rest of this section is not further shaded for this option).

#### 17610 Scheduling Policies

17611 The scheduling semantics described in this volume of POSIX.1-2008 are defined in terms of a  
 17612 conceptual model that contains a set of thread lists. No implementation structures are  
 17613 necessarily implied by the use of this conceptual model. It is assumed that no time elapses  
 17614 during operations described using this model, and therefore no simultaneous operations are  
 17615 possible. This model discusses only processor scheduling for runnable threads, but it should be  
 17616 noted that greatly enhanced predictability of realtime applications results if the sequencing of  
 17617 other resources takes processor scheduling policy into account.

17618 There is, conceptually, one thread list for each priority. A runnable thread will be on the thread  
 17619 list for that thread's priority. Multiple scheduling policies shall be provided. Each non-empty  
 17620 thread list is ordered, contains a head as one end of its order, and a tail as the other. The purpose  
 17621 of a scheduling policy is to define the allowable operations on this set of lists (for example,  
 17622 moving threads between and within lists).

17623 The POSIX model treats a "process" as an aggregation of system resources, including one or  
 17624 more threads that may be scheduled by the operating system on the processor(s) it controls.  
 17625 Although a process has its own set of scheduling attributes, these have an indirect effect (if any)  
 17626 on the scheduling behavior of individual threads as described below.

17627 Each thread shall be controlled by an associated scheduling policy and priority. These  
 17628 parameters may be specified by explicit application execution of the `pthread_setschedparam()`



function. Additionally, the scheduling parameters of a thread (but not its scheduling policy) may be changed by application execution of the `pthread_setschedprio()` function.

Each process shall be controlled by an associated scheduling policy and priority. These parameters may be specified by explicit application execution of the `sched_setscheduler()` or `sched_setparam()` functions.

The effect of the process scheduling attributes on individual threads in the process is dependent on the scheduling contention scope of the threads (see [Section 2.9.4](#), on page 515):

- For threads with system scheduling contention scope, the process scheduling attributes shall have no effect on the scheduling attributes or behavior either of the thread or an underlying kernel scheduling entity dedicated to that thread.
- For threads with process scheduling contention scope, the process scheduling attributes shall have no effect on the scheduling attributes of the thread. However, any underlying kernel scheduling entity used by these threads shall at all times behave as specified by the scheduling attributes of the containing process, and this behavior may affect the scheduling behavior of the process contention scope threads. For example, a process contention scope thread with scheduling policy `SCHED_FIFO` and the system maximum priority  $H$  (the value returned by `sched_get_priority_max(SCHED_FIFO)`) in a process with scheduling policy `SCHED_RR` and system minimum priority  $L$  (the value returned by `sched_get_priority_min(SCHED_RR)`) shall be subject to timeslicing and to preemption by any thread with an effective priority higher than  $L$ .

Associated with each policy is a priority range. Each policy definition shall specify the minimum priority range for that policy. The priority ranges for each policy may but need not overlap the priority ranges of other policies.

A conforming implementation shall select the thread that is defined as being at the head of the highest priority non-empty thread list to become a running thread, regardless of its associated policy. This thread is then removed from its thread list.

Four scheduling policies are specifically required. Other implementation-defined scheduling policies may be defined. The following symbols are defined in the Base Definitions volume of POSIX.1-2008, `<sched.h>`:

`SCHED_FIFO` First in, first out (FIFO) scheduling policy.

`SCHED_RR` Round robin scheduling policy.

`SCHED_SPORADIC` Sporadic server scheduling policy.

`SCHED_OTHER` Another scheduling policy.

The values of these symbols shall be distinct.

## **SCHED\_FIFO**

Conforming implementations shall include a scheduling policy called the FIFO scheduling policy.

Threads scheduled under this policy are chosen from a thread list that is ordered by the time its threads have been on the list without being executed; generally, the head of the list is the thread that has been on the list the longest time, and the tail is the thread that has been on the list the shortest time.

Under the `SCHED_FIFO` policy, the modification of the definitional thread lists is as follows:

1. When a running thread becomes a preempted thread, it becomes the head of the thread list for its priority.
2. When a blocked thread becomes a runnable thread, it becomes the tail of the thread list for its priority.
3. When a running thread calls the *sched\_setscheduler()* function, the process specified in the function call is modified to the specified policy and the priority specified by the *param* argument.
4. When a running thread calls the *sched\_setparam()* function, the priority of the process specified in the function call is modified to the priority specified by the *param* argument.
5. When a running thread calls the *pthread\_setschedparam()* function, the thread specified in the function call is modified to the specified policy and the priority specified by the *param* argument.
6. When a running thread calls the *pthread\_setschedprio()* function, the thread specified in the function call is modified to the priority specified by the *prio* argument.
7. If a thread whose policy or priority has been modified other than by *pthread\_setschedprio()* is a running thread or is runnable, it then becomes the tail of the thread list for its new priority.
8. If a thread whose priority has been modified by *pthread\_setschedprio()* is a running thread or is runnable, the effect on its position in the thread list depends on the direction of the modification, as follows:
  - a. If the priority is raised, the thread becomes the tail of the thread list.
  - b. If the priority is unchanged, the thread does not change position in the thread list.
  - c. If the priority is lowered, the thread becomes the head of the thread list.
9. When a running thread issues the *sched\_yield()* function, the thread becomes the tail of the thread list for its priority.
10. At no other time is the position of a thread with this scheduling policy within the thread lists affected.

For this policy, valid priorities shall be within the range returned by the *sched\_get\_priority\_max()* and *sched\_get\_priority\_min()* functions when SCHED\_FIFO is provided as the parameter. Conforming implementations shall provide a priority range of at least 32 priorities for this policy.

## SCHED\_RR

Conforming implementations shall include a scheduling policy called the “round robin” scheduling policy. This policy shall be identical to the SCHED\_FIFO policy with the additional condition that when the implementation detects that a running thread has been executing as a running thread for a time period of the length returned by the *sched\_rr\_get\_interval()* function or longer, the thread shall become the tail of its thread list and the head of that thread list shall be removed and made a running thread.

The effect of this policy is to ensure that if there are multiple SCHED\_RR threads at the same priority, one of them does not monopolize the processor. An application should not rely only on the use of SCHED\_RR to ensure application progress among multiple threads if the application includes threads using the SCHED\_FIFO policy at the same or higher priority levels or SCHED\_RR threads at a higher priority level.

A thread under this policy that is preempted and subsequently resumes execution as a running



thread completes the unexpired portion of its round robin interval time period.

For this policy, valid priorities shall be within the range returned by the *sched\_get\_priority\_max()* and *sched\_get\_priority\_min()* functions when *SCHED\_RR* is provided as the parameter. Conforming implementations shall provide a priority range of at least 32 priorities for this policy.

## SCHED\_SPORADIC

The functionality described in this section shall be provided on implementations that support the Process Sporadic Server or Thread Sporadic Server options (and the rest of this section is not further shaded for these options).

If *\_POSIX\_SPORADIC\_SERVER* or *\_POSIX\_THREAD\_SPORADIC\_SERVER* is defined, the implementation shall include a scheduling policy identified by the value *SCHED\_SPORADIC*.

The sporadic server policy is based primarily on two time parameters: the replenishment period and the available execution capacity. The replenishment period is given by the *sched\_ss\_repl\_period* member of the **sched\_param** structure. The available execution capacity is initialized to the value given by the *sched\_ss\_init\_budget* member of the same parameter. The sporadic server policy is identical to the *SCHED\_FIFO* policy with some additional conditions that cause the thread's assigned priority to be switched between the values specified by the *sched\_priority* and *sched\_ss\_low\_priority* members of the **sched\_param** structure.

The priority assigned to a thread using the sporadic server scheduling policy is determined in the following manner: if the available execution capacity is greater than zero and the number of pending replenishment operations is strictly less than *sched\_ss\_max\_repl*, the thread is assigned the priority specified by *sched\_priority*; otherwise, the assigned priority shall be *sched\_ss\_low\_priority*. If the value of *sched\_priority* is less than or equal to the value of *sched\_ss\_low\_priority*, the results are undefined. When active, the thread shall belong to the thread list corresponding to its assigned priority level, according to the mentioned priority assignment. The modification of the available execution capacity and, consequently of the assigned priority, is done as follows:

1. When the thread at the head of the *sched\_priority* list becomes a running thread, its execution time shall be limited to at most its available execution capacity, plus the resolution of the execution time clock used for this scheduling policy. This resolution shall be implementation-defined.
2. Each time the thread is inserted at the tail of the list associated with *sched\_priority*—because as a blocked thread it became runnable with priority *sched\_priority* or because a replenishment operation was performed—the time at which this operation is done is posted as the *activation\_time*.
3. When the running thread with assigned priority equal to *sched\_priority* becomes a preempted thread, it becomes the head of the thread list for its priority, and the execution time consumed is subtracted from the available execution capacity. If the available execution capacity would become negative by this operation, it shall be set to zero.
4. When the running thread with assigned priority equal to *sched\_priority* becomes a blocked thread, the execution time consumed is subtracted from the available execution capacity, and a replenishment operation is scheduled, as described in 6 and 7. If the available execution capacity would become negative by this operation, it shall be set to zero.
5. When the running thread with assigned priority equal to *sched\_priority* reaches the limit imposed on its execution time, it becomes the tail of the thread list for *sched\_ss\_low\_priority*, the execution time consumed is subtracted from the available execution capacity (which becomes zero), and a replenishment operation is scheduled, as

described in 6 and 7.

6. Each time a replenishment operation is scheduled, the amount of execution capacity to be replenished, *replenish\_amount*, is set equal to the execution time consumed by the thread since the *activation\_time*. The replenishment is scheduled to occur at *activation\_time* plus *sched\_ss\_repl\_period*. If the scheduled time obtained is before the current time, the replenishment operation is carried out immediately. Several replenishment operations may be pending at the same time, each of which will be serviced at its respective scheduled time. With the above rules, the number of replenishment operations simultaneously pending for a given thread that is scheduled under the sporadic server policy shall not be greater than *sched\_ss\_max\_repl*.
7. A replenishment operation consists of adding the corresponding *replenish\_amount* to the available execution capacity at the scheduled time. If, as a consequence of this operation, the execution capacity would become larger than *sched\_ss\_initial\_budget*, it shall be rounded down to a value equal to *sched\_ss\_initial\_budget*. Additionally, if the thread was runnable or running, and had assigned priority equal to *sched\_ss\_low\_priority*, then it becomes the tail of the thread list for *sched\_priority*.

Execution time is defined in XBD [Section 3.118](#) (on page 52).

For this policy, changing the value of a CPU-time clock via *clock\_settime()* shall have no effect on its behavior.

For this policy, valid priorities shall be within the range returned by the *sched\_get\_priority\_min()* and *sched\_get\_priority\_max()* functions when SCHED\_SPORADIC is provided as the parameter. Conforming implementations shall provide a priority range of at least 32 distinct priorities for this policy.

If the scheduling policy of the target process is either SCHED\_FIFO or SCHED\_RR, the *sched\_ss\_low\_priority*, *sched\_ss\_repl\_period*, and *sched\_ss\_init* budget members of the *param* argument shall have no effect on the scheduling behavior. If the scheduling policy of this process is not SCHED\_FIFO, SCHED\_RR, or SCHED\_SPORADIC, the effects of these members are implementation-defined; this case includes the SCHED\_OTHER policy.

## SCHED\_OTHER

Conforming implementations shall include one scheduling policy identified as SCHED\_OTHER (which may execute identically with either the FIFO or round robin scheduling policy). The effect of scheduling threads with the SCHED\_OTHER policy in a system in which other threads are executing under SCHED\_FIFO, SCHED\_RR, or SCHED\_SPORADIC is implementation-defined.

This policy is defined to allow strictly conforming applications to be able to indicate in a portable manner that they no longer need a realtime scheduling policy.

For threads executing under this policy, the implementation shall use only priorities within the range returned by the *sched\_get\_priority\_max()* and *sched\_get\_priority\_min()* functions when SCHED\_OTHER is provided as the parameter.

## 2.8.5 Clocks and Timers

The `<time.h>` header defines the types and manifest constants used by the timing facility.

### Time Value Specification Structures

Many of the timing facility functions accept or return time value specifications. A time value structure **timespec** specifies a single time value and includes at least the following members:

Member Type	Member Name	Description
<b>time_t</b>	<i>tv_sec</i>	Seconds.
<b>long</b>	<i>tv_nsec</i>	Nanoseconds.

The *tv\_nsec* member is only valid if greater than or equal to zero, and less than the number of nanoseconds in a second (1 000 million). The time interval described by this structure is  $(tv\_sec * 10^9 + tv\_nsec)$  nanoseconds.

A time value structure **itimerspec** specifies an initial timer value and a repetition interval for use by the per-process timer functions. This structure includes at least the following members:

Member Type	Member Name	Description
<b>struct timespec</b>	<i>it_interval</i>	Timer period.
<b>struct timespec</b>	<i>it_value</i>	Timer expiration.

If the value described by *it\_value* is non-zero, it indicates the time to or time of the next timer expiration (for relative and absolute timer values, respectively). If the value described by *it\_value* is zero, the timer shall be disarmed.

If the value described by *it\_interval* is non-zero, it specifies an interval which shall be used in reloading the timer when it expires; that is, a periodic timer is specified. If the value described by *it\_interval* is zero, the timer is disarmed after its next expiration; that is, a one-shot timer is specified.

### Timer Event Notification Control Block

Per-process timers may be created that notify the process of timer expirations by queuing a realtime extended signal. The **sigevent** structure, defined in the Base Definitions volume of POSIX.1-2008, `<signal.h>`, is used in creating such a timer. The **sigevent** structure contains the signal number and an application-specific data value which shall be used when notifying the calling process of timer expiration events.

### Manifest Constants

The following constants are defined in the Base Definitions volume of POSIX.1-2008, `<time.h>`:

<b>CLOCK_REALTIME</b>	The identifier for the system-wide realtime clock.
<b>TIMER_ABSTIME</b>	Flag indicating time is absolute with respect to the clock associated with a timer.
<b>CLOCK_MONOTONIC</b>	The identifier for the system-wide monotonic clock, which is defined as a clock whose value cannot be set via <i>clock_settime()</i> and which cannot have backward clock jumps. The maximum possible clock jump is implementation-defined.
The maximum allowable resolution for <b>CLOCK_REALTIME</b> and <b>CLOCK_MONOTONIC</b> clocks and all time services based on these clocks is represented by <code>{_POSIX_CLOCKRES_MIN}</code> and shall be defined as 20 ms (1/50 of a second). Implementations may support smaller values of	

17842 resolution for these clocks to provide finer granularity time bases. The actual resolution  
 17843 supported by an implementation for a specific clock is obtained using the *clock\_getres()* function.  
 17844 If the actual resolution supported for a time service based on one of these clocks differs from the  
 17845 resolution supported for that clock, the implementation shall document this difference.

17846 MON The minimum allowable maximum value for `CLOCK_REALTIME` and `CLOCK_MONOTONIC`  
 17847 clocks and all absolute time services based on them is the same as that defined by the ISO C  
 17848 standard for the `time_t` type. If the maximum value supported by a time service based on one of  
 17849 these clocks differs from the maximum value supported by that clock, the implementation shall  
 17850 document this difference.

## 17851 Execution Time Monitoring

17852 CPT If `_POSIX_CPUTIME` is defined, process CPU-time clocks shall be supported in addition to the  
 17853 clocks described in [Manifest Constants](#) (on page 511).

17854 TCT If `_POSIX_THREAD_CPUTIME` is defined, thread CPU-time clocks shall be supported.

17855 CPT|TCT CPU-time clocks measure execution or CPU time, which is defined in XBD [Section 3.118](#) (on  
 17856 page 52). The mechanism used to measure execution time is described in XBD [Section 4.11](#) (on  
 17857 page 110).

17858 CPT If `_POSIX_CPUTIME` is defined, the following constant of the type `clockid_t` is defined in  
 17859 `<time.h>`:

17860 `CLOCK_PROCESS_CPUTIME_ID`

17861 When this value of the type `clockid_t` is used in a *clock()* or *timer\*()* function call, it is  
 17862 interpreted as the identifier of the CPU-time clock associated with the process making the  
 17863 function call.

17864 TCT If `_POSIX_THREAD_CPUTIME` is defined, the following constant of the type `clockid_t` is  
 17865 defined in `<time.h>`:

17866 `CLOCK_THREAD_CPUTIME_ID`

17867 When this value of the type `clockid_t` is used in a *clock()* or *timer\*()* function call, it is  
 17868 interpreted as the identifier of the CPU-time clock associated with the thread making the  
 17869 function call.

## 17870 2.9 Threads

17871 This section defines functionality to support multiple flows of control, called “threads”, within a  
 17872 process. For the definition of threads, see XBD [Section 3.404](#) (on page 99).

17873 The specific functional areas covered by threads and their scope include:

- 17874 • Thread management: the creation, control, and termination of multiple flows of control in  
 17875 the same process under the assumption of a common shared address space
- 17876 • Synchronization primitives optimized for tightly coupled operation of multiple control  
 17877 flows in a common, shared address space

## 2.9.1 Thread-Safety

All functions defined by this volume of POSIX.1-2008 shall be thread-safe, except that the following functions<sup>7</sup> need not be thread-safe.

<i>asctime()</i>	<i>ftw()</i>	<i>getutxent()</i>	<i>putenv()</i>
<i>basename()</i>	<i>getdate()</i>	<i>getutxid()</i>	<i>pututxline()</i>
<i>catgets()</i>	<i>getenv()</i>	<i>getutxline()</i>	<i>rand()</i>
<i>crypt()</i>	<i>getgrent()</i>	<i>gmtime()</i>	<i>readdir()</i>
<i>ctime()</i>	<i>getgrgid()</i>	<i>hcreate()</i>	<i>setenv()</i>
<i>dbm_clearerr()</i>	<i>getgrnam()</i>	<i>hdestroy()</i>	<i>setgrent()</i>
<i>dbm_close()</i>	<i>gethostent()</i>	<i>hsearch()</i>	<i>setkey()</i>
<i>dbm_delete()</i>	<i>getlogin()</i>	<i>inet_ntoa()</i>	<i>setlocale()</i>
<i>dbm_error()</i>	<i>getnetbyaddr()</i>	<i>l64a()</i>	<i>setpwent()</i>
<i>dbm_fetch()</i>	<i>getnetbyname()</i>	<i>lgamma()</i>	<i>setutxent()</i>
<i>dbm_firstkey()</i>	<i>getnetent()</i>	<i>lgammaf()</i>	<i>strerror()</i>
<i>dbm_nextkey()</i>	<i>getopt()</i>	<i>lgammal()</i>	<i>strsignal()</i>
<i>dbm_open()</i>	<i>getprotobyname()</i>	<i>localeconv()</i>	<i>strtok()</i>
<i>dbm_store()</i>	<i>getprotobyname()</i>	<i>localtime()</i>	<i>system()</i>
<i>dirname()</i>	<i>getprotoent()</i>	<i>lrand48()</i>	<i>ttyname()</i>
<i>derror()</i>	<i>getpwent()</i>	<i>mblen()</i>	<i>unsetenv()</i>
<i>drand48()</i>	<i>getpwnam()</i>	<i>mbtowc()</i>	<i>wctomb()</i>
<i>encrypt()</i>	<i>getpwuid()</i>	<i>mrnd48()</i>	
<i>endgrent()</i>	<i>getservbyname()</i>	<i>nftw()</i>	
<i>endpwent()</i>	<i>getservbyport()</i>	<i>nl_langinfo()</i>	
<i>endutxent()</i>	<i>getservent()</i>	<i>ptsname()</i>	

The *ctermid()* and *tmpnam()* functions need not be thread-safe if passed a NULL argument. The *mbrlen()*, *mbrtowc()*, *mbsnrtowcs()*, *mbsrtowcs()*, *wcrtomb()*, *wcsnrtombs()*, and *wcsrtombs()* functions need not be thread-safe if passed a NULL *ps* argument. The *getc\_unlocked()*, *getchar\_unlocked()*, *putc\_unlocked()*, and *putchar\_unlocked()* functions need not be thread-safe unless the invoking thread owns the (FILE \*) object accessed by the call, as is the case after a successful call to the *flockfile()* or *ftrylockfile()* functions.

Implementations shall provide internal synchronization as necessary in order to satisfy this requirement.

Since multi-threaded applications are not allowed to use the *environ* variable to access or modify any environment variable while any other thread is concurrently modifying any environment variable, any function dependent on any environment variable is not thread-safe if another thread is modifying the environment; see XSH *exec* (on page 781).

## 2.9.2 Thread IDs

Although implementations may have thread IDs that are unique in a system, applications should only assume that thread IDs are usable and unique within a single process. The effect of calling any of the functions defined in this volume of POSIX.1-2008 and passing as an argument the thread ID of a thread from another process is unspecified. The lifetime of a thread ID ends after the thread terminates if it was created with the *detachstate* attribute set to *PTHREAD\_CREATE\_DETACHED* or if *pthread\_detach()* or *pthread\_join()* has been called for that thread. A conforming implementation is free to reuse a thread ID after its lifetime has ended. If an application attempts to use a thread ID whose lifetime has ended, the behavior is undefined.

7. The functions in the table are not shaded to denote applicable options. Individual reference pages should be consulted.



17924 If a thread is detached, its thread ID is invalid for use as an argument in a call to *pthread\_detach()*  
 17925 or *pthread\_join()*.

### 17926 2.9.3 Thread Mutexes

17927 A thread that has blocked shall not prevent any unblocked thread that is eligible to use the same  
 17928 processing resources from eventually making forward progress in its execution. Eligibility for  
 17929 processing resources is determined by the scheduling policy.

17930 A thread shall become the owner of a mutex, *m*, when one of the following occurs:

- 17931 • It calls *pthread\_mutex\_lock()* with *m* as the *mutex* argument and the call returns zero or  
 17932 [EOWNERDEAD].
- 17933 • It calls *pthread\_mutex\_trylock()* with *m* as the *mutex* argument and the call returns zero or  
 17934 [EOWNERDEAD].
- 17935 • It calls *pthread\_mutex\_timedlock()* with *m* as the *mutex* argument and the call returns zero or  
 17936 [EOWNERDEAD].
- 17937 • It calls *pthread\_mutex\_setprioceiling()* with *m* as the *mutex* argument and the call returns  
 17938 [EOWNERDEAD].
- 17939 • It calls *pthread\_cond\_wait()* with *m* as the *mutex* argument and the call returns zero or  
 17940 certain error numbers (see *pthread\_cond\_timedwait()*).
- 17941 • It calls *pthread\_cond\_timedwait()* with *m* as the *mutex* argument and the call returns zero or  
 17942 certain error numbers (see *pthread\_cond\_timedwait()*).

17943 The thread shall remain the owner of *m* until one of the following occurs:

- 17944 • It executes *pthread\_mutex\_unlock()* with *m* as the *mutex* argument
- 17945 • It blocks in a call to *pthread\_cond\_wait()* with *m* as the *mutex* argument.
- 17946 • It blocks in a call to *pthread\_cond\_timedwait()* with *m* as the *mutex* argument.

17947 The implementation shall behave as if at all times there is at most one owner of any mutex.

17948 A thread that becomes the owner of a mutex is said to have “acquired” the mutex and the mutex  
 17949 is said to have become “locked”; when a thread gives up ownership of a mutex it is said to have  
 17950 “released” the mutex and the mutex is said to have become “unlocked”.

17951 A problem can occur if a process terminates while one of its threads holds a mutex lock.  
 17952 Depending on the mutex type, it might be possible for another thread to unlock the mutex and  
 17953 recover the state of the mutex. However, it is difficult to perform this recovery reliably.

17954 Robust mutexes provide a means to enable the implementation to notify other threads in the  
 17955 event of a process terminating while one of its threads holds a mutex lock. The next thread that  
 17956 acquires the mutex is notified about the termination by the return value [EOWNERDEAD] from  
 17957 the locking function. The notified thread can then attempt to recover the state protected by the  
 17958 mutex, and if successful mark the state protected by the mutex as consistent by a call to  
 17959 *pthread\_mutex\_consistent()*. If the notified thread is unable to recover the state, it can declare the  
 17960 state as not recoverable by a call to *pthread\_mutex\_unlock()* without a prior call to  
 17961 *pthread\_mutex\_consistent()*.

17962 Whether or not the state protected by a mutex can be recovered is dependent solely on the  
 17963 application using robust mutexes. The robust mutex support provided in the implementation  
 17964 provides notification only that a mutex owner has terminated while holding a lock, or that the  
 17965 state of the mutex is not recoverable.

## 2.9.4 Thread Scheduling

TPS The functionality described in this section shall be provided on implementations that support the Thread Execution Scheduling option (and the rest of this section is not further shaded for this option).

### Thread Scheduling Attributes

In support of the scheduling function, threads have attributes which are accessed through the **pthread\_attr\_t** thread creation attributes object.

The *contentionscope* attribute defines the scheduling contention scope of the thread to be either PTHREAD\_SCOPE\_PROCESS or PTHREAD\_SCOPE\_SYSTEM.

The *inheritsched* attribute specifies whether a newly created thread is to inherit the scheduling attributes of the creating thread or to have its scheduling values set according to the other scheduling attributes in the **pthread\_attr\_t** object.

The *schedpolicy* attribute defines the scheduling policy for the thread. The *schedparam* attribute defines the scheduling parameters for the thread. The interaction of threads having different policies within a process is described as part of the definition of those policies.

If the Thread Execution Scheduling option is defined, and the *schedpolicy* attribute specifies one of the priority-based policies defined under this option, the *schedparam* attribute contains the scheduling priority of the thread. A conforming implementation ensures that the priority value in *schedparam* is in the range associated with the scheduling policy when the thread attributes object is used to create a thread, or when the scheduling attributes of a thread are dynamically modified. The meaning of the priority value in *schedparam* is the same as that of *priority*.

TSP If **\_POSIX\_THREAD\_SPORADIC\_SERVER** is defined, the *schedparam* attribute supports four new members that are used for the sporadic server scheduling policy. These members are *sched\_ss\_low\_priority*, *sched\_ss\_repl\_period*, *sched\_ss\_init\_budget*, and *sched\_ss\_max\_repl*. The meaning of these attributes is the same as in the definitions that appear under [Section 2.8.4](#) (on page 506).

When a process is created, its single thread has a scheduling policy and associated attributes equal to the policy and attributes of the process. The default scheduling contention scope value is implementation-defined. The default values of other scheduling attributes are implementation-defined.

### Thread Scheduling Contention Scope

The scheduling contention scope of a thread defines the set of threads with which the thread competes for use of the processing resources. The scheduling operation selects at most one thread to execute on each processor at any point in time and the thread's scheduling attributes (for example, *priority*), whether under process scheduling contention scope or system scheduling contention scope, are the parameters used to determine the scheduling decision.

The scheduling contention scope, in the context of scheduling a mixed scope environment, affects threads as follows:

- A thread created with PTHREAD\_SCOPE\_SYSTEM scheduling contention scope contends for resources with all other threads in the same scheduling allocation domain relative to their system scheduling attributes. The system scheduling attributes of a thread created with PTHREAD\_SCOPE\_SYSTEM scheduling contention scope are the scheduling attributes with which the thread was created. The system scheduling attributes of a thread created with PTHREAD\_SCOPE\_PROCESS scheduling contention scope are the implementation-defined mapping into system attribute space of the scheduling attributes

- 18011 with which the thread was created.
- 18012 • Threads created with PTHREAD\_SCOPE\_PROCESS scheduling contention scope contend
  - 18013 directly with other threads within their process that were created with
  - 18014 PTHREAD\_SCOPE\_PROCESS scheduling contention scope. The contention is resolved
  - 18015 based on the threads' scheduling attributes and policies. It is unspecified how such threads
  - 18016 are scheduled relative to threads in other processes or threads with
  - 18017 PTHREAD\_SCOPE\_SYSTEM scheduling contention scope.
  - 18018 • Conforming implementations shall support the PTHREAD\_SCOPE\_PROCESS scheduling
  - 18019 contention scope, the PTHREAD\_SCOPE\_SYSTEM scheduling contention scope, or both.

## 18020 Scheduling Allocation Domain

18021 Implementations shall support scheduling allocation domains containing one or more

18022 processors. It should be noted that the presence of multiple processors does not automatically

18023 indicate a scheduling allocation domain size greater than one. Conforming implementations on

18024 multi-processors may map all or any subset of the CPUs to one or multiple scheduling allocation

18025 domains, and could define these scheduling allocation domains on a per-thread, per-process, or

18026 per-system basis, depending on the types of applications intended to be supported by the

18027 implementation. The scheduling allocation domain is independent of scheduling contention

18028 scope, as the scheduling contention scope merely defines the set of threads with which a thread

18029 contends for processor resources, while scheduling allocation domain defines the set of

18030 processors for which it contends. The semantics of how this contention is resolved among

18031 threads for processors is determined by the scheduling policies of the threads.

18032 The choice of scheduling allocation domain size and the level of application control over

18033 scheduling allocation domains is implementation-defined. Conforming implementations may

18034 change the size of scheduling allocation domains and the binding of threads to scheduling

18035 allocation domains at any time.

18036 For application threads with scheduling allocation domains of size equal to one, the scheduling

18037 rules defined for SCHED\_FIFO and SCHED\_RR shall be used; see [Scheduling Policies](#) (on page

18038 506). All threads with system scheduling contention scope, regardless of the processes in which

18039 they reside, compete for the processor according to their priorities. Threads with process

18040 scheduling contention scope compete only with other threads with process scheduling

18041 contention scope within their process.

18042 For application threads with scheduling allocation domains of size greater than one, the rules

18043 TSP defined for SCHED\_FIFO, SCHED\_RR, and SCHED\_SPORADIC shall be used in an

18044 implementation-defined manner. Each thread with system scheduling contention scope

18045 competes for the processors in its scheduling allocation domain in an implementation-defined

18046 manner according to its priority. Threads with process scheduling contention scope are

18047 scheduled relative to other threads within the same scheduling contention scope in the process.

18048 TSP If \_POSIX\_THREAD\_SPORADIC\_SERVER is defined, the rules defined for SCHED\_SPORADIC

18049 in [Scheduling Policies](#) (on page 506) shall be used in an implementation-defined manner for

18050 application threads whose scheduling allocation domain size is greater than one.



## Scheduling Documentation

If `_POSIX_PRIORITY_SCHEDULING` is defined, then any scheduling policies beyond `SCHED_OTHER`, `SCHED_FIFO`, `SCHED_RR`, and `SCHED_SPORADIC`, as well as the effects of the scheduling policies indicated by these other values, and the attributes required in order to support such a policy, are implementation-defined. Furthermore, the implementation shall document the effect of all processor scheduling allocation domain values supported for these policies.

## 2.9.5 Thread Cancellation

The thread cancellation mechanism allows a thread to terminate the execution of any other thread in the process in a controlled manner. The target thread (that is, the one that is being canceled) is allowed to hold cancellation requests pending in a number of ways and to perform application-specific cleanup processing when the notice of cancellation is acted upon.

Cancellation is controlled by the cancellation control functions. Each thread maintains its own cancelability state. Cancellation may only occur at cancellation points or when the thread is asynchronously cancelable.

The thread cancellation mechanism described in this section depends upon programs having set *deferred* cancelability state, which is specified as the default. Applications shall also carefully follow static lexical scoping rules in their execution behavior. For example, use of `setjmp()`, `return`, `goto`, and so on, to leave user-defined cancellation scopes without doing the necessary scope pop operation results in undefined behavior.

Use of asynchronous cancelability while holding resources which potentially need to be released may result in resource loss. Similarly, cancellation scopes may only be safely manipulated (pushed and popped) when the thread is in the *deferred* or *disabled* cancelability states.

### 2.9.5.1 Cancelability States

The cancelability state of a thread determines the action taken upon receipt of a cancellation request. The thread may control cancellation in a number of ways.

Each thread maintains its own cancelability state, which may be encoded in two bits:

1. Cancelability-Enable: When cancelability is `PTHREAD_CANCEL_DISABLE` (as defined in the Base Definitions volume of POSIX.1-2008, `<pthread.h>`), cancellation requests against the target thread are held pending. By default, cancelability is set to `PTHREAD_CANCEL_ENABLE` (as defined in `<pthread.h>`).
2. Cancelability Type: When cancelability is enabled and the cancelability type is `PTHREAD_CANCEL_ASYNCHRONOUS` (as defined in `<pthread.h>`), new or pending cancellation requests may be acted upon at any time. When cancelability is enabled and the cancelability type is `PTHREAD_CANCEL_DEFERRED` (as defined in `<pthread.h>`), cancellation requests are held pending until a cancellation point (see below) is reached. If cancelability is disabled, the setting of the cancelability type has no immediate effect as all cancellation requests are held pending; however, once cancelability is enabled again the new type is in effect. The cancelability type is `PTHREAD_CANCEL_DEFERRED` in all newly created threads including the thread in which `main()` was first invoked.

## 18091 2.9.5.2 Cancellation Points

18092 Cancellation points shall occur when a thread is executing the following functions:

18093	<i>accept()</i>	<i>nanosleep()</i>	<i>select()</i>
18094	<i>aio_suspend()</i>	<i>open()</i>	<i>sem_timedwait()</i>
18095	<i>clock_nanosleep()</i>	<i>openat()</i>	<i>sem_wait()</i>
18096	<i>close()</i>	<i>pause()</i>	<i>send()</i>
18097	<i>connect()</i>	<i>poll()</i>	<i>sendmsg()</i>
18098	<i>creat()</i>	<i>pread()</i>	<i>sendto()</i>
18099	<i>fcntl()</i> †	<i>pselect()</i>	<i>sigsuspend()</i>
18100	<i>fdatasync()</i>	<i>pthread_cond_timedwait()</i>	<i>sigtimedwait()</i>
18101	<i>fsync()</i>	<i>pthread_cond_wait()</i>	<i>sigwait()</i>
18102	<i>getmsg()</i>	<i>pthread_join()</i>	<i>sigwaitinfo()</i>
18103	<i>getpmsg()</i>	<i>pthread_testcancel()</i>	<i>sleep()</i>
18104	<i>lockf()</i> ††	<i>putmsg()</i>	<i>tcdrain()</i>
18105	<i>mq_receive()</i>	<i>putpmsg()</i>	<i>wait()</i>
18106	<i>mq_send()</i>	<i>pwrite()</i>	<i>waitid()</i>
18107	<i>mq_timedreceive()</i>	<i>read()</i>	<i>waitpid()</i>
18108	<i>mq_timedsend()</i>	<i>readv()</i>	<i>write()</i>
18109	<i>msgrcv()</i>	<i>recv()</i>	<i>writev()</i>
18110	<i>msgsnd()</i>	<i>recvfrom()</i>	
18111	<i>msync()</i>	<i>recvmsg()</i>	

---

18112 † When the *cmd* argument is F\_SETLK.

18113 †† When the *function* argument is F\_LOCK.

18114 A cancellation point may also occur when a thread is executing the following functions:

18115	<i>access()</i>	<i>fstatat()</i>	<i>mkstemp()</i>
18116	<i>asctime_r()</i>	<i>ftell()</i>	<i>mktime()</i>
18117	<i>catclose()</i>	<i>ftello()</i>	<i>opendir()</i>
18118	<i>catopen()</i>	<i>futimens()</i>	<i>openlog()</i>
18119	<i>chmod()</i>	<i>fwprintf()</i>	<i>pathconf()</i>
18120	<i>chown()</i>	<i>fwrite()</i>	<i>perror()</i>
18121	<i>closedir()</i>	<i>fwscanf()</i>	<i>popen()</i>
18122	<i>closelog()</i>	<i>getaddrinfo()</i>	<i>posix_fadvise()</i>
18123	<i>ctermid()</i>	<i>getc()</i>	<i>posix_fallocate()</i>
18124	<i>ctime_r()</i>	<i>getc_unlocked()</i>	<i>posix_madvise()</i>
18125	<i>dlclose()</i>	<i>getchar()</i>	<i>posix_openpt()</i>
18126	<i>dlopen()</i>	<i>getchar_unlocked()</i>	<i>posix_spawn()</i>
18127	<i>dprintf()</i>	<i>getcwd()</i>	<i>posix_spawnnp()</i>
18128	<i>endhostent()</i>	<i>getdelim()</i>	<i>posix_trace_clear()</i>
18129	<i>endnetent()</i>	<i>getgrgid_r()</i>	<i>posix_trace_close()</i>
18130	<i>endprotoent()</i>	<i>getgrnam_r()</i>	<i>posix_trace_create()</i>
18131	<i>endservent()</i>	<i>gethostid()</i>	<i>posix_trace_create_withlog()</i>
18132	<i>faccessat()</i>	<i>gethostname()</i>	<i>posix_trace_eventtypelist_getnext_id()</i>
18133	<i>fchmod()</i>	<i>getline()</i>	<i>posix_trace_eventtypelist_rewind()</i>
18134	<i>fchmodat()</i>	<i>getlogin_r()</i>	<i>posix_trace_flush()</i>
18135	<i>fchown()</i>	<i>getnameinfo()</i>	<i>posix_trace_get_attr()</i>
18136	<i>fchownat()</i>	<i>getpwnam_r()</i>	<i>posix_trace_get_filter()</i>
18137	<i>fclose()</i>	<i>getpwuid_r()</i>	<i>posix_trace_get_status()</i>
18138	<i>fcntl()</i> †	<i>gets()</i>	<i>posix_trace_getnext_event()</i>
18139	<i>fflush()</i>	<i>getwc()</i>	<i>posix_trace_open()</i>
18140	<i>fgetc()</i>	<i>getwchar()</i>	<i>posix_trace_rewind()</i>
18141	<i>fgetpos()</i>	<i>glob()</i>	<i>posix_trace_set_filter()</i>
18142	<i>fgets()</i>	<i>iconv_close()</i>	<i>posix_trace_shutdown()</i>
18143	<i>fgetwc()</i>	<i>iconv_open()</i>	<i>posix_trace_timedgetnext_event()</i>
18144	<i>fgetws()</i>	<i>ioctl()</i>	<i>posix_typed_mem_open()</i>
18145	<i>fntmsg()</i>	<i>link()</i>	<i>printf()</i>
18146	<i>fopen()</i>	<i>linkat()</i>	<i>psiginfo()</i>
18147	<i>fpathconf()</i>	<i>lio_listio()</i>	<i>psignal()</i>
18148	<i>fprintf()</i>	<i>localtime_r()</i>	<i>pthread_rwlock_rdlock()</i>
18149	<i>fputc()</i>	<i>lockf()</i>	<i>pthread_rwlock_timedrdlock()</i>
18150	<i>fputs()</i>	<i>lseek()</i>	<i>pthread_rwlock_timedwrlock()</i>
18151	<i>fputwc()</i>	<i>lstat()</i>	<i>pthread_rwlock_wrlock()</i>
18152	<i>fputws()</i>	<i>mkdir()</i>	<i>putc()</i>
18153	<i>fread()</i>	<i>mkdirat()</i>	<i>putc_unlocked()</i>
18154	<i>freopen()</i>	<i>mkdtemp()</i>	<i>putchar()</i>
18155	<i>fscanf()</i>	<i>mkfifo()</i>	<i>putchar_unlocked()</i>
18156	<i>fseek()</i>	<i>mkfifoat()</i>	<i>puts()</i>
18157	<i>fseeko()</i>	<i>mknod()</i>	<i>putwc()</i>
18158	<i>fsetpos()</i>	<i>mknodat()</i>	<i>putwchar()</i>
18159	<i>fstat()</i>		<i>readdir_r()</i>

18160	<code>readlink()</code>	<code>sigpause()</code>	<code>ungetwc()</code>
18161	<code>readlinkat()</code>	<code>stat()</code>	<code>unlink()</code>
18162	<code>remove()</code>	<code>strerror_l()</code>	<code>unlinkat()</code>
18163	<code>rename()</code>	<code>strerror_r()</code>	<code>utime()</code>
18164	<code>renameat()</code>	<code>strftime()</code>	<code>utimensat()</code>
18165	<code>rewind()</code>	<code>strftime_l()</code>	<code>utimes()</code>
18166	<code>rewinddir()</code>	<code>symlink()</code>	<code>vdprintf()</code>
18167	<code>scandir()</code>	<code>symlinkat()</code>	<code>vfprintf()</code>
18168	<code>scanf()</code>	<code>sync()</code>	<code>vfwprintf()</code>
18169	<code>seekdir()</code>	<code>syslog()</code>	<code>vprintf()</code>
18170	<code>semop()</code>	<code>tmpfile()</code>	<code>vwprintf()</code>
18171	<code>sethostent()</code>	<code>tmpnam()</code>	<code>wcsftime()</code>
18172	<code>setnetent()</code>	<code>ttynam_r()</code>	<code>wordexp()</code>
18173	<code>setprotoent()</code>	<code>tzset()</code>	<code>wprintf()</code>
18174	<code>setservent()</code>	<code>ungetc()</code>	<code>wscanf()</code>

In addition, a cancellation point may occur when a thread is executing any function that this standard does not require to be thread-safe but the implementation documents as being thread-safe. If a thread is cancelled while executing a non-thread-safe function, the behavior is undefined.

An implementation shall not introduce cancellation points into any other functions specified in this volume of POSIX.1-2008.

The side-effects of acting upon a cancellation request while suspended during a call of a function are the same as the side-effects that may be seen in a single-threaded program when a call to a function is interrupted by a signal and the given function returns [EINTR]. Any such side-effects occur before any cancellation cleanup handlers are called. For functions that are explicitly required not to return when interrupted (for example, `pclose()`), if a thread is canceled while executing the function, the behavior is undefined.

Whenever a thread has cancelability enabled and a cancellation request has been made with that thread as the target, and the thread then calls any function that is a cancellation point (such as `pthread_testcancel()` or `read()`), the cancellation request shall be acted upon before the function returns. If a thread has cancelability enabled and a cancellation request is made with the thread as a target while the thread is suspended at a cancellation point, the thread shall be awakened and the cancellation request shall be acted upon. It is unspecified whether the cancellation request is acted upon or whether the cancellation request remains pending and the thread resumes normal execution if:

- The thread is suspended at a cancellation point and the event for which it is waiting occurs
- A specified timeout expired

before the cancellation request is acted upon.

### 2.9.5.3 Thread Cancellation Cleanup Handlers

Each thread maintains a list of cancellation cleanup handlers. The programmer uses the `pthread_cleanup_push()` and `pthread_cleanup_pop()` functions to place routines on and remove routines from this list.

When a cancellation request is acted upon, or when a thread calls `pthread_exit()`, the thread first disables cancellation by setting its cancelability state to `PTHREAD_CANCEL_DISABLE` and its

---

† For any value of the *cmd* argument.

cancelability type to `PTHREAD_CANCEL_DEFERRED`. The cancelability state shall remain set to `PTHREAD_CANCEL_DISABLE` until the thread has terminated. The behavior is undefined if a cancellation cleanup handler or thread-specific data destructor routine changes the cancelability state to `PTHREAD_CANCEL_ENABLE`.

The routines in the thread's list of cancellation cleanup handlers are invoked one by one in LIFO sequence; that is, the last routine pushed onto the list (Last In) is the first to be invoked (First Out). When the cancellation cleanup handler for a scope is invoked, the storage for that scope remains valid. If the last cancellation cleanup handler returns, thread-specific data destructors (if any) associated with thread-specific data keys for which the thread has non-NULL values will be run, in unspecified order, as described for *pthread\_key\_create()*.

After all cancellation cleanup handlers and thread-specific data destructors have returned, thread execution is terminated. If the thread has terminated because of a call to *pthread\_exit()*, the *value\_ptr* argument is made available to any threads joining with the target. If the thread has terminated by acting on a cancellation request, a status of `PTHREAD_CANCELED` is made available to any threads joining with the target. The symbolic constant `PTHREAD_CANCELED` expands to a constant expression of type (**void** \*) whose value matches no pointer to an object in memory nor the value `NULL`.

A side-effect of acting upon a cancellation request while in a condition variable wait is that the mutex is re-acquired before calling the first cancellation cleanup handler. In addition, the thread is no longer considered to be waiting for the condition and the thread shall not have consumed any pending condition signals on the condition.

A cancellation cleanup handler cannot exit via *longjmp()* or *siglongjmp()*.

#### 2.9.5.4 Async-Cancel Safety

The *pthread\_cancel()*, *pthread\_setcancelstate()*, and *pthread\_setcanceltype()* functions are defined to be async-cancel safe.

No other functions in this volume of POSIX.1-2008 are required to be async-cancel-safe.

If a thread has asynchronous cancellation enabled and is cancelled during execution of a function that is not async-cancel-safe, the behavior is undefined. +

If a thread has deferred cancellation enabled, a signal-catching function is called in that thread during execution of a function that is not async-cancel-safe, and the signal-catching function calls any function that is a cancellation point while a cancellation is pending for the thread, the behavior is undefined. +

### 2.9.6 Thread Read-Write Locks

Multiple readers, single writer (read-write) locks allow many threads to have simultaneous read-only access to data while allowing only one thread to have exclusive write access at any given time. They are typically used to protect data that is read more frequently than it is changed.

One or more readers acquire read access to the resource by performing a read lock operation on the associated read-write lock. A writer acquires exclusive write access by performing a write lock operation. Basically, all readers exclude any writers and a writer excludes all readers and any other writers.

A thread that has blocked on a read-write lock (for example, has not yet returned from a *pthread\_rwlock\_rdlock()* or *pthread\_rwlock\_wrlock()* call) shall not prevent any unblocked thread

that is eligible to use the same processing resources from eventually making forward progress in its execution. Eligibility for processing resources shall be determined by the scheduling policy.

Read-write locks can be used to synchronize threads in the current process and other processes if they are allocated in memory that is writable and shared among the cooperating processes and have been initialized for this behavior.

### 2.9.7 Thread Interactions with Regular File Operations

All of the following functions shall be atomic with respect to each other in the effects specified in POSIX.1-2008 when they operate on regular files or symbolic links:

<i>chmod()</i>	<i>fchownat()</i>	<i>lseek()</i>	<i>readv()</i>	<i>unlink()</i>
<i>chown()</i>	<i>fcntl()</i>	<i>lstat()</i>	<i>pwrite()</i>	<i>unlinkat()</i>
<i>close()</i>	<i>fstat()</i>	<i>open()</i>	<i>rename()</i>	<i>utime()</i>
<i>creat()</i>	<i>fstatat()</i>	<i>openat()</i>	<i>renameat()</i>	<i>utimensat()</i>
<i>dup2()</i>	<i>ftruncate()</i>	<i>pread()</i>	<i>stat()</i>	<i>utimes()</i>
<i>fchmod()</i>	<i>lchown()</i>	<i>read()</i>	<i>symlink()</i>	<i>write()</i>
<i>fchmodat()</i>	<i>link()</i>	<i>readlink()</i>	<i>symlinkat()</i>	<i>writew()</i>
<i>fchown()</i>	<i>linkat()</i>	<i>readlinkat()</i>	<i>truncate()</i>	

If two threads each call one of these functions, each call shall either see all of the specified effects of the other call, or none of them. The requirement on the *close()* function shall also apply whenever a file descriptor is successfully closed, however caused (for example, as a consequence of calling *close()*, calling *dup2()*, or of process termination).

### 2.9.8 Use of Application-Managed Thread Stacks

An “application-managed thread stack” is a region of memory allocated by the application—for example, memory returned by the *malloc()* or *mmap()* functions—and designated as a stack through the act of passing the address and size of the stack, respectively, as the *stackaddr* and *stacksize* arguments to *pthread\_attr\_setstack()*. Application-managed stacks allow the application to precisely control the placement and size of a stack.

The application grants to the implementation permanent ownership of and control over the application-managed stack when the attributes object in which the *stack* or *stackaddr* attribute has been set is used, either by presenting that attribute’s object as the *attr* argument in a call to *pthread\_create()* that completes successfully, or by storing a pointer to the attributes object in the *sigev\_notify\_attributes* member of a **struct sigevent** and passing that **struct sigevent** to a function accepting such argument that completes successfully. The application may thereafter utilize the memory within the stack only within the normal context of stack usage within or properly synchronized with a thread that has been scheduled by the implementation with stack pointer value(s) that are within the range of that stack. In particular, the region of memory cannot be freed, nor can it be later specified as the stack for another thread.

When specifying an attributes object with an application-managed stack through the *sigev\_notify\_attributes* member of a **struct sigevent**, the results are undefined if the requested signal is generated multiple times (as for a repeating timer).

Until an attributes object in which the *stack* or *stackaddr* attribute has been set is used, the application retains ownership of and control over the memory allocated to the stack. It may free or reuse the memory as long as it either deletes the attributes object, or before using the attributes object replaces the stack by making an additional call to *pthread\_attr\_setstack()*, that was used originally to designate the stack. There is no mechanism to retract the reference to an



18292 application-managed stack by an existing attributes object.

18293 Once an attributes object with an application-managed stack has been used, that attributes object  
 18294 cannot be used again by a subsequent call to *pthread\_create()* or any function accepting a **struct**  
 18295 **sigevent** with *sigev\_notify\_attributes* containing a pointer to the attributes object, without  
 18296 designating an unused application-managed stack by making an additional call to  
 18297 *pthread\_attr\_setstack()*.

## 18298 2.9.9 Synchronization Object Copies and Alternative Mappings

18299 TSH For barriers, condition variables, mutexes, and read-write locks, if the process-shared attribute +  
 18300 is set to `PTHREAD_PROCESS_PRIVATE`, only the synchronization object at the address used to +  
 18301 initialize it can be used for performing synchronization. The effect of referring to another +  
 18302 TSH mapping of the same object when locking, unlocking, or destroying the object is undefined. If +  
 18303 the process-shared attribute is set to `PTHREAD_PROCESS_SHARED`, only the synchronization +  
 18304 object itself can be used for performing synchronization; however, it need not be referenced at +  
 18305 the address used to initialize it (that is, another mapping of the same object can be used). The +  
 18306 effect of referring to a copy of the object when locking, unlocking, or destroying it is undefined. +  
 18307 For spin locks, the above requirements shall apply as if spin locks have a process-shared +  
 18308 attribute that is set from the *pshared* argument to *pthread\_spin\_init()*. For semaphores, the above +  
 18309 requirements shall apply as if semaphores have a process-shared attribute that is set to +  
 18310 `PTHREAD_PROCESS_PRIVATE` if the *pshared* argument to *sem\_init()* is zero and set to +  
 18311 `PTHREAD_PROCESS_SHARED` if *pshared* is non-zero.

## 18312 2.10 Sockets

18313 A socket is an endpoint for communication using the facilities described in this section. A socket  
 18314 is created with a specific socket type, described in [Section 2.10.6](#) (on page 524), and is associated  
 18315 with a specific protocol, detailed in [Section 2.10.3](#) (on page 524). A socket is accessed via a file  
 18316 descriptor obtained when the socket is created.

### 18317 2.10.1 Address Families

18318 All network protocols are associated with a specific address family. An address family provides  
 18319 basic services to the protocol implementation to allow it to function within a specific network  
 18320 environment. These services may include packet fragmentation and reassembly, routing,  
 18321 addressing, and basic transport. An address family is normally comprised of a number of  
 18322 protocols, one per socket type. Each protocol is characterized by an abstract socket type. It is not  
 18323 required that an address family support all socket types. An address family may contain  
 18324 multiple protocols supporting the same socket abstraction.

18325 [Section 2.10.17](#) (on page 531), [Section 2.10.19](#) (on page 532), and [Section 2.10.20](#) (on page 532),  
 18326 respectively, describe the use of sockets for local UNIX connections, for Internet protocols based  
 18327 on IPv4, and for Internet protocols based on IPv6.

## 18328 2.10.2 Addressing

18329 An address family defines the format of a socket address. All network addresses are described  
 18330 using a general structure, called a **sockaddr**, as defined in the Base Definitions volume of  
 18331 POSIX.1-2008, **<sys/socket.h>**. However, each address family imposes finer and more specific  
 18332 structure, generally defining a structure with fields specific to the address family. The field  
 18333 *sa\_family* in the **sockaddr** structure contains the address family identifier, specifying the format  
 18334 of the *sa\_data* area. The size of the *sa\_data* area is unspecified.

## 18335 2.10.3 Protocols

18336 A protocol supports one of the socket abstractions detailed in [Section 2.10.6](#). Selecting a protocol  
 18337 involves specifying the address family, socket type, and protocol number to the *socket()* function.  
 18338 Certain semantics of the basic socket abstractions are protocol-specific. All protocols are  
 18339 expected to support the basic model for their particular socket type, but may, in addition,  
 18340 provide non-standard facilities or extensions to a mechanism.

## 18341 2.10.4 Routing

18342 Sockets provides packet routing facilities. A routing information database is maintained, which  
 18343 is used in selecting the appropriate network interface when transmitting packets.

## 18344 2.10.5 Interfaces

18345 Each network interface in a system corresponds to a path through which messages can be sent  
 18346 and received. A network interface usually has a hardware device associated with it, though  
 18347 certain interfaces such as the loopback interface, do not.

## 18348 2.10.6 Socket Types

18349 A socket is created with a specific type, which defines the communication semantics and which  
 18350 allows the selection of an appropriate communication protocol. Four types are defined:  
 18351 RS SOCK\_DGRAM, **SOCK\_RAW**, SOCK\_SEQPACKET, and SOCK\_STREAM. Implementations  
 18352 may specify additional socket types.

18353 The SOCK\_STREAM socket type provides reliable, sequenced, full-duplex octet streams  
 18354 between the socket and a peer to which the socket is connected. A socket of type  
 18355 SOCK\_STREAM must be in a connected state before any data may be sent or received. Record  
 18356 boundaries are not maintained; data sent on a stream socket using output operations of one size  
 18357 may be received using input operations of smaller or larger sizes without loss of data. Data may  
 18358 be buffered; successful return from an output function does not imply that the data has been  
 18359 delivered to the peer or even transmitted from the local system. If data cannot be successfully  
 18360 transmitted within a given time then the connection is considered broken, and subsequent  
 18361 operations shall fail. A SIGPIPE signal is raised if a thread attempts to send data on a broken  
 18362 stream (one that is no longer connected), except that the signal is suppressed if the  
 18363 MSG\_NOSIGNAL flag is used in calls to *send()*, *sendto()*, and *sendmsg()*. Support for an out-of-  
 18364 band data transmission facility is protocol-specific.

18365 The SOCK\_SEQPACKET socket type is similar to the SOCK\_STREAM type, and is also  
 18366 connection-oriented. The only difference between these types is that record boundaries are  
 18367 maintained using the SOCK\_SEQPACKET type. A record can be sent using one or more output



operations and received using one or more input operations, but a single operation never transfers parts of more than one record. Record boundaries are visible to the receiver via the MSG\_EOR flag in the received message flags returned by the *recvmsg()* function. It is protocol-specific whether a maximum record size is imposed.

The SOCK\_DGRAM socket type supports connectionless data transfer which is not necessarily acknowledged or reliable. Datagrams may be sent to the address specified (possibly multicast or broadcast) in each output operation, and incoming datagrams may be received from multiple sources. The source address of each datagram is available when receiving the datagram. An application may also pre-specify a peer address, in which case calls to output functions that do not specify a peer address shall send to the pre-specified peer. If a peer has been specified, only datagrams from that peer shall be received. A datagram must be sent in a single output operation, and must be received in a single input operation. The maximum size of a datagram is protocol-specific; with some protocols, the limit is implementation-defined. Output datagrams may be buffered within the system; thus, a successful return from an output function does not guarantee that a datagram is actually sent or received. However, implementations should attempt to detect any errors possible before the return of an output function, reporting any error by an unsuccessful return value.

The SOCK\_RAW socket type is similar to the SOCK\_DGRAM type. It differs in that it is normally used with communication providers that underlie those used for the other socket types. For this reason, the creation of a socket with type SOCK\_RAW shall require appropriate privileges. The format of datagrams sent and received with this socket type generally include specific protocol headers, and the formats are protocol-specific and implementation-defined.

## 2.10.7 Socket I/O Mode

The I/O mode of a socket is described by the O\_NONBLOCK file status flag which pertains to the open file description for the socket. This flag is initially off when a socket is created, but may be set and cleared by the use of the F\_SETFL command of the *fcntl()* function.

When the O\_NONBLOCK flag is set, certain functions that would normally block until they are complete shall return immediately.

The *bind()* function initiates an address assignment and shall return without blocking when O\_NONBLOCK is set; if the socket address cannot be assigned immediately, *bind()* shall return the [EINPROGRESS] error to indicate that the assignment was initiated successfully, but that it has not yet completed.

The *connect()* function initiates a connection and shall return without blocking when O\_NONBLOCK is set; it shall return the error [EINPROGRESS] to indicate that the connection was initiated successfully, but that it has not yet completed.

Data transfer operations (the *read()*, *write()*, *send()*, and *recv()* functions) shall complete immediately, transfer only as much as is available, and then return without blocking, or return an error indicating that no transfer could be made without blocking.

## 2.10.8 Socket Owner

The owner of a socket is unset when a socket is created. The owner may be set to a process ID or process group ID using the `F_SETOWN` command of the `fcntl()` function.

## 2.10.9 Socket Queue Limits

The transmit and receive queue sizes for a socket are set when the socket is created. The default sizes used are both protocol-specific and implementation-defined. The sizes may be changed using the `setsockopt()` function.

## 2.10.10 Pending Error

Errors may occur asynchronously, and be reported to the socket in response to input from the network protocol. The socket stores the pending error to be reported to a user of the socket at the next opportunity. The error is returned in response to a subsequent `send()`, `recv()`, or `getsockopt()` operation on the socket, and the pending error is then cleared.

## 2.10.11 Socket Receive Queue

A socket has a receive queue that buffers data when it is received by the system until it is removed by a receive call. Depending on the type of the socket and the communication provider, the receive queue may also contain ancillary data such as the addressing and other protocol data associated with the normal data in the queue, and may contain out-of-band or expedited data. The limit on the queue size includes any normal, out-of-band data, datagram source addresses, and ancillary data in the queue. The description in this section applies to all sockets, even though some elements cannot be present in some instances.

The contents of a receive buffer are logically structured as a series of data segments with associated ancillary data and other information. A data segment may contain normal data or out-of-band data, but never both. A data segment may complete a record if the protocol supports records (always true for types `SOCK_SEQPACKET` and `SOCK_DGRAM`). A record may be stored as more than one segment; the complete record might never be present in the receive buffer at one time, as a portion might already have been returned to the application, and another portion might not yet have been received from the communications provider. A data segment may contain ancillary protocol data, which is logically associated with the segment. Ancillary data is received as if it were queued along with the first normal data octet in the segment (if any). A segment may contain ancillary data only, with no normal or out-of-band data. For the purposes of this section, a datagram is considered to be a data segment that terminates a record, and that includes a source address as a special type of ancillary data. Data segments are placed into the queue as data is delivered to the socket by the protocol. Normal data segments are placed at the end of the queue as they are delivered. If a new segment contains the same type of data as the preceding segment and includes no ancillary data, and if the preceding segment does not terminate a record, the segments are logically merged into a single segment.

The receive queue is logically terminated if an end-of-file indication has been received or a connection has been terminated. A segment shall be considered to be terminated if another segment follows it in the queue, if the segment completes a record, or if an end-of-file or other connection termination has been reported. The last segment in the receive queue shall also be considered to be terminated while the socket has a pending error to be reported.

18448 A receive operation shall never return data or ancillary data from more than one segment.

## 18449 2.10.12 Socket Out-of-Band Data State

18450 The handling of received out-of-band data is protocol-specific. Out-of-band data may be placed  
 18451 in the socket receive queue, either at the end of the queue or before all normal data in the queue.  
 18452 In this case, out-of-band data is returned to an application program by a normal receive call.  
 18453 Out-of-band data may also be queued separately rather than being placed in the socket receive  
 18454 queue, in which case it shall be returned only in response to a receive call that requests out-of-  
 18455 band data. It is protocol-specific whether an out-of-band data mark is placed in the receive  
 18456 queue to demarcate data preceding the out-of-band data and following the out-of-band data. An  
 18457 out-of-band data mark is logically an empty data segment that cannot be merged with other  
 18458 segments in the queue. An out-of-band data mark is never returned in response to an input  
 18459 operation. The *socketmark()* function can be used to test whether an out-of-band data mark is the  
 18460 first element in the queue. If an out-of-band data mark is the first element in the queue when an  
 18461 input function is called without the MSG\_PEEK option, the mark is removed from the queue  
 18462 and the following data (if any) is processed as if the mark had not been present.

## 18463 2.10.13 Connection Indication Queue

18464 Sockets that are used to accept incoming connections maintain a queue of outstanding  
 18465 connection indications. This queue is a list of connections that are awaiting acceptance by the  
 18466 application; see *listen()*.

## 18467 2.10.14 Signals

18468 One category of event at the socket interface is the generation of signals. These signals report  
 18469 protocol events or process errors relating to the state of the socket. The generation or delivery of  
 18470 a signal does not change the state of the socket, although the generation of the signal may have  
 18471 been caused by a state change.

18472 The SIGPIPE signal shall be sent to a thread that attempts to send data on a socket that is no  
 18473 longer able to send (one that is no longer connected), except that the signal is suppressed if the  
 18474 MSG\_NOSIGNAL flag is used in calls to *send()*, *sendto()*, and *sendmsg()*. Regardless of whether  
 18475 the generation of the signal is suppressed, the send operation shall fail with the [EPIPE] error.

18476 If a socket has an owner, the SIGURG signal is sent to the owner of the socket when it is notified  
 18477 of expedited or out-of-band data. The socket state at this time is protocol-dependent, and the  
 18478 status of the socket is specified in [Section 2.10.17](#) (on page 531), [Section 2.10.19](#) (on page 532),  
 18479 and [Section 2.10.20](#) (on page 532). Depending on the protocol, the expedited data may or may  
 18480 not have arrived at the time of signal generation.

## 18481 2.10.15 Asynchronous Errors

18482 If any of the following conditions occur asynchronously for a socket, the corresponding value  
 18483 listed below shall become the pending error for the socket:

18484 [ECONNABORTED]

18485 The connection was aborted locally.

**[ECONNREFUSED]**

For a connection-mode socket attempting a non-blocking connection, the attempt to connect was forcefully rejected. For a connectionless-mode socket, an attempt to deliver a datagram was forcefully rejected.

**[ECONNRESET]**

The peer has aborted the connection.

**[EHOSTDOWN]**

The destination host has been determined to be down or disconnected.

**[EHOSTUNREACH]**

The destination host is not reachable.

**[EMSGSIZE]**

For a connectionless-mode socket, the size of a previously sent datagram prevented delivery.

**[ENETDOWN]**

The local network connection is not operational.

**[ENETRESET]**

The connection was aborted by the network.

**[ENETUNREACH]**

The destination network is not reachable.

**2.10.16 Use of Options**

There are a number of socket options which either specialize the behavior of a socket or provide useful information. These options may be set at different protocol levels and are always present at the uppermost “socket” level.

Socket options are manipulated by two functions, *getsockopt()* and *setsockopt()*. These functions allow an application program to customize the behavior and characteristics of a socket to provide the desired effect.

All of the options have default values. The type and meaning of these values is defined by the protocol level to which they apply. Instead of using the default values, an application program may choose to customize one or more of the options. However, in the bulk of cases, the default values are sufficient for the application.

Some of the options are used to enable or disable certain behavior within the protocol modules (for example, turn on debugging) while others may be used to set protocol-specific information (for example, IP time-to-live on all the application’s outgoing packets). As each of the options is introduced, its effect on the underlying protocol modules is described.

Table 2-1 shows the value for the socket level.

**Table 2-1** Value of Level for Socket Options

Name	Description
SOL_SOCKET	Options are intended for the sockets level.

Table 2-2 (on page 529) lists those options present at the socket level; that is, when the *level* parameter of the *getsockopt()* or *setsockopt()* function is SOL\_SOCKET, the types of the option value parameters associated with each option, and a brief synopsis of the meaning of the option

value parameter. Unless otherwise noted, each may be examined with *getsockopt()* and set with *setsockopt()* on all types of socket. Options at other protocol levels vary in format and name.

Table 2-2 Socket-Level Options

Option	Parameter Type	Parameter Meaning
SO_ACCEPTCONN	int	Non-zero indicates that socket listening is enabled ( <i>getsockopt()</i> only).
SO_BROADCAST	int	Non-zero requests permission to transmit broadcast datagrams (SOCK_DGRAM sockets only).
SO_DEBUG	int	Non-zero requests debugging in underlying protocol modules.
SO_DONTROUTE	int	Non-zero requests bypass of normal routing; route based on destination address only.
SO_ERROR	int	Requests and clears pending error information on the socket ( <i>getsockopt()</i> only).
SO_KEEPALIVE	int	Non-zero requests periodic transmission of keepalive messages (protocol-specific).
SO_LINGER	struct linger	Specify actions to be taken for queued, unsent data on <i>close()</i> : linger on/off and linger time in seconds.
SO_OOINLINE	int	Non-zero requests that out-of-band data be placed into normal data input queue as received.
SO_RCVBUF	int	Size of receive buffer (in bytes).
SO_RCVLOWAT	int	Minimum amount of data to return to application for input operations (in bytes).
SO_RCVTIMEO	struct timeval	Timeout value for a socket receive operation.
SO_REUSEADDR	int	Non-zero requests reuse of local addresses in <i>bind()</i> (protocol-specific).
SO_SNDBUF	int	Size of send buffer (in bytes).
SO_SNDLOWAT	int	Minimum amount of data to send for output operations (in bytes).
SO_SNDTIMEO	struct timeval	Timeout value for a socket send operation.
SO_TYPE	int	Identify socket type ( <i>getsockopt()</i> only).

The SO\_ACCEPTCONN option is used only on *getsockopt()*. When this option is specified, *getsockopt()* shall report whether socket listening is enabled for the socket. A value of zero shall indicate that socket listening is disabled; non-zero that it is enabled. SO\_ACCEPTCONN has no default value.

The SO\_BROADCAST option requests permission to send broadcast datagrams on the socket. Support for SO\_BROADCAST is protocol-specific. The default for SO\_BROADCAST is that the ability to send broadcast datagrams on a socket is disabled.

The SO\_DEBUG option enables debugging in the underlying protocol modules. This can be useful for tracing the behavior of the underlying protocol modules during normal system operation. The semantics of the debug reports are implementation-defined. The default value for SO\_DEBUG is for debugging to be turned off.

The SO\_DONTROUTE option requests that outgoing messages bypass the standard routing facilities. The destination must be on a directly-connected network, and messages are directed to the appropriate network interface according to the destination address. It is protocol-specific whether this option has any effect and how the outgoing network interface is chosen. Support



for this option with each protocol is implementation-defined.

The `SO_ERROR` option is used only on `getsockopt()`. When this option is specified, `getsockopt()` shall return any pending error on the socket and clear the error status. It shall return a value of 0 if there is no pending error. `SO_ERROR` may be used to check for asynchronous errors on connected connectionless-mode sockets or for other types of asynchronous errors. `SO_ERROR` has no default value.

The `SO_KEEPALIVE` option enables the periodic transmission of messages on a connected socket. The behavior of this option is protocol-specific. On a connection-mode socket for which a connection has been established, if `SO_KEEPALIVE` is enabled and the connected socket fails to respond to the keep-alive messages, the connection shall be broken. The default value for `SO_KEEPALIVE` is zero, specifying that this capability is turned off.

The `SO_LINGER` option controls the action of the interface when unsent messages are queued on a socket and a `close()` is performed. The details of this option are protocol-specific. If `SO_LINGER` is enabled, the system shall block the calling thread during `close()` until it can transmit the data or until the end of the interval indicated by the `l_linger` member, whichever comes first. If `SO_LINGER` is not specified, and `close()` is issued, the system handles the call in a way that allows the calling thread to continue as quickly as possible. The default value for `SO_LINGER` is zero, or off, for the `l_onoff` element of the option value and zero seconds for the linger time specified by the `l_linger` element.

The `SO_OOBINLINE` option is valid only on protocols that support out-of-band data. The `SO_OOBINLINE` option requests that out-of-band data be placed in the normal data input queue as received; it is then accessible using the `read()` or `recv()` functions without the `MSG_OOB` flag set. The default for `SO_OOBINLINE` is off; that is, for out-of-band data not to be placed in the normal data input queue.

The `SO_RCVBUF` option requests that the buffer space allocated for receive operations on this socket be set to the value, in bytes, of the option value. Applications may wish to increase buffer size for high volume connections, or may decrease buffer size to limit the possible backlog of incoming data. The default value for the `SO_RCVBUF` option value is implementation-defined, and may vary by protocol.

The `SO_RCVLOWAT` option sets the minimum number of bytes to process for socket input operations. In general, receive calls block until any (non-zero) amount of data is received, then return the smaller of the amount available or the amount requested. The default value for `SO_RCVLOWAT` is 1, and does not affect the general case. If `SO_RCVLOWAT` is set to a larger value, blocking receive calls normally wait until they have received the smaller of the low water mark value or the requested amount. Receive calls may still return less than the low water mark if an error occurs, a signal is caught, or the type of data next in the receive queue is different from that returned (for example, out-of-band data). As mentioned previously, the default value for `SO_RCVLOWAT` is 1 byte. It is implementation-defined whether the `SO_RCVLOWAT` option can be set.

The `SO_RCVTIMEO` option is an option to set a timeout value for input operations. It accepts a **timeval** structure with the number of seconds and microseconds specifying the limit on how long to wait for an input operation to complete. If a receive operation has blocked for this much time without receiving additional data, it shall return with a partial count or `errno` shall be set to `[EAGAIN]` or `[EWOULDBLOCK]` if no data were received. The default for this option is the value zero, which indicates that a receive operation will not time out. It is implementation-defined whether the `SO_RCVTIMEO` option can be set.

The `SO_REUSEADDR` option indicates that the rules used in validating addresses supplied in a `bind()` should allow reuse of local addresses. Operation of this option is protocol-specific. The default value for `SO_REUSEADDR` is off; that is, reuse of local addresses is not permitted.

The `SO_SNDBUF` option requests that the buffer space allocated for send operations on this socket be set to the value, in bytes, of the option value. The default value for the `SO_SNDBUF` option value is implementation-defined, and may vary by protocol.

The `SO_SNDLOWAT` option sets the minimum number of bytes to process for socket output operations. Most output operations process all of the data supplied by the call, delivering data to the protocol for transmission and blocking as necessary for flow control. Non-blocking output operations process as much data as permitted subject to flow control without blocking, but process no data if flow control does not allow the smaller of the send low water mark value or the entire request to be processed. A `select()` operation testing the ability to write to a socket shall return true only if the send low water mark could be processed. The default value for `SO_SNDLOWAT` is implementation-defined and protocol-specific. It is implementation-defined whether the `SO_SNDLOWAT` option can be set.

The `SO_SNDTIMEO` option is an option to set a timeout value for the amount of time that an output function shall block because flow control prevents data from being sent. As noted in [Table 2-2](#) (on page 529), the option value is a **timeval** structure with the number of seconds and microseconds specifying the limit on how long to wait for an output operation to complete. If a send operation has blocked for this much time, it shall return with a partial count or *errno* set to `[EAGAIN]` or `[EWOULDBLOCK]` if no data were sent. The default for this option is the value zero, which indicates that a send operation will not time out. It is implementation-defined whether the `SO_SNDTIMEO` option can be set.

The `SO_TYPE` option is used only on `getsockopt()`. When this option is specified, `getsockopt()` shall return the type of the socket (for example, `SOCK_STREAM`). This option is useful to servers that inherit sockets on start-up. `SO_TYPE` has no default value.

## 2.10.17 Use of Sockets for Local UNIX Connections

Support for UNIX domain sockets is mandatory.

UNIX domain sockets provide process-to-process communication in a single system.

### 2.10.17.1 Headers

The symbolic constant `AF_UNIX` defined in the `<sys/socket.h>` header is used to identify the UNIX domain address family. The `<sys/un.h>` header contains other definitions used in connection with UNIX domain sockets. See [XBD Chapter 13](#) (on page 219).

The **sockaddr\_storage** structure defined in `<sys/socket.h>` shall be large enough to accommodate a **sockaddr\_un** structure (see the `<sys/un.h>` header defined in [XBD Chapter 13](#), on page 219) and shall be aligned at an appropriate boundary so that pointers to it can be cast as pointers to **sockaddr\_un** structures and used to access the fields of those structures without alignment problems. When a **sockaddr\_storage** structure is cast as a **sockaddr\_un** structure, the *ss\_family* field maps onto the *sun\_family* field.

18660 **2.10.18 Use of Sockets over Internet Protocols**

18661 When a socket is created in the Internet family with a protocol value of zero, the implementation  
18662 shall use the protocol listed below for the type of socket created.

18663 SOCK\_STREAM IPPROTO\_TCP.

18664 SOCK\_DGRAM IPPROTO\_UDP.

18665 RS SOCK\_RAW IPPROTO\_RAW.

18666 SOCK\_SEQPACKET Unspecified.

18667 RS A raw interface to IP is available by creating an Internet socket of type SOCK\_RAW. The default  
18668 protocol for type SOCK\_RAW shall be identified in the IP header with the value  
18669 IPPROTO\_RAW. Applications should not use the default protocol when creating a socket with  
18670 type SOCK\_RAW, but should identify a specific protocol by value. The ICMP control protocol is  
18671 accessible from a raw socket by specifying a value of IPPROTO\_ICMP for protocol.

18672 **2.10.19 Use of Sockets over Internet Protocols Based on IPv4**

18673 Support for sockets over Internet protocols based on IPv4 is mandatory.

18674 **2.10.19.1 Headers**

18675 The symbolic constant AF\_INET defined in the `<sys/socket.h>` header is used to identify the  
18676 IPv4 Internet address family. The `<netinet/in.h>` header contains other definitions used in  
18677 connection with IPv4 Internet sockets. See XBD [Chapter 13](#) (on page 219).

18678 The `sockaddr_storage` structure defined in `<sys/socket.h>` shall be large enough to  
18679 accommodate a `sockaddr_in` structure (see the `<netinet/in.h>` header defined in XBD [Chapter](#)  
18680 [13](#), on page 219) and shall be aligned at an appropriate boundary so that pointers to it can be  
18681 cast as pointers to `sockaddr_in` structures and used to access the fields of those structures  
18682 without alignment problems. When a `sockaddr_storage` structure is cast as a `sockaddr_in`  
18683 structure, the `ss_family` field maps onto the `sin_family` field.

18684 **2.10.20 Use of Sockets over Internet Protocols Based on IPv6**

18685 IP6 This section describes extensions to support sockets over Internet protocols based on IPv6. The  
18686 functionality described in this section shall be provided on implementations that support the  
18687 IPV6 option (and the rest of this section is not further shaded for this option).

18688 To enable smooth transition from IPv4 to IPv6, the features defined in this section may, in certain  
18689 circumstances, also be used in connection with IPv4; see [Section 2.10.20.2](#) (on page 533).

18690 **2.10.20.1 Addressing**

18691 IPv6 overcomes the addressing limitations of earlier versions by using 128-bit addresses instead  
18692 of 32-bit addresses. The IPv6 address architecture is described in RFC 2373.

18693 There are three kinds of IPv6 address:

18694 Unicast

18695 Identifies a single interface.

18696 A unicast address can be global, link-local (designed for use on a single link), or site-local



(designed for systems not connected to the Internet). Link-local and site-local addresses need not be globally unique.

#### Anycast

Identifies a set of interfaces such that a packet sent to the address can be delivered to any member of the set.

An anycast address is similar to a unicast address; the nodes to which an anycast address is assigned must be explicitly configured to know that it is an anycast address.

#### Multicast

Identifies a set of interfaces such that a packet sent to the address should be delivered to every member of the set.

An application can send multicast datagrams by simply specifying an IPv6 multicast address in the *address* argument of *sendto()*. To receive multicast datagrams, an application must join the multicast group (using *setsockopt()* with `IPV6_JOIN_GROUP`) and must bind to the socket the UDP port on which datagrams will be received. Some applications should also bind the multicast group address to the socket, to prevent other datagrams destined to that port from being delivered to the socket.

A multicast address can be global, node-local, link-local, site-local, or organization-local.

The following special IPv6 addresses are defined:

#### Unspecified

An address that is not assigned to any interface and is used to indicate the absence of an address.

#### Loopback

A unicast address that is not assigned to any interface and can be used by a node to send packets to itself.

Two sets of IPv6 addresses are defined to correspond to IPv4 addresses:

#### IPv4-compatible addresses

These are assigned to nodes that support IPv6 and can be used when traffic is “tunneled” through IPv4.

#### IPv4-mapped addresses

These are used to represent IPv4 addresses in IPv6 address format; see [Section 2.10.20.2](#).

Note that the unspecified address and the loopback address must not be treated as IPv4-compatible addresses.

### 2.10.20.2 Compatibility with IPv4

The API provides the ability for IPv6 applications to interoperate with applications using IPv4, by using IPv4-mapped IPv6 addresses. These addresses can be generated automatically by the *getaddrinfo()* function when the specified host has only IPv4 addresses.

Applications can use `AF_INET6` sockets to open TCP connections to IPv4 nodes, or send UDP packets to IPv4 nodes, by simply encoding the destination’s IPv4 address as an IPv4-mapped IPv6 address, and passing that address, within a `sockaddr_in6` structure, in the *connect()*, *sendto()*, or *sendmsg()* function. When applications use `AF_INET6` sockets to accept TCP connections from IPv4 nodes, or receive UDP packets from IPv4 nodes, the system shall return the peer’s address to the application in the *accept()*, *recvfrom()*, *recvmsg()*, or *getpeername()* function using a `sockaddr_in6` structure encoded this way. If a node has an IPv4 address, then the implementation shall allow applications to communicate using that address via an

AF\_INET6 socket. In such a case, the address will be represented at the API by the corresponding IPv4-mapped IPv6 address. Also, the implementation may allow an AF\_INET6 socket bound to **in6addr\_any** to receive inbound connections and packets destined to one of the node's IPv4 addresses.

An application can use AF\_INET6 sockets to bind to a node's IPv4 address by specifying the address as an IPv4-mapped IPv6 address in a **sockaddr\_in6** structure in the *bind()* function. For an AF\_INET6 socket bound to a node's IPv4 address, the system shall return the address in the *getsockname()* function as an IPv4-mapped IPv6 address in a **sockaddr\_in6** structure.

#### 2.10.20.3 Interface Identification

Each local interface is assigned a unique positive integer as a numeric index. Indexes start at 1; zero is not used. There may be gaps so that there is no current interface for a particular positive index. Each interface also has a unique implementation-defined name.

#### 2.10.20.4 Options

The following options apply at the IPPROTO\_IPV6 level:

##### IPV6\_JOIN\_GROUP

When set via *setsockopt()*, it joins the application to a multicast group on an interface (identified by its index) and addressed by a given multicast address, enabling packets sent to that address to be read via the socket. If the interface index is specified as zero, the system selects the interface (for example, by looking up the address in a routing table and using the resulting interface).

An attempt to read this option using *getsockopt()* shall result in an [EOPNOTSUPP] error.

The parameter type of this option is a pointer to an **ipv6\_mreq** structure.

##### IPV6\_LEAVE\_GROUP

When set via *setsockopt()*, it removes the application from the multicast group on an interface (identified by its index) and addressed by a given multicast address.

An attempt to read this option using *getsockopt()* shall result in an [EOPNOTSUPP] error.

The parameter type of this option is a pointer to an **ipv6\_mreq** structure.

##### IPV6\_MULTICAST\_HOPS

The value of this option is the hop limit for outgoing multicast IPv6 packets sent via the socket. Its possible values are the same as those of IPV6\_UNICAST\_HOPS. If the IPV6\_MULTICAST\_HOPS option is not set, a value of 1 is assumed. This option can be set via *setsockopt()* and read via *getsockopt()*.

The parameter type of this option is a pointer to an **int**. (Default value: 1)

##### IPV6\_MULTICAST\_IF

The index of the interface to be used for outgoing multicast packets. It can be set via *setsockopt()* and read via *getsockopt()*. If the interface index is specified as zero, the system selects the interface (for example, by looking up the address in a routing table and using the resulting interface).

The parameter type of this option is a pointer to an **unsigned int**. (Default value: 0)

##### IPV6\_MULTICAST\_LOOP

This option controls whether outgoing multicast packets should be delivered back to the local application when the sending interface is itself a member of the destination multicast

18783 group. If it is set to 1 they are delivered. If it is set to 0 they are not. Other values result in an  
 18784 [EINVAL] error. This option can be set via *setsockopt()* and read via *getsockopt()*.

18785 The parameter type of this option is a pointer to an **unsigned int** which is used as a Boolean  
 18786 value. (Default value: 1)

18787 IPV6\_UNICAST\_HOPS

18788 The value of this option is the hop limit for outgoing unicast IPv6 packets sent via the  
 18789 socket. If the option is not set, or is set to -1, the system selects a default value. Attempts to  
 18790 set a value less than -1 or greater than 255 shall result in an [EINVAL] error. This option can  
 18791 be set via *setsockopt()* and read via *getsockopt()*.

18792 The parameter type of this option is a pointer to an **int**. (Default value: Unspecified)

18793 IPV6\_V6ONLY

18794 This socket option restricts AF\_INET6 sockets to IPv6 communications only. AF\_INET6  
 18795 sockets may be used for both IPv4 and IPv6 communications. Some applications may want  
 18796 to restrict their use of an AF\_INET6 socket to IPv6 communications only. For these  
 18797 applications, the IPV6\_V6ONLY socket option is defined. When this option is turned on, the  
 18798 socket can be used to send and receive IPv6 packets only. This is an IPPROTO\_IPv6-level  
 18799 option.

18800 The parameter type of this option is a pointer to an **int** which is used as a Boolean value.  
 18801 (Default value: 0)

18802 An [EOPNOTSUPP] error shall result if IPV6\_JOIN\_GROUP or IPV6\_LEAVE\_GROUP is used  
 18803 with *getsockopt()*.

#### 18804 2.10.20.5 Headers

18805 The symbolic constant AF\_INET6 is defined in the `<sys/socket.h>` header to identify the IPv6  
 18806 Internet address family. See XBD [Chapter 13](#) (on page 219).

18807 The **sockaddr\_storage** structure defined in `<sys/socket.h>` shall be large enough to  
 18808 accommodate a **sockaddr\_in6** structure (see the `<netinet/in.h>` header defined in XBD [Chapter](#)  
 18809 [13](#), on page 219) and shall be aligned at an appropriate boundary so that pointers to it can be  
 18810 cast as pointers to **sockaddr\_in6** structures and used to access the fields of those structures  
 18811 without alignment problems. When a **sockaddr\_storage** structure is cast as a **sockaddr\_in6**  
 18812 structure, the *ss\_family* field maps onto the *sin6\_family* field.

18813 The `<netinet/in.h>`, `<arpa/inet.h>`, and `<netdb.h>` headers contain other definitions used in  
 18814 connection with IPv6 Internet sockets; see XBD [Chapter 13](#) (on page 219).

## 2.11 Tracing

OB TRC

This section describes extensions to support tracing of user applications. The functionality described in this section is dependent on support of the Trace option (and the rest of this section is not further shaded for this option).

The tracing facilities defined in POSIX.1-2008 allow a process to select a set of trace event types, to activate a trace stream of the selected trace events as they occur in the flow of execution, and to retrieve the recorded trace events.

The tracing operation relies on three logically different components: the traced process, the controller process, and the analyzer process. During the execution of the traced process, when a trace point is reached, a trace event is recorded into the trace streams created for that process in which the associated trace event type identifier is not being filtered out. The controller process controls the operation of recording the trace events into the trace stream. It shall be able to:

- Initialize the attributes of a trace stream
- Create the trace stream (for a specified traced process) using those attributes
- Start and stop tracing for the trace stream
- Filter the type of trace events to be recorded, if the Trace Event Filter option is supported
- Shut a trace stream down

These operations can be done for an active trace stream. The analyzer process retrieves the traced events either at runtime, when the trace stream has not yet been shut down, but is still recording trace events; or after opening a trace log that had been previously recorded and shut down. These three logically different operations can be performed by the same process, or can be distributed into different processes.

A trace stream identifier can be created by a call to `posix_trace_create()`, `posix_trace_create_withlog()`, or `posix_trace_open()`. The `posix_trace_create()` and `posix_trace_create_withlog()` functions should be used by a controller process. The `posix_trace_open()` should be used by an analyzer process.

The tracing functions can serve different purposes. One purpose is debugging the possibly pre-instrumented code, while another is post-mortem fault analysis. These two potential uses differ in that the first requires pre-filtering capabilities to avoid overwhelming the trace stream and permits focusing on expected information; while the second needs comprehensive trace capabilities in order to be able to record all types of information.

The events to be traced belong to two classes:

1. User trace events (generated by the application instrumentation)
2. System trace events (generated by the operating system)

The trace interface defines several system trace event types associated with control of and operation of the trace stream. This small set of system trace events includes the minimum required to interpret correctly the trace event information present in the stream. Other desirable system trace events for some particular application profile may be implemented and are encouraged; for example, process and thread scheduling, signal occurrence, and so on.

Each traced process shall have a mapping of the trace event names to trace event type identifiers that have been defined for that process. Each active trace stream shall have a mapping that incorporates all the trace event type identifiers predefined by the trace system plus all the mappings of trace event names to trace event type identifiers of the processes that are being traced into that trace stream. These mappings are defined from the instrumented application by calling the `posix_trace_eventid_open()` function and from the controller process by calling the

18860 *posix\_trace\_trid\_eventid\_open()* function. For a pre-recorded trace stream, the list of trace event  
18861 types is obtained from the pre-recorded trace log.

18862 The last data modification and file status change timestamps of a file associated with an active  
18863 trace stream shall be marked for update every time any of the tracing operations modifies that  
18864 file.

18865 The last data access timestamp of a file associated with a trace stream shall be marked for  
18866 update every time any of the tracing operations causes data to be read from that file.

18867 Results are undefined if the application performs any operation on a file descriptor associated  
18868 with an active or pre-recorded trace stream until *posix\_trace\_shutdown()* or *posix\_trace\_close()* is  
18869 called for that trace stream. Results are also undefined if the analyzer process and the traced  
18870 process do not share the same programming environment (see *c99*, Programming Environments  
18871 in the Shell and Utilities volume of POSIX.1-2008).

18872 The main purpose of this option is to define a complete set of functions and concepts that allow  
18873 a conforming application to be traced from creation to termination, whatever its realtime  
18874 constraints and properties.

## 2.11.1 Tracing Data Definitions

### 2.11.1.1 Structures

The **<trace.h>** header shall define the *posix\_trace\_status\_info* and *posix\_trace\_event\_info* structures described below. Implementations may add extensions to these structures.

#### **posix\_trace\_status\_info** Structure

To facilitate control of a trace stream, information about the current state of an active trace stream can be obtained dynamically. This structure is returned by a call to the *posix\_trace\_get\_status()* function.

The **posix\_trace\_status\_info** structure defined in **<trace.h>** shall contain at least the following members:

Member Type	Member Name	Description
<b>int</b>	<i>posix_stream_status</i>	The operating mode of the trace stream.
<b>int</b>	<i>posix_stream_full_status</i>	The full status of the trace stream.
<b>int</b>	<i>posix_stream_overrun_status</i>	Indicates whether trace events were lost in the trace stream.

If the Trace Log option is supported in addition to the Trace option, the **posix\_trace\_status\_info** structure defined in **<trace.h>** shall contain at least the following additional members:

Member Type	Member Name	Description
<b>int</b>	<i>posix_stream_flush_status</i>	Indicates whether a flush is in progress.
<b>int</b>	<i>posix_stream_flush_error</i>	Indicates whether any error occurred during the last flush operation.
<b>int</b>	<i>posix_log_overrun_status</i>	Indicates whether trace events were lost in the trace log.
<b>int</b>	<i>posix_log_full_status</i>	The full status of the trace log.

The *posix\_stream\_status* member indicates the operating mode of the trace stream and shall have one of the following values defined by manifest constants in the **<trace.h>** header:

#### POSIX\_TRACE\_RUNNING

Tracing is in progress; that is, the trace stream is accepting trace events.

#### POSIX\_TRACE\_SUSPENDED

The trace stream is not accepting trace events. The tracing operation has not yet started or has stopped, either following a *posix\_trace\_stop()* function call or because the trace resources are exhausted.

The *posix\_stream\_full\_status* member indicates the full status of the trace stream, and it shall have one of the following values defined by manifest constants in the **<trace.h>** header:

#### POSIX\_TRACE\_FULL

The space in the trace stream for trace events is exhausted.

#### POSIX\_TRACE\_NOT\_FULL

There is still space available in the trace stream.

The combination of the *posix\_stream\_status* and *posix\_stream\_full\_status* members also indicates the actual status of the stream. The status shall be interpreted as follows:



18915 POSIX\_TRACE\_RUNNING and POSIX\_TRACE\_NOT\_FULL

18916 This status combination indicates that tracing is in progress, and there is space available for  
18917 recording more trace events.

18918 POSIX\_TRACE\_RUNNING and POSIX\_TRACE\_FULL

18919 This status combination indicates that tracing is in progress and that the trace stream is full  
18920 of trace events. This status combination cannot occur unless the *stream-full-policy* is set to  
18921 POSIX\_TRACE\_LOOP. The trace stream contains trace events recorded during a moving  
18922 time window of prior trace events, and some older trace events may have been overwritten  
18923 and thus lost.

18924 POSIX\_TRACE\_SUSPENDED and POSIX\_TRACE\_NOT\_FULL

18925 This status combination indicates that tracing has not yet been started, has been stopped by  
18926 the *posix\_trace\_stop()* function, or has been cleared by the *posix\_trace\_clear()* function.

18927 POSIX\_TRACE\_SUSPENDED and POSIX\_TRACE\_FULL

18928 This status combination indicates that tracing has been stopped by the implementation  
18929 because the *stream-full-policy* attribute was POSIX\_TRACE\_UNTIL\_FULL and trace  
18930 resources were exhausted, or that the trace stream was stopped by the function  
18931 *posix\_trace\_stop()* at a time when trace resources were exhausted.

18932 The *posix\_stream\_overrun\_status* member indicates whether trace events were lost in the trace  
18933 stream, and shall have one of the following values defined by manifest constants in the  
18934 **<trace.h>** header:

18935 POSIX\_TRACE\_OVERRUN

18936 At least one trace event was lost and thus was not recorded in the trace stream.

18937 POSIX\_TRACE\_NO\_OVERRUN

18938 No trace events were lost.

18939 When the corresponding trace stream is created, the *posix\_stream\_overrun\_status* member shall be  
18940 set to POSIX\_TRACE\_NO\_OVERRUN.

18941 Whenever an overrun occurs, the *posix\_stream\_overrun\_status* member shall be set to  
18942 POSIX\_TRACE\_OVERRUN.

18943 An overrun occurs when:

- 18944 • The policy is POSIX\_TRACE\_LOOP and a recorded trace event is overwritten.
- 18945 • The policy is POSIX\_TRACE\_UNTIL\_FULL and the trace stream is full when a trace event  
18946 is generated.
- 18947 • If the Trace Log option is supported, the policy is POSIX\_TRACE\_FLUSH and at least one  
18948 trace event is lost while flushing the trace stream to the trace log.

18949 The *posix\_stream\_overrun\_status* member is reset to zero after its value is read.

18950 If the Trace Log option is supported in addition to the Trace option, the *posix\_stream\_flush\_status*,  
18951 *posix\_stream\_flush\_error*, *posix\_log\_overrun\_status*, and *posix\_log\_full\_status* members are defined  
18952 as follows; otherwise, they are undefined.

18953 The *posix\_stream\_flush\_status* member indicates whether a flush operation is being performed  
18954 and shall have one of the following values defined by manifest constants in the header  
18955 **<trace.h>**:

18956 POSIX\_TRACE\_FLUSHING

18957 The trace stream is currently being flushed to the trace log.

## POSIX\_TRACE\_NOT\_FLUSHING

No flush operation is in progress.

The *posix\_stream\_flush\_status* member shall be set to POSIX\_TRACE\_FLUSHING if a flush operation is in progress either due to a call to the *posix\_trace\_flush()* function (explicit or caused by a trace stream shutdown operation) or because the trace stream has become full with the *stream-full-policy* attribute set to POSIX\_TRACE\_FLUSH. The *posix\_stream\_flush\_status* member shall be set to POSIX\_TRACE\_NOT\_FLUSHING if no flush operation is in progress.

The *posix\_stream\_flush\_error* member shall be set to zero if no error occurred during flushing. If an error occurred during a previous flushing operation, the *posix\_stream\_flush\_error* member shall be set to the value of the first error that occurred. If more than one error occurs while flushing, error values after the first shall be discarded. The *posix\_stream\_flush\_error* member is reset to zero after its value is read.

The *posix\_log\_overnrun\_status* member indicates whether trace events were lost in the trace log, and shall have one of the following values defined by manifest constants in the **<trace.h>** header:

## POSIX\_TRACE\_OVERRUN

At least one trace event was lost.

## POSIX\_TRACE\_NO\_OVERRUN

No trace events were lost.

When the corresponding trace stream is created, the *posix\_log\_overnrun\_status* member shall be set to POSIX\_TRACE\_NO\_OVERRUN. Whenever an overrun occurs, this status shall be set to POSIX\_TRACE\_OVERRUN. The *posix\_log\_overnrun\_status* member is reset to zero after its value is read.

The *posix\_log\_full\_status* member indicates the full status of the trace log, and it shall have one of the following values defined by manifest constants in the **<trace.h>** header:

## POSIX\_TRACE\_FULL

The space in the trace log is exhausted.

## POSIX\_TRACE\_NOT\_FULL

There is still space available in the trace log.

The *posix\_log\_full\_status* member is only meaningful if the *log-full-policy* attribute is either POSIX\_TRACE\_UNTIL\_FULL or POSIX\_TRACE\_LOOP.

For an active trace stream without log, that is created by the *posix\_trace\_create()* function, the *posix\_log\_overnrun\_status* member shall be set to POSIX\_TRACE\_NO\_OVERRUN and the *posix\_log\_full\_status* member shall be set to POSIX\_TRACE\_NOT\_FULL.

## posix\_trace\_event\_info Structure

The trace event structure **posix\_trace\_event\_info** contains the information for one recorded trace event. This structure is returned by the set of functions *posix\_trace\_getnext\_event()*, *posix\_trace\_timedgetnext\_event()*, and *posix\_trace\_trygetnext\_event()*.

The **posix\_trace\_event\_info** structure defined in **<trace.h>** shall contain at least the following members:



Member Type	Member Name	Description
<b>trace_event_id_t</b>	<i>posix_event_id</i>	Trace event type identification.
<b>pid_t</b>	<i>posix_pid</i>	Process ID of the process that generated the trace event.
<b>void *</b>	<i>posix_prog_address</i>	Address at which the trace point was invoked.
<b>int</b>	<i>posix_truncation_status</i>	Status about the truncation of the data associated with this trace event.
<b>struct timespec</b>	<i>posix_timestamp</i>	Time at which the trace event was generated.

In addition, the **posix\_trace\_event\_info** structure defined in **<trace.h>** shall contain the following additional member:

Member Type	Member Name	Description
<b>pthread_t</b>	<i>posix_thread_id</i>	Thread ID of the thread that generated the trace event.

The *posix\_event\_id* member represents the identification of the trace event type and its value is not directly defined by the user. This identification is returned by a call to one of the following functions: *posix\_trace\_trid\_eventid\_open()*, *posix\_trace\_eventtypelist\_getnext\_id()*, or *posix\_trace\_eventid\_open()*. The name of the trace event type can be obtained by calling *posix\_trace\_eventid\_get\_name()*.

The *posix\_pid* is the process identifier of the traced process which generated the trace event. If the *posix\_event\_id* member is one of the implementation-defined system trace events and that trace event is not associated with any process, the *posix\_pid* member shall be set to zero.

For a user trace event, the *posix\_prog\_address* member is the process mapped address of the point at which the associated call to the *posix\_trace\_event()* function was made. For a system trace event, if the trace event is caused by a system service explicitly called by the application, the *posix\_prog\_address* member shall be the address of the process at the point where the call to that system service was made.

The *posix\_truncation\_status* member defines whether the data associated with a trace event has been truncated at the time the trace event was generated, or at the time the trace event was read from the trace stream, or (if the Trace Log option is supported) from the trace log (see the *event* argument from the *posix\_trace\_getnext\_event()* function). The *posix\_truncation\_status* member shall have one of the following values defined by manifest constants in the **<trace.h>** header:

**POSIX\_TRACE\_NOT\_TRUNCATED**

All the traced data is available.

**POSIX\_TRACE\_TRUNCATED\_RECORD**

Data was truncated at the time the trace event was generated.

**POSIX\_TRACE\_TRUNCATED\_READ**

Data was truncated at the time the trace event was read from a trace stream or a trace log because the reader's buffer was too small. This truncation status overrides the **POSIX\_TRACE\_TRUNCATED\_RECORD** status.

The *posix\_timestamp* member shall be the time at which the trace event was generated. The clock used is implementation-defined, but the resolution of this clock can be retrieved by a call to the *posix\_trace\_attr\_getclockres()* function.

The *posix\_thread\_id* member is the identifier of the thread that generated the trace event. If the *posix\_event\_id* member is one of the implementation-defined system trace events and that trace

event is not associated with any thread, the *posix\_thread\_id* member shall be set to zero.

#### 2.11.1.2 Trace Stream Attributes

Trace streams have attributes that compose the **posix\_trace\_attr\_t** trace stream attributes object. This object shall contain at least the following attributes:

- The *generation-version* attribute identifies the origin and version of the trace system.
- The *trace-name* attribute is a character string defined by the trace controller, and that identifies the trace stream.
- The *creation-time* attribute represents the time of the creation of the trace stream.
- The *clock-resolution* attribute defines the clock resolution of the clock used to generate timestamps.
- The *stream-min-size* attribute defines the minimum size in bytes of the trace stream strictly reserved for the trace events.
- The *stream-full-policy* attribute defines the policy followed when the trace stream is full; its value is `POSIX_TRACE_LOOP`, `POSIX_TRACE_UNTIL_FULL`, or `POSIX_TRACE_FLUSH`.
- The *max-data-size* attribute defines the maximum record size in bytes of a trace event.

In addition, if the Trace option and the Trace Inherit option are both supported, the **posix\_trace\_attr\_t** trace stream creation attributes object shall contain at least the following attributes:

- The *inheritance* attribute specifies whether a newly created trace stream will inherit tracing in its parent's process trace stream. It is either `POSIX_TRACE_INHERITED` or `POSIX_TRACE_CLOSE_FOR_CHILD`.

In addition, if the Trace option and the Trace Log option are both supported, the **posix\_trace\_attr\_t** trace stream creation attributes object shall contain at least the following attribute:

- If the file type corresponding to the trace log supports the `POSIX_TRACE_LOOP` or the `POSIX_TRACE_UNTIL_FULL` policies, the *log-max-size* attribute defines the maximum size in bytes of the trace log associated with an active trace stream. Other stream data—for example, trace attribute values—shall not be included in this size.
- The *log-full-policy* attribute defines the policy of a trace log associated with an active trace stream to be `POSIX_TRACE_LOOP`, `POSIX_TRACE_UNTIL_FULL`, or `POSIX_TRACE_APPEND`.

## 2.11.2 Trace Event Type Definitions

### 2.11.2.1 System Trace Event Type Definitions

The following system trace event types, defined in the **<trace.h>** header, track the invocation of the trace operations:

- `POSIX_TRACE_START` shall be associated with a trace start operation.

- POSIX\_TRACE\_STOP shall be associated with a trace stop operation.
- If the Trace Event Filter option is supported, POSIX\_TRACE\_FILTER shall be associated with a trace event type filter change operation.

The following system trace event types, defined in the **<trace.h>** header, report operational trace events:

- POSIX\_TRACE\_OVERFLOW shall mark the beginning of a trace overflow condition.
- POSIX\_TRACE\_RESUME shall mark the end of a trace overflow condition.
- If the Trace Log option is supported, POSIX\_TRACE\_FLUSH\_START shall mark the beginning of a flush operation.
- If the Trace Log option is supported, POSIX\_TRACE\_FLUSH\_STOP shall mark the end of a flush operation.
- If an implementation-defined trace error condition is reported, it shall be marked POSIX\_TRACE\_ERROR.

The interpretation of a trace stream or a trace log by a trace analyzer process relies on the information recorded for each trace event, and also on system trace events that indicate the invocation of trace control operations and trace system operational trace events.

The POSIX\_TRACE\_START and POSIX\_TRACE\_STOP trace events specify the time windows during which the trace stream is running.

- The POSIX\_TRACE\_STOP trace event with an associated data that is equal to zero indicates a call of the function *posix\_trace\_stop()*.
- The POSIX\_TRACE\_STOP trace event with an associated data that is different from zero indicates an automatic stop of the trace stream (see the definition of the *posix\_trace\_attr\_getstreamfullpolicy()* function in *posix\_trace\_attr\_getinherited()*).

The POSIX\_TRACE\_FILTER trace event indicates that a trace event type filter value changed while the trace stream was running.

The POSIX\_TRACE\_ERROR serves to inform the analyzer process that an implementation-defined internal error of the trace system occurred.

The POSIX\_TRACE\_OVERFLOW trace event shall be reported with a timestamp equal to the timestamp of the first trace event overwritten. This is an indication that some generated trace events have been lost.

The POSIX\_TRACE\_RESUME trace event shall be reported with a timestamp equal to the timestamp of the first valid trace event reported after the overflow condition ends and shall be reported before this first valid trace event. This is an indication that the trace system is reliably recording trace events after an overflow condition.

Each of these trace event types shall be defined by a constant trace event name and a **trace\_event\_id\_t** constant; trace event data is associated with some of these trace events.

If the Trace option is supported and the Trace Event Filter option and the Trace Log option are not supported, the following predefined system trace events in [Table 2-3](#) (on page 544) shall be defined:

19119

**Table 2-3** Trace Option: System Trace Events

19120

19121

19122

19123

19124

19125

19126

19127

19128

Event Name	Constant	Associated Data
		Data Type
posix_trace_error	POSIX_TRACE_ERROR	error <b>int</b>
posix_trace_start	POSIX_TRACE_START	None.
posix_trace_stop	POSIX_TRACE_STOP	auto <b>int</b>
posix_trace_overflow	POSIX_TRACE_OVERFLOW	None.
posix_trace_resume	POSIX_TRACE_RESUME	None.

19129

19130

19131

If the Trace option and the Trace Event Filter option are both supported, and if the Trace Log option is not supported, the following predefined system trace events in [Table 2-4](#) shall be defined:

19132

**Table 2-4** Trace and Trace Event Filter Options: System Trace Events

19133

19134

19135

19136

19137

19138

19139

19140

19141

19142

19143

19144

19145

Event Name	Constant	Associated Data
		Data Type
posix_trace_error	POSIX_TRACE_ERROR	error <b>int</b>
posix_trace_start	POSIX_TRACE_START	event_filter <b>trace_event_set_t</b>
posix_trace_stop	POSIX_TRACE_STOP	auto <b>int</b>
posix_trace_filter	POSIX_TRACE_FILTER	old_event_filter new_event_filter <b>trace_event_set_t</b>
posix_trace_overflow	POSIX_TRACE_OVERFLOW	None.
posix_trace_resume	POSIX_TRACE_RESUME	None.

19146

19147

19148

If the Trace option and the Trace Log option are both supported, and if the Trace Event Filter option is not supported, the following predefined system trace events in [Table 2-5](#) (on page 545) shall be defined:

**Table 2-5** Trace and Trace Log Options: System Trace Events

Event Name	Constant	Associated Data
		Data Type
posix_trace_error	POSIX_TRACE_ERROR	error <b>int</b>
posix_trace_start	POSIX_TRACE_START	None.
posix_trace_stop	POSIX_TRACE_STOP	auto <b>int</b>
posix_trace_overflow	POSIX_TRACE_OVERFLOW	None.
posix_trace_resume	POSIX_TRACE_RESUME	None.
posix_trace_flush_start	POSIX_TRACE_FLUSH_START	None.
posix_trace_flush_stop	POSIX_TRACE_FLUSH_STOP	None.

If the Trace option, the Trace Event Filter option, and the Trace Log option are all supported, the following predefined system trace events in [Table 2-6](#) shall be defined:

**Table 2-6** Trace, Trace Log, and Trace Event Filter Options: System Trace Events

Event Name	Constant	Associated Data
		Data Type
posix_trace_error	POSIX_TRACE_ERROR	error <b>int</b>
posix_trace_start	POSIX_TRACE_START	event_filter <b>trace_event_set_t</b>
posix_trace_stop	POSIX_TRACE_STOP	auto <b>int</b>
posix_trace_filter	POSIX_TRACE_FILTER	old_event_filter new_event_filter <b>trace_event_set_t</b>
posix_trace_overflow	POSIX_TRACE_OVERFLOW	None.
posix_trace_resume	POSIX_TRACE_RESUME	None.
posix_trace_flush_start	POSIX_TRACE_FLUSH_START	None.
posix_trace_flush_stop	POSIX_TRACE_FLUSH_STOP	None.

#### 2.11.2.2 User Trace Event Type Definitions

The user trace event POSIX\_TRACE\_UNNAMED\_USEREVENT is defined in the **<trace.h>** header. If the limit of per-process user trace event names represented by {TRACE\_USER\_EVENT\_MAX} has already been reached, this predefined user event shall be returned when the application tries to register more events than allowed. The data associated with this trace event is application-defined.

The following predefined user trace event in [Table 2-7](#) (on page 546) shall be defined:

19186

**Table 2-7** Trace Option: User Trace Event

19187

19188

Event Name	Constant
posix_trace_unnamed_userevent	POSIX_TRACE_UNNAMED_USEREVENT

19189

**2.11.3 Trace Functions**

19190

19191

19192

19193

19194

19195

The trace interface is built and structured to improve portability through use of trace data of opaque type. The object-oriented approach for the manipulation of trace attributes and trace event type identifiers requires definition of many constructor and selector functions which operate on these opaque types. Also, the trace interface must support several different tracing roles. To facilitate reading the trace interface, the trace functions are grouped into small functional sets supporting the three different roles:

19196

19197

- A trace controller process requires functions to set up and customize all the resources needed to run a trace stream, including:

19198

- Attribute initialization and destruction (*posix\_trace\_attr\_init()*)

19199

- Identification information manipulation (*posix\_trace\_attr\_getgenversion()*)

19200

- Trace system behavior modification (*posix\_trace\_attr\_getinherited()*)

19201

- Trace stream and trace log size set (*posix\_trace\_attr\_getmaxusereventsize()*)

19202

- Trace stream creation, flush, and shutdown (*posix\_trace\_create()*)

19203

- Trace stream and trace log clear (*posix\_trace\_clear()*)

19204

- Trace event type identifier manipulation (*posix\_trace\_trid\_eventid\_open()*)

19205

- Trace event type identifier list exploration (*posix\_trace\_eventtypelist\_getnext\_id()*)

19206

- Trace event type set manipulation (*posix\_trace\_eventset\_empty()*)

19207

- Trace event type filter set (*posix\_trace\_set\_filter()*)

19208

- Trace stream start and stop (*posix\_trace\_start()*)

19209

- Trace stream information and status read (*posix\_trace\_get\_attr()*)

19210

- A traced process requires functions to instrument trace points:

19211

- Trace event type identifiers definition and trace points insertion (*posix\_trace\_event()*)

19212

19213

- A trace analyzer process requires functions to retrieve information from a trace stream and trace log:

19214

- Identification information read (*posix\_trace\_attr\_getgenversion()*)

19215

- Trace system behavior information read (*posix\_trace\_attr\_getinherited()*)

19216

- Trace stream and trace log size get (*posix\_trace\_attr\_getmaxusereventsize()*)

19217

- Trace event type identifier manipulation (*posix\_trace\_trid\_eventid\_open()*)

19218

- Trace event type identifier list exploration (*posix\_trace\_eventtypelist\_getnext\_id()*)

19219

- Trace log open, rewind, and close (*posix\_trace\_open()*)

- 19220 — Trace stream information and status read (*posix\_trace\_get\_attr()*)
- 19221 — Trace event read (*posix\_trace\_getnext\_event()*)

## 19222 2.12 Data Types

### 19223 2.12.1 Defined Types

19224 All of the data types used by various functions are defined by the implementation. The  
 19225 following table describes some of these types. Other types referenced in the description of a  
 19226 function, not mentioned here, can be found in the appropriate header for that function.

19227	Defined Type	Description
19228	<b>cc_t</b>	Type used for terminal special characters.
19229	<b>clock_t</b>	Integer or real-floating type used for processor times, as defined in the ISO C standard.
19230		
19231	<b>clockid_t</b>	Used for clock ID type in some timer functions.
19232	<b>dev_t</b>	Integer type used for device numbers.
19233	<b>DIR</b>	Type representing a directory stream.
19234	<b>div_t</b>	Structure type returned by the <i>div()</i> function.
19235	<b>FILE</b>	Structure containing information about a file.
19236	<b>glob_t</b>	Structure type used in pathname pattern matching.
19237	<b>fpos_t</b>	Type containing all information needed to specify uniquely every position within a file.
19238		
19239	<b>gid_t</b>	Integer type used for group IDs.
19240	<b>iconv_t</b>	Type used for conversion descriptors.
19241	<b>id_t</b>	Integer type used as a general identifier; can be used to contain at least the largest of a <b>pid_t</b> , <b>uid_t</b> , or <b>gid_t</b> .
19242		
19243	<b>ino_t</b>	Unsigned integer type used for file serial numbers.
19244	<b>key_t</b>	Arithmetic type used for XSI interprocess communication.
19245	<b>ldiv_t</b>	Structure type returned by the <i>ldiv()</i> function.
19246	<b>mode_t</b>	Integer type used for file attributes.
19247	<b>mqd_t</b>	Used for message queue descriptors.
19248	<b>nfds_t</b>	Integer type used for the number of file descriptors.
19249	<b>nlink_t</b>	Integer type used for link counts.
19250	<b>off_t</b>	Signed integer type used for file sizes.
19251	<b>pid_t</b>	Signed integer type used for process and process group IDs.
19252	<b>pthread_attr_t</b>	Used to identify a thread attribute object.
19253	<b>pthread_cond_t</b>	Used for condition variables.
19254	<b>pthread_condattr_t</b>	Used to identify a condition attribute object.
19255	<b>pthread_key_t</b>	Used for thread-specific data keys.
19256	<b>pthread_mutex_t</b>	Used for mutexes.
19257	<b>pthread_mutexattr_t</b>	Used to identify a mutex attribute object.
19258	<b>pthread_once_t</b>	Used for dynamic package initialization.
19259	<b>pthread_rwlock_t</b>	Used for read-write locks.
19260	<b>pthread_rwlockattr_t</b>	Used for read-write lock attributes.
19261	<b>pthread_t</b>	Used to identify a thread.



Defined Type	Description
<b>ptrdiff_t</b>	Signed integer type of the result of subtracting two pointers.
<b>regex_t</b>	Structure type used in regular expression matching.
<b>regmatch_t</b>	Structure type used in regular expression matching.
<b>rlim_t</b>	Unsigned integer type used for limit values, to which objects of type <b>int</b> and <b>off_t</b> can be cast without loss of value.
<b>sem_t</b>	Type used in performing semaphore operations.
<b>sig_atomic_t</b>	Possibly volatile-qualified integer type of an object that can be accessed as an atomic entity, even in the presence of asynchronous interrupts.
<b>sigset_t</b>	Integer or structure type of an object used to represent sets of signals.
<b>size_t</b>	Unsigned integer type used for size of objects.
<b>speed_t</b>	Type used for terminal baud rates.
<b>ssize_t</b>	Signed integer type used for a count of bytes or an error indication.
<b>suseconds_t</b>	Signed integer type used for time in microseconds.
<b>tcflag_t</b>	Type used for terminal modes.
<b>time_t</b>	Integer type used for time in seconds, as defined in the ISO C standard.
<b>timer_t</b>	Used for timer ID returned by the <i>timer_create()</i> function.
<b>uid_t</b>	Integer type used for user IDs.
<b>va_list</b>	Type used for traversing variable argument lists.
<b>wchar_t</b>	Integer type whose range of values can represent distinct codes for all members of the largest extended character set specified by the supported locales.
<b>wctype_t</b>	Scalar type which represents a character class descriptor.
<b>wint_t</b>	Integer type capable of storing any valid value of <b>wchar_t</b> or WEOF.
<b>wordexp_t</b>	Structure type used in word expansion.

## 2.12.2 The char Type

The type **char** is defined as a single byte; see XBD [Chapter 3](#) (on page 33) (Byte and Character).

## 2.13 Status Information

Status information is data associated with a process detailing a change in the state of the process. It shall consist of:

- The state the process transitioned into (*stopped*, *continued*, or *terminated*)
  - The information necessary to populate the **siginfo\_t** structure provided by *waitid()*
  - If the new state is *terminated*:
    - The low-order 8 bits of the status argument that the process passed to *\_Exit()*, *\_exit()*, or *exit()*, or the low-order 8 bits of the value the process returned from *main()*
- Note that these 8 bits are part of the complete value that is used to set the *si\_status* member of the **siginfo\_t** structure provided by *waitid()*



19304	— Whether the process terminated due to the receipt of a signal that was not caught	+
19305	and, if so, the number of the signal that caused the termination of the process	+
19306	• If the new state is <i>stopped</i> :	+
19307	— The number of the signal that caused the process to stop	+
19308	A process might not have any status information (such as immediately after a process has	+
19309	started).	+
19310	Status information for a process shall be generated (made available to the parent process) when	+
19311	the process stops, continues, or terminates except in the following case:	+
19312	• If the parent process sets the action for the SIGCHLD signal to SIG_IGN, or if the parent	+
19313	sets the SA_NOCLDWAIT flag for the SIGCHLD signal action, process termination shall	+
19314	not generate new status information but shall cause any existing status information for the	+
19315	process to be discarded.	+
19316	If new status information is generated, and the process already had status information, the	+
19317	existing status information shall be discarded and replaced with the new status information.	+
19318	Only the process' parent process can obtain the process' status information. The parent obtains a	+
19319	child's status information by calling <i>wait()</i> , <i>waitid()</i> , or <i>waitpid()</i> . Except when <i>waitid()</i> is called	+
19320	with the WNOWAIT flag set in the <i>options</i> argument, the status information obtained by a <i>wait()</i>	+
19321	function shall be consumed (discarded) by that <i>wait()</i> function; no two calls to <i>wait()</i> , <i>waitid()</i>	+
19322	(without WNOWAIT), or <i>waitpid()</i> shall obtain the same status information.	+
19323	When status information becomes available to the parent process and more than one thread in	+
19324	the parent process is waiting for the status information (blocked in a call to <i>wait()</i> , <i>waitid()</i> , or	+
19325	<i>waitpid()</i> with arguments that would match the status information):	+
19326	• If none of the matching threads is in a call to <i>waitid()</i> with the WNOWAIT flag set in the	+
19327	<i>options</i> argument, the thread that obtains the status information is unspecified.	+
19328	• Otherwise (at least one of the matching threads is in a call to <i>waitid()</i> with the WNOWAIT	+
19329	flag set), the matching thread or threads that obtain the status information is unspecified	+
19330	except that at least one of the matching threads shall obtain the status information and at	+
19331	most one of the matching threads that are not calling <i>waitid()</i> with the WNOWAIT flag set	+
19332	shall obtain it.	+

## 19333 2.14 File Descriptor Allocation +


19334	All functions that open one or more file descriptors shall, unless specified otherwise, atomically	+
19335	allocate the lowest numbered available (that is, not already open in the calling process) file	+
19336	descriptor at the time of each allocation. Where a single function allocates two file descriptors	+
19337	(for example, <i>pipe()</i> or <i>socketpair()</i> ), the allocations may be independent and therefore	+
19338	applications should not expect them to have adjacent values or depend on which has the higher	+
19339	value.	+



19341

*Chapter 3*

19342



# *System Interfaces*

19343

This chapter describes the functions, macros, and external variables to support applications portability at the C-language source level.

19344

19345 **NAME**

19346 FD\_CLR — macros for synchronous I/O multiplexing

19347 **SYNOPSIS**

```
19348     #include <sys/select.h>
19349     void FD_CLR(int fd, fd_set *fdset);
19350     int FD_ISSET(int fd, fd_set *fdset);
19351     void FD_SET(int fd, fd_set *fdset);
19352     void FD_ZERO(fd_set *fdset);
```

19353 **DESCRIPTION**19354 Refer to *pselect()*.

19355 **NAME**19356        `_Exit`, `_exit` — terminate a process19357 **SYNOPSIS**19358        `#include <stdlib.h>`19359        `void _Exit(int status);`19360        `#include <unistd.h>`19361        `void _exit(int status);`19362 **DESCRIPTION**

19363 CX       For `_Exit()`: The functionality described on this reference page is aligned with the ISO C  
 19364 standard. Any conflict between the requirements described here and the ISO C standard is  
 19365 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

19366 CX       The value of *status* may be 0, `EXIT_SUCCESS`, `EXIT_FAILURE`, or any other value, though only  
 19367 the least significant 8 bits (that is, *status* & 0377) shall be available to a waiting parent process.

19368 CX       The `_Exit()` and `_exit()` functions shall be functionally equivalent.

19369 CX       The `_Exit()` and `_exit()` functions shall not call functions registered with `atexit()` nor any  
 19370 CX registered signal handlers. Open streams shall not be flushed. Whether open streams are  
 19371 closed (without flushing) is implementation-defined. Finally, the calling process shall be  
 19372 terminated with the consequences described below.

19373 **Consequences of Process Termination**

19374 CX       Process termination caused by any reason shall have the following consequences:

19375 **Note:**       These consequences are all extensions to the ISO C standard and are not further CX shaded.  
 19376               However, functionality relating to the XSI option is shaded.

- 19377 • All of the file descriptors, directory streams, conversion descriptors, and message catalog  
 19378 descriptors open in the calling process shall be closed.

- 19379 XSI • If the parent process of the calling process is executing a `wait()`, `waitid()`, or `waitpid()`, and  
 19380 has neither set its `SA_NOCLDWAIT` flag nor set `SIGCHLD` to `SIG_IGN`, it shall be notified  
 19381 of termination of the calling process and the low-order eight bits (that is, bits 0377) of *status*  
 19382 shall be made available to it. If the parent is not waiting, the child's status shall be made  
 19383 available to it when the parent subsequently executes `wait()`, `waitid()`, or `waitpid()`.

19384       The semantics of the `waitid()` function shall be equivalent to `wait()`.

- 19385 XSI • If the parent process of the calling process is not executing a `wait()`, `waitid()`, or `waitpid()`,  
 19386 and has neither set its `SA_NOCLDWAIT` flag nor set `SIGCHLD` to `SIG_IGN`, the calling  
 19387 process shall be transformed into a *zombie process*. A *zombie process* is an inactive process  
 19388 and it shall be deleted at some later time when its parent process executes `wait()`, `waitid()`,  
 19389 or `waitpid()`.

19390 XSI       The semantics of the `waitid()` function shall be equivalent to `wait()`.

- 19391 • Termination of a process does not directly terminate its children. The sending of a `SIGHUP`  
 19392 signal as described below indirectly terminates children in some circumstances.

- 19393 • Either:

19394       If the implementation supports the `SIGCHLD` signal, a `SIGCHLD` shall be sent to the  
 19395 parent process.

19396       Or:

- 19397 XSI If the parent process has set its SA\_NOCLDWAIT flag, or set SIGCHLD to SIG\_IGN, the  
19398 status shall be discarded, and the lifetime of the calling process shall end immediately. If  
19399 SA\_NOCLDWAIT is set, it is implementation-defined whether a SIGCHLD signal is sent to  
19400 the parent process.
- 19401 • The parent process ID of all of the existing child processes and zombie processes of the  
19402 calling process shall be set to the process ID of an implementation-defined system process.  
19403 That is, these processes shall be inherited by a special system process.
- 19404 XSI • Each attached shared-memory segment is detached and the value of *shm\_nattch* (see  
19405 *shmget()*) in the data structure associated with its shared memory ID shall be decremented  
19406 by 1.
- 19407 XSI • For each semaphore for which the calling process has set a *semadj* value (see *semop()*), that  
19408 value shall be added to the *semval* of the specified semaphore.
- 19409 • If the process is a controlling process, the SIGHUP signal shall be sent to each process in  
19410 the foreground process group of the controlling terminal belonging to the calling process.
- 19411 • If the process is a controlling process, the controlling terminal associated with the session  
19412 shall be disassociated from the session, allowing it to be acquired by a new controlling  
19413 process.
- 19414 • If the exit of the process causes a process group to become orphaned, and if any member of  
19415 the newly-orphaned process group is stopped, then a SIGHUP signal followed by a  
19416 SIGCONT signal shall be sent to each process in the newly-orphaned process group.
- 19417 • All open named semaphores in the calling process shall be closed as if by appropriate calls  
19418 to *sem\_close()*.
- 19419 ML • Any memory locks established by the process via calls to *mlockall()* or *mlock()* shall be  
19420 removed. If locked pages in the address space of the calling process are also mapped into  
19421 the address spaces of other processes and are locked by those processes, the locks  
19422 established by the other processes shall be unaffected by the call by this process to *\_Exit()*  
19423 or *\_exit()*.
- 19424 • Memory mappings that were created in the process shall be unmapped before the process  
19425 is destroyed.
- 19426 TYM • Any blocks of typed memory that were mapped in the calling process shall be unmapped,  
19427 as if *munmap()* was implicitly called to unmap them.
- 19428 MSG • All open message queue descriptors in the calling process shall be closed as if by  
19429 appropriate calls to *mq\_close()*.
- 19430 • Any outstanding cancelable asynchronous I/O operations may be canceled. Those  
19431 asynchronous I/O operations that are not canceled shall complete as if the *\_Exit()* or  
19432 *\_exit()* operation had not yet occurred, but any associated signal notifications shall be  
19433 suppressed. The *\_Exit()* or *\_exit()* operation may block awaiting such I/O completion.  
19434 Whether any I/O is canceled, and which I/O may be canceled upon *\_Exit()* or *\_exit()*, is  
19435 implementation-defined.
- 19436 • Threads terminated by a call to *\_Exit()* or *\_exit()* shall not invoke their cancellation  
19437 cleanup handlers or per-thread data destructors.
- 19438 OB TRC • If the calling process is a trace controller process, any trace streams that were created by  
19439 the calling process shall be shut down as described by the *posix\_trace\_shutdown()* function,  
19440 and mapping of trace event names to trace event type identifiers of any process built for  
19441 these trace streams may be deallocated.

**RETURN VALUE**

These functions do not return.

**ERRORS**

No errors are defined.

**EXAMPLES**

None.

**APPLICATION USAGE**

Normally applications should use *exit()* rather than *\_Exit()* or *\_exit()*.

**RATIONALE****Process Termination**

Early proposals drew a distinction between normal and abnormal process termination. Abnormal termination was caused only by certain signals and resulted in implementation-defined “actions”, as discussed below. Subsequent proposals distinguished three types of termination: *normal termination* (as in the current specification), *simple abnormal termination*, and *abnormal termination with actions*. Again the distinction between the two types of abnormal termination was that they were caused by different signals and that implementation-defined actions would result in the latter case. Given that these actions were completely implementation-defined, the early proposals were only saying when the actions could occur and how their occurrence could be detected, but not what they were. This was of little or no use to conforming applications, and thus the distinction is not made in this volume of POSIX.1-2008.

The implementation-defined actions usually include, in most historical implementations, the creation of a file named **core** in the current working directory of the process. This file contains an image of the memory of the process, together with descriptive information about the process, perhaps sufficient to reconstruct the state of the process at the receipt of the signal.

There is a potential security problem in creating a **core** file if the process was set-user-ID and the current user is not the owner of the program, if the process was set-group-ID and none of the user’s groups match the group of the program, or if the user does not have permission to write in the current directory. In this situation, an implementation either should not create a **core** file or should make it unreadable by the user.

Despite the silence of this volume of POSIX.1-2008 on this feature, applications are advised not to create files named **core** because of potential conflicts in many implementations. Some implementations use a name other than **core** for the file; for example, by appending the process ID to the filename.

**Terminating a Process**

It is important that the consequences of process termination as described occur regardless of whether the process called *\_exit()* (perhaps indirectly through *exit()*) or instead was terminated due to a signal or for some other reason. Note that in the specific case of *exit()* this means that the *status* argument to *exit()* is treated in the same way as the *status* argument to *\_exit()*.

A language other than C may have other termination primitives than the C-language *exit()* function, and programs written in such a language should use its native termination primitives, but those should have as part of their function the behavior of *\_exit()* as described. Implementations in languages other than C are outside the scope of this version of this volume of POSIX.1-2008, however.

As required by the ISO C standard, using **return** from *main()* has the same behavior (other than

with respect to language scope issues) as calling *exit()* with the returned value. Reaching the end of the *main()* function has the same behavior as calling *exit(0)*.

A value of zero (or *EXIT\_SUCCESS*, which is required to be zero) for the argument *status* conventionally indicates successful termination. This corresponds to the specification for *exit()* in the ISO C standard. The convention is followed by utilities such as *make* and various shells, which interpret a zero status from a child process as success. For this reason, applications should not call *exit(0)* or *\_exit(0)* when they terminate unsuccessfully; for example, in signal-catching functions.

Historically, the implementation-defined process that inherits children whose parents have terminated without waiting on them is called *init* and has a process ID of 1.

The sending of a *SIGHUP* to the foreground process group when a controlling process terminates corresponds to somewhat different historical implementations. In System V, the kernel sends a *SIGHUP* on termination of (essentially) a controlling process. In 4.2 BSD, the kernel does not send *SIGHUP* in a case like this, but the termination of a controlling process is usually noticed by a system daemon, which arranges to send a *SIGHUP* to the foreground process group with the *vhangup()* function. However, in 4.2 BSD, due to the behavior of the shells that support job control, the controlling process is usually a shell with no other processes in its process group. Thus, a change to make *\_exit()* behave this way in such systems should not cause problems with existing applications.

The termination of a process may cause a process group to become orphaned in either of two ways. The connection of a process group to its parent(s) outside of the group depends on both the parents and their children. Thus, a process group may be orphaned by the termination of the last connecting parent process outside of the group or by the termination of the last direct descendant of the parent process(es). In either case, if the termination of a process causes a process group to become orphaned, processes within the group are disconnected from their job control shell, which no longer has any information on the existence of the process group. Stopped processes within the group would languish forever. In order to avoid this problem, newly orphaned process groups that contain stopped processes are sent a *SIGHUP* signal and a *SIGCONT* signal to indicate that they have been disconnected from their session. The *SIGHUP* signal causes the process group members to terminate unless they are catching or ignoring *SIGHUP*. Under most circumstances, all of the members of the process group are stopped if any of them are stopped.

The action of sending a *SIGHUP* and a *SIGCONT* signal to members of a newly orphaned process group is similar to the action of 4.2 BSD, which sends *SIGHUP* and *SIGCONT* to each stopped child of an exiting process. If such children exit in response to the *SIGHUP*, any additional descendants receive similar treatment at that time. In this volume of POSIX.1-2008, the signals are sent to the entire process group at the same time. Also, in this volume of POSIX.1-2008, but not in 4.2 BSD, stopped processes may be orphaned, but may be members of a process group that is not orphaned; therefore, the action taken at *\_exit()* must consider processes other than child processes.

It is possible for a process group to be orphaned by a call to *setpgid()* or *setsid()*, as well as by process termination. This volume of POSIX.1-2008 does not require sending *SIGHUP* and *SIGCONT* in those cases, because, unlike process termination, those cases are not caused accidentally by applications that are unaware of job control. An implementation can choose to send *SIGHUP* and *SIGCONT* in those cases as an extension; such an extension must be documented as required in **<signal.h>**.

The ISO/IEC 9899:1999 standard adds the *\_Exit()* function that results in immediate program termination without triggering signals or *atexit()*-registered functions. In POSIX.1-2008, this is



19534 equivalent to the `_exit()` function.

#### 19535 FUTURE DIRECTIONS

19536 None.

#### 19537 SEE ALSO

19538 *atexit()*, *exit()*, *mlock()*, *mlockall()*, *mq\_close()*, *munmap()*, *posix\_trace\_create()*, *sem\_close()*,  
19539 *semop()*, *setpgid()*, *setsid()*, *shmget()*, *wait()*, *waitid()*

19540 XBD `<stdlib.h>`, `<unistd.h>`

#### 19541 CHANGE HISTORY

19542 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 19543 Issue 5

19544 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX  
19545 Threads Extension.

19546 Interactions with the SA\_NOCLDWAIT flag and SIGCHLD signal are further clarified.

19547 The values of *status* from *exit()* are better described.

#### 19548 Issue 6

19549 Extensions beyond the ISO C standard are marked.

19550 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding semantics  
19551 for typed memory.

19552 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 19553 • The `_Exit()` function is included.
- 19554 • The DESCRIPTION is updated.

19555 The description of tracing semantics is added for alignment with IEEE Std 1003.1q-2000.

19556 References to the *wait3()* function are removed.

19557 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/16 is applied, correcting grammar in the  
19558 DESCRIPTION.

#### 19559 Issue 7

19560 Austin Group Interpretation 1003.1-2001 #031 is applied, separating these functions from the  
19561 *exit()* function.

19562 Austin Group Interpretation 1003.1-2001 #085 is applied, clarifying the text regarding flushing of  
19563 streams and closing of temporary files.

19564 Functionality relating to the Asynchronous Input and Output, Memory Mapped Files, and  
19565 Semaphores options is moved to the Base.

19566 **NAME**

19567        \_longjmp, \_setjmp — non-local goto

19568 **SYNOPSIS**

```
19569 OB XSI  #include <setjmp.h>
19570          void _longjmp(jmp_buf env, int val);
19571          int _setjmp(jmp_buf env);
```

19572 **DESCRIPTION**

19573        The *\_longjmp()* and *\_setjmp()* functions shall be equivalent to *longjmp()* and *setjmp()*,  
19574        respectively, with the additional restriction that *\_longjmp()* and *\_setjmp()* shall not manipulate  
19575        the signal mask.

19576        If *\_longjmp()* is called even though *env* was never initialized by a call to *\_setjmp()*, or when the  
19577        last such call was in a function that has since returned, the results are undefined.

19578 **RETURN VALUE**19579        Refer to *longjmp()* and *setjmp()*.19580 **ERRORS**

19581        No errors are defined.

19582 **EXAMPLES**

19583        None.

19584 **APPLICATION USAGE**

19585        If *\_longjmp()* is executed and the environment in which *\_setjmp()* was executed no longer exists,  
19586        errors can occur. The conditions under which the environment of the *\_setjmp()* no longer exists  
19587        include exiting the function that contains the *\_setjmp()* call, and exiting an inner block with  
19588        temporary storage. This condition might not be detectable, in which case the *\_longjmp()* occurs  
19589        and, if the environment no longer exists, the contents of the temporary storage of an inner block  
19590        are unpredictable. This condition might also cause unexpected process termination. If the  
19591        function has returned, the results are undefined.

19592        Passing *longjmp()* a pointer to a buffer not created by *setjmp()*, passing *\_longjmp()* a pointer to a  
19593        buffer not created by *\_setjmp()*, passing *siglongjmp()* a pointer to a buffer not created by  
19594        *sigsetjmp()*, or passing any of these three functions a buffer that has been modified by the user  
19595        can cause all the problems listed above, and more.

19596        The *\_longjmp()* and *\_setjmp()* functions are included to support programs written to historical  
19597        system interfaces. New applications should use *siglongjmp()* and *sigsetjmp()* respectively.

19598 **RATIONALE**

19599        None.

19600 **FUTURE DIRECTIONS**19601        The *\_longjmp()* and *\_setjmp()* functions may be removed in a future version.19602 **SEE ALSO**19603        *longjmp()*, *setjmp()*, *siglongjmp()*, *sigsetjmp()*19604        XBD *<setjmp.h>*19605 **CHANGE HISTORY**

19606        First released in Issue 4, Version 2.

19607 **Issue 5**

19608 Moved from X/OPEN UNIX extension to BASE.

19609 **Issue 7**19610 The *\_longjmp()* and *\_setjmp()* functions are marked obsolescent.

**19611 NAME**

19612        \_toupper — transliterate uppercase characters to lowercase

**19613 SYNOPSIS**

```
19614 OB XSI #include <ctype.h>  
19615 int _tolower(int c);
```

**19616 DESCRIPTION**

19617        The *\_tolower()* macro shall be equivalent to *tolower(c)* except that the application shall ensure  
19618        that the argument *c* is an uppercase letter.

**19619 RETURN VALUE**

19620        Upon successful completion, *\_tolower()* shall return the lowercase letter corresponding to the  
19621        argument passed.

**19622 ERRORS**

19623        No errors are defined.

**19624 EXAMPLES**

19625        None.

**19626 APPLICATION USAGE**

19627        Applications should use the *tolower()* function instead of the obsolescent *\_tolower()* function.

**19628 RATIONALE**

19629        None.

**19630 FUTURE DIRECTIONS**

19631        The *\_tolower()* function may be removed in a future version.

**19632 SEE ALSO**

19633        *tolower()*, *isupper()*

19634        XBD Chapter 7 (on page 135), *<ctype.h>*

**19635 CHANGE HISTORY**

19636        First released in Issue 1. Derived from Issue 1 of the SVID.

**19637 Issue 6**

19638        The normative text is updated to avoid use of the term “must” for application requirements.

**19639 Issue 7**

19640        The *\_tolower()* function is marked obsolescent.

**19641 NAME**

19642        \_toupper — transliterate lowercase characters to uppercase

**19643 SYNOPSIS**

```
19644 OB XSI  #include <ctype.h>
19645          int _toupper(int c);
```

**19646 DESCRIPTION**

19647        The *\_toupper()* macro shall be equivalent to *toupper()* except that the application shall ensure  
19648        that the argument *c* is a lowercase letter.

**19649 RETURN VALUE**

19650        Upon successful completion, *\_toupper()* shall return the uppercase letter corresponding to the  
19651        argument passed.

**19652 ERRORS**

19653        No errors are defined.

**19654 EXAMPLES**

19655        None.

**19656 APPLICATION USAGE**

19657        Applications should use the *toupper()* function instead of the obsolescent *\_toupper()* function.

**19658 RATIONALE**

19659        None.

**19660 FUTURE DIRECTIONS**

19661        The *\_toupper()* function may be removed in a future version.

**19662 SEE ALSO**

19663        *islower()*, *toupper()*

19664        XBD Chapter 7 (on page 135), *<ctype.h>*

**19665 CHANGE HISTORY**

19666        First released in Issue 1. Derived from Issue 1 of the SVID.

**19667 Issue 6**

19668        The normative text is updated to avoid use of the term “must” for application requirements.

**19669 Issue 7**

19670        The *\_toupper()* function is marked obsolescent.

19671 **NAME**

19672 a64l, l64a — convert between a 32-bit integer and a radix-64 ASCII string

19673 **SYNOPSIS**

```
19674 XSI      #include <stdlib.h>
19675          long a64l(const char *s);
19676          char *l64a(long value);
```

19677 **DESCRIPTION**

19678 These functions maintain numbers stored in radix-64 ASCII characters. This is a notation by  
 19679 which 32-bit integers can be represented by up to six characters; each character represents a digit  
 19680 in radix-64 notation. If the type **long** contains more than 32 bits, only the low-order 32 bits shall  
 19681 be used for these operations.

19682 The characters used to represent digits are ' .' (dot) for 0, ' / ' for 1, ' 0 ' through ' 9 ' for [2,11],  
 19683 ' A ' through ' Z ' for [12,37], and ' a ' through ' z ' for [38,63].

19684 The *a64l()* function shall take a pointer to a radix-64 representation, in which the first digit is the  
 19685 least significant, and return the corresponding **long** value. If the string pointed to by *s* contains  
 19686 more than six characters, *a64l()* shall use the first six. If the first six characters of the string  
 19687 contain a null terminator, *a64l()* shall use only characters preceding the null terminator. The  
 19688 *a64l()* function shall scan the character string from left to right with the least significant digit on  
 19689 the left, decoding each character as a 6-bit radix-64 number. If the type **long** contains more than  
 19690 32 bits, the resulting value is sign-extended. The behavior of *a64l()* is unspecified if *s* is a null  
 19691 pointer or the string pointed to by *s* was not generated by a previous call to *l64a()*.

19692 The *l64a()* function shall take a **long** argument and return a pointer to the corresponding  
 19693 radix-64 representation. The behavior of *l64a()* is unspecified if *value* is negative.

19694 The value returned by *l64a()* may be a pointer into a static buffer. Subsequent calls to *l64a()* may  
 19695 overwrite the buffer.

19696 The *l64a()* function need not be thread-safe.

19697 **RETURN VALUE**

19698 Upon successful completion, *a64l()* shall return the **long** value resulting from conversion of the  
 19699 input string. If a string pointed to by *s* is an empty string, *a64l()* shall return 0L.

19700 The *l64a()* function shall return a pointer to the radix-64 representation. If *value* is 0L, *l64a()* shall  
 19701 return a pointer to an empty string.

19702 **ERRORS**

19703 No errors are defined.

19704 **EXAMPLES**

19705 None.

19706 **APPLICATION USAGE**

19707 If the type **long** contains more than 32 bits, the result of *a64l(l64a(x))* is *x* in the low-order 32 bits.

19708 **RATIONALE**

19709 This is not the same encoding as used by either encoding variant of the *uuencode* utility.

19710 **FUTURE DIRECTIONS**

19711 None.

19712 **SEE ALSO**19713 *strtoul()*

19714 XBD &lt;stdlib.h&gt;

19715 XCU *uuencode*19716 **CHANGE HISTORY**

19717 First released in Issue 4, Version 2.

19718 **Issue 5**

19719 Moved from X/OPEN UNIX extension to BASE.

19720 Normative text previously in the APPLICATION USAGE section is moved to the  
19721 DESCRIPTION.19722 A note indicating that the *l64a()* function need not be reentrant is added to the DESCRIPTION.19723 **Issue 7**

19724 Austin Group Interpretation 1003.1-2001 #156 is applied.

19725 **NAME**

19726 abort — generate an abnormal process abort

19727 **SYNOPSIS**

19728 #include &lt;stdlib.h&gt;

19729 void abort(void);

19730 **DESCRIPTION**

19731 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
19732 conflict between the requirements described here and the ISO C standard is unintentional. This  
19733 volume of POSIX.1-2008 defers to the ISO C standard.

19734 The *abort()* function shall cause abnormal process termination to occur, unless the signal  
19735 SIGABRT is being caught and the signal handler does not return.

19736 CX The abnormal termination processing shall include the default actions defined for SIGABRT and  
19737 may include an attempt to effect *fclose()* on all open streams.

19738 The SIGABRT signal shall be sent to the calling process as if by means of *raise()* with the  
19739 argument SIGABRT.

19740 CX The status made available to *wait()*, *waitid()*, or *waitpid()* by *abort()* shall be that of a process  
19741 terminated by the SIGABRT signal. The *abort()* function shall override blocking or ignoring the  
19742 SIGABRT signal.

19743 **RETURN VALUE**19744 The *abort()* function shall not return.19745 **ERRORS**

19746 No errors are defined.

19747 **EXAMPLES**

19748 None.

19749 **APPLICATION USAGE**

19750 Catching the signal is intended to provide the application developer with a portable means to  
19751 abort processing, free from possible interference from any implementation-supplied functions.

19752 **RATIONALE**

19753 The ISO/IEC 9899:1999 standard requires the *abort()* function to be async-signal-safe. Since  
19754 POSIX.1-2008 defers to the ISO C standard, this required a change to the DESCRIPTION from  
19755 “shall include the effect of *fclose()*” to “may include an attempt to effect *fclose()*.”

19756 The revised wording permits some backwards-compatibility and avoids a potential deadlock  
19757 situation.

19758 The Open Group Base Resolution bwg2002-003 is applied, removing the following XSI shaded  
19759 paragraph from the DESCRIPTION:

19760 “On XSI-conformant systems, in addition the abnormal termination processing shall include the  
19761 effect of *fclose()* on message catalog descriptors.”

19762 There were several reasons to remove this paragraph:

- 19763 • No special processing of open message catalogs needs to be performed prior to abnormal  
19764 process termination.
- 19765 • The main reason to specifically mention that *abort()* includes the effect of *fclose()* on open  
19766 streams is to flush output queued on the stream. Message catalogs in this context are read-  
19767 only and, therefore, do not need to be flushed.



- The effect of *fclose()* on a message catalog descriptor is unspecified. Message catalog descriptors are allowed, but not required to be implemented using a file descriptor, but there is no mention in POSIX.1-2008 of a message catalog descriptor using a standard I/O stream FILE object as would be expected by *fclose()*.

#### 19772 FUTURE DIRECTIONS

19773 None.

#### 19774 SEE ALSO

19775 *exit()*, *kill()*, *raise()*, *signal()*, *wait()*, *waitid()*

19776 XBD <stdlib.h>

#### 19777 CHANGE HISTORY

19778 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 19779 Issue 6

19780 Extensions beyond the ISO C standard are marked.

19781 Changes are made to the DESCRIPTION for alignment with the ISO/IEC 9899:1999 standard.

19782 The Open Group Base Resolution bwg2002-003 is applied.

19783 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/10 is applied, changing the  
19784 DESCRIPTION of abnormal termination processing and adding to the RATIONALE section.

19785 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/9 is applied, changing “implementation-  
19786 defined functions” to “implementation-supplied functions” in the APPLICATION USAGE  
19787 section.

19788 **NAME**

19789       abs — return an integer absolute value

19790 **SYNOPSIS**

19791       #include &lt;stdlib.h&gt;

19792       int abs(int i);

19793 **DESCRIPTION**

19794 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
19795       conflict between the requirements described here and the ISO C standard is unintentional. This  
19796       volume of POSIX.1-2008 defers to the ISO C standard.

19797       The *abs()* function shall compute the absolute value of its integer operand, *i*. If the result cannot  
19798       be represented, the behavior is undefined.

19799 **RETURN VALUE**19800       The *abs()* function shall return the absolute value of its integer operand.19801 **ERRORS**

19802       No errors are defined.

19803 **EXAMPLES**

19804       None.

19805 **APPLICATION USAGE**

19806       In two's-complement representation, the absolute value of the negative integer with largest  
19807       magnitude {INT\_MIN} might not be representable.

19808 **RATIONALE**

19809       None.

19810 **FUTURE DIRECTIONS**

19811       None.

19812 **SEE ALSO**19813       *fabs()*, *labs()*

19814       XBD &lt;stdlib.h&gt;

19815 **CHANGE HISTORY**

19816       First released in Issue 1. Derived from Issue 1 of the SVID.

19817 **Issue 6**

19818       Extensions beyond the ISO C standard are marked.

**NAME**

accept — accept a new connection on a socket

**SYNOPSIS**

```
#include <sys/socket.h>

int accept(int socket, struct sockaddr *restrict address,
           socklen_t *restrict address_len);
```

**DESCRIPTION**

The *accept()* function shall extract the first connection on the queue of pending connections, create a new socket with the same socket type protocol and address family as the specified socket, and allocate a new file descriptor for that socket. The file descriptor shall be allocated as described in [Section 2.14](#) (on page 549).

The *accept()* function takes the following arguments:

<i>socket</i>	Specifies a socket that was created with <i>socket()</i> , has been bound to an address with <i>bind()</i> , and has issued a successful call to <i>listen()</i> .
<i>address</i>	Either a null pointer, or a pointer to a <b>sockaddr</b> structure where the address of the connecting socket shall be returned.
<i>address_len</i>	Either a null pointer, if <i>address</i> is a null pointer, or a pointer to a <b>socklen_t</b> object which on input specifies the length of the supplied <b>sockaddr</b> structure, and on output specifies the length of the stored address.

If *address* is not a null pointer, the address of the peer for the accepted connection shall be stored in the **sockaddr** structure pointed to by *address*, and the length of this address shall be stored in the object pointed to by *address\_len*.

If the actual length of the address is greater than the length of the supplied **sockaddr** structure, the stored address shall be truncated.

If the protocol permits connections by unbound clients, and the peer is not bound, then the value stored in the object pointed to by *address* is unspecified.

If the listen queue is empty of connection requests and **O\_NONBLOCK** is not set on the file descriptor for the socket, *accept()* shall block until a connection is present. If the *listen()* queue is empty of connection requests and **O\_NONBLOCK** is set on the file descriptor for the socket, *accept()* shall fail and set *errno* to **[EAGAIN]** or **[EWOULDBLOCK]**.

The accepted socket cannot itself accept more connections. The original socket remains open and can accept more connections.

**RETURN VALUE**

Upon successful completion, *accept()* shall return the non-negative file descriptor of the accepted socket. Otherwise, **-1** shall be returned, *errno* shall be set to indicate the error, and any object pointed to by *address\_len* shall remain unchanged.

**ERRORS**

The *accept()* function shall fail if:

**[EAGAIN]** or **[EWOULDBLOCK]**

**O\_NONBLOCK** is set for the socket file descriptor and no connections are present to be accepted.

**[EBADF]** The *socket* argument is not a valid file descriptor.

19861		[ECONNABORTED]	
19862			A connection has been aborted.
19863		[EINTR]	The <i>accept()</i> function was interrupted by a signal that was caught before a
19864			valid connection arrived.
19865		[EINVAL]	The <i>socket</i> is not accepting connections.
19866		[EMFILE]	All file descriptors available to the process are currently open.
19867		[ENFILE]	The maximum number of file descriptors in the system are already open.
19868		[ENOBUFS]	No buffer space is available.
19869		[ENOMEM]	There was insufficient memory available to complete the operation.
19870		[ENOTSOCK]	The <i>socket</i> argument does not refer to a socket.
19871		[EOPNOTSUPP]	The socket type of the specified socket does not support accepting
19872			connections.
19873		The <i>accept()</i> function may fail if:	
19874	OB XSR	[EPROTO]	A protocol error has occurred; for example, the STREAMS protocol stack has
19875			not been initialized.

## 19876 EXAMPLES

19877 None.

## 19878 APPLICATION USAGE

19879 When a connection is available, *select()* indicates that the file descriptor for the socket is ready  
19880 for reading.

## 19881 RATIONALE

19882 None.

## 19883 FUTURE DIRECTIONS

19884 None.

## 19885 SEE ALSO

19886 [Section 2.14](#) (on page 549), *bind()*, *connect()*, *listen()*, *socket()*

19887 XBD [<sys/socket.h>](#)

## 19888 CHANGE HISTORY

19889 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

19890 The **restrict** keyword is added to the *accept()* prototype for alignment with the  
19891 ISO/IEC 9899:1999 standard.

## 19892 Issue 7

19893 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

19894 Austin Group Interpretation 1003.1-2001 #044 is applied, changing the “may fail” [ENOBUFS]  
19895 and [ENOMEM] errors to become “shall fail” errors.

19896 Functionality relating to XSI STREAMS is marked obsolescent.

19897 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0018 [464] is applied.

19898 **NAME**

19899       access, faccessat — determine accessibility of a file descriptor |

19900 **SYNOPSIS**

19901       #include &lt;unistd.h&gt;

19902       int access(const char \*path, int amode);

19903 OH     #include &lt;fcntl.h&gt; +

19904 +

19905       int faccessat(int fd, const char \*path, int amode, int flag);

19906 **DESCRIPTION**

19907       The *access()* function shall check the file named by the pathname pointed to by the *path*  
 19908       argument for accessibility according to the bit pattern contained in *amode*. The checks for  
 19909       accessibility (including directory permissions checked during pathname resolution) shall be  
 19910       performed using the real user ID in place of the effective user ID and the real group ID in place  
 19911       of the effective group ID.

19912       The value of *amode* is either the bitwise-inclusive OR of the access permissions to be checked  
 19913       (R\_OK, W\_OK, X\_OK) or the existence test (F\_OK).

19914       If any access permissions are checked, each shall be checked individually, as described in XBD  
 19915       Section 4.5 (on page 108), except that where that description refers to execute permission for a  
 19916       process with appropriate privileges, an implementation may indicate success for X\_OK even if  
 19917       execute permission is not granted to any user.

19918       The *faccessat()* function, when called with a *flag* value of zero, shall be equivalent to the *access()*  
 19919       function, except in the case where *path* specifies a relative path. In this case the file whose  
 19920       accessibility is to be determined shall be located relative to the directory associated with the file  
 19921       descriptor *fd* instead of the current working directory. If the access mode of the open file  
 19922       description associated with the file descriptor is not O\_SEARCH, the function shall check  
 19923       whether directory searches are permitted using the current permissions of the directory  
 19924       underlying the file descriptor. If the access mode is O\_SEARCH, the function shall not perform  
 19925       the check.

19926       If *faccessat()* is passed the special value AT\_FDCWD in the *fd* parameter, the current working  
 19927       directory shall be used and, if *flag* is zero, the behavior shall be identical to a call to *access()*.

19928       Values for *flag* are constructed by a bitwise-inclusive OR of flags from the following list, defined  
 19929       in <fcntl.h>:

19930	AT_EACCESS	The checks for accessibility (including directory permissions checked during	
19931		pathname resolution) shall be performed using the effective user ID and	
19932		group ID instead of the real user ID and group ID as required in a call to	
19933		<i>access()</i> .	

19934 **RETURN VALUE**

19935       Upon successful completion, these functions shall return 0. Otherwise, these functions shall  
 19936       return -1 and set *errno* to indicate the error.

19937 **ERRORS**

19938       These functions shall fail if:

19939	[EACCES]	Permission bits of the file mode do not permit the requested access, or search	
19940		permission is denied on a component of the path prefix.	

19941	[ELOOP]	A loop exists in symbolic links encountered during resolution of the <i>path</i>
19942		argument.
19943	[ENAMETOOLONG]	
19944		The length of a component of a pathname is longer than {NAME_MAX}.
19945	[ENOENT]	A component of <i>path</i> does not name an existing file or <i>path</i> is an empty string.
19946	[ENOTDIR]	A component of the path prefix names an existing file that is neither a
19947		directory nor a symbolic link to a directory, or the <i>path</i> argument contains at
19948		least one non- <code>&lt;slash&gt;</code> character and ends with one or more trailing <code>&lt;slash&gt;</code>
19949		characters and the last pathname component names an existing file that is
19950		neither a directory nor a symbolic link to a directory.
19951	[EROFS]	Write access is requested for a file on a read-only file system.
19952	The <i>faccessat</i> ( ) function shall fail if:	
19953	[EACCES]	The access mode of the open file description associated with <i>fd</i> is not
19954		O_SEARCH and the permissions of the directory underlying <i>fd</i> do not permit
19955		directory searches.
19956	[EBADF]	The <i>path</i> argument does not specify an absolute path and the <i>fd</i> argument is
19957		neither AT_FDCWD nor a valid file descriptor open for reading or searching.
19958	[ENOTDIR]	The <i>path</i> argument is not an absolute path and <i>fd</i> is a file descriptor associated
19959		with a non-directory file.
19960	These functions may fail if:	
19961	[EINVAL]	The value of the <i>amode</i> argument is invalid.
19962	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during
19963		resolution of the <i>path</i> argument.
19964	[ENAMETOOLONG]	
19965		The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
19966		symbolic link produced an intermediate result with a length that exceeds
19967		{PATH_MAX}.
19968	[ETXTBSY]	Write access is requested for a pure procedure (shared text) file that is being
19969		executed.
19970	The <i>faccessat</i> ( ) function may fail if:	
19971	[EINVAL]	The value of the <i>flag</i> argument is not valid.

## EXAMPLES

### Testing for the Existence of a File

The following example tests whether a file named **myfile** exists in the **/tmp** directory.

```
#include <unistd.h>
...
int result;
const char *pathname = "/tmp/myfile";
result = access (pathname, F_OK);
```

## APPLICATION USAGE

Use of these functions is discouraged since by the time the returned information is acted upon, it is out-of-date. (That is, acting upon the information always leads to a time-of-check-to-time-of-use race condition.) An application should instead attempt the action itself and handle the [EACCES] error that occurs if the file is not accessible (with a change of effective user and group IDs beforehand, and perhaps a change back afterwards, in the case where *access()* or *faccessat()* without AT\_EACCESS would have been used.)

Historically, one of the uses of *access()* was in set-user-ID root programs to check whether the user running the program had access to a file. This relied on “super-user” privileges which were granted based on the effective user ID being zero, so that when *access()* used the real user ID to check accessibility those privileges were not taken into account. On newer systems where privileges can be assigned which have no association with user or group IDs, if a program with such privileges calls *access()*, the change of IDs has no effect on the privileges and therefore they are taken into account in the accessibility checks. Thus, *access()* (and *faccessat()* with flag zero) cannot be used for this historical purpose in such programs. Likewise, if a system provides any additional or alternate file access control mechanisms that are not user ID-based, they will still be taken into account.

If a relative pathname is used, no account is taken of whether the current directory (or the directory associated with the file descriptor *fd*) is accessible via any absolute pathname. Applications using *access()*, or *faccessat()* without AT\_EACCESS, may consequently act as if the file would be accessible to a user with the real user ID and group ID of the process when such a user would not in practice be able to access the file because access would be denied at some point above the current directory (or the directory associated with the file descriptor *fd*) in the file hierarchy.

If *access()* or *faccessat()* is used with W\_OK to check for write access to a directory which has the S\_ISVTX bit set, a return value indicating the directory is writable can be misleading since some operations on files in the directory would not be permitted based on the ownership of those files (see XBD Section 4.3, on page 108).

Additional values of *amode* other than the set defined in the description may be valid; for example, if a system has extended access controls.

The use of the AT\_EACCESS value for *flag* enables functionality not available in *access()*.

## RATIONALE

In early proposals, some inadequacies in the *access()* function led to the creation of an *eaccess()* function because:

1. Historical implementations of *access()* do not test file access correctly when the process' real user ID is superuser. In particular, they always return zero when testing execute permissions without regard to whether the file is executable.
2. The superuser has complete access to all files on a system. As a consequence, programs started by the superuser and switched to the effective user ID with lesser privileges cannot use *access()* to test their file access permissions.

However, the historical model of *eaccess()* does not resolve problem (1), so this volume of POSIX.1-2008 now allows *access()* to behave in the desired way because several implementations have corrected the problem. It was also argued that problem (2) is more easily solved by using *open()*, *chdir()*, or one of the *exec* functions as appropriate and responding to the error, rather than creating a new function that would not be as reliable. Therefore, *eaccess()* is not included in this volume of POSIX.1-2008.

The sentence concerning appropriate privileges and execute permission bits reflects the two



possibilities implemented by historical implementations when checking superuser access for X\_OK.

New implementations are discouraged from returning X\_OK unless at least one execution permission bit is set.

The purpose of the *faccessat()* function is to enable the checking of the accessibility of files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *access()*, resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *faccessat()* function it can be guaranteed that the file tested for accessibility is located relative to the desired directory.

#### FUTURE DIRECTIONS

These functions may be formally deprecated (for example, by shading them OB) in a future version of this standard.

#### SEE ALSO

*chmod()*, *fstatat()*

XBD Section 4.5 (on page 108), *<fcntl.h>*, *<unistd.h>*

#### CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

#### Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.
- The [ETXTBSY] optional error condition is added.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

#### Issue 7

Austin Group Interpretations 1003.1-2001 #046 and #143 are applied.

The *faccessat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Changes are made to allow a directory to be opened for searching.

The [ENOTDIR] error condition is clarified to cover the condition where the last component of a pathname exists but is not a directory or a symbolic link to a directory.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0019 [461], XSH/TC1-2008/0020 [324], XSH/TC1-2008/0021 [278], XSH/TC1-2008/0022 [278], and XSH/TC1-2008/0023 [291] are applied.



20062 **NAME**

20063       acos, acosf, acosl — arc cosine functions

20064 **SYNOPSIS**

```
20065       #include <math.h>
20066       double acos(double x);
20067       float acosf(float x);
20068       long double acosl(long double x);
```

20069 **DESCRIPTION**

20070 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 20071       conflict between the requirements described here and the ISO C standard is unintentional. This  
 20072       volume of POSIX.1-2008 defers to the ISO C standard.

20073       These functions shall compute the principal value of the arc cosine of their argument  $x$ . The  
 20074       value of  $x$  should be in the range  $[-1,1]$ .

20075       An application wishing to check for error situations should set *errno* to zero and call  
 20076       *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 20077       *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 20078       zero, an error has occurred.

20079 **RETURN VALUE**

20080       Upon successful completion, these functions shall return the arc cosine of  $x$ , in the range  $[0,\pi]$   
 20081       radians.

20082 MX       For finite values of  $x$  not in the range  $[-1,1]$ , a domain error shall occur, and either a NaN (if  
 20083       supported), or an implementation-defined value shall be returned.

20084 MX       If  $x$  is NaN, a NaN shall be returned.

20085       If  $x$  is  $+1$ ,  $+0$  shall be returned.

20086       If  $x$  is  $\pm\text{Inf}$ , a domain error shall occur, and a NaN shall be returned.

20087 **ERRORS**

20088       These functions shall fail if:

20089 MX       Domain Error     The  $x$  argument is finite and is not in the range  $[-1,1]$ , or is  $\pm\text{Inf}$ .

20090       If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 20091       then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 20092       & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 20093       shall be raised.

20094 **EXAMPLES**

20095       None.

20096 **APPLICATION USAGE**

20097       On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 20098       MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

20099 **RATIONALE**

20100       None.

20101 **FUTURE DIRECTIONS**

20102       None.

20103 **SEE ALSO**20104 *cos()*, *feclearexcept()*, *fetestexcept()*, *isnan()*20105 XBD Section 4.20 (on page 117), **<math.h>**20106 **CHANGE HISTORY**

20107 First released in Issue 1. Derived from Issue 1 of the SVID.

20108 **Issue 5**20109 The DESCRIPTION is updated to indicate how an application should check for an error. This  
20110 text was previously published in the APPLICATION USAGE section.20111 **Issue 6**20112 The *acosf()* and *acosl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.20113 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
20114 revised to align with the ISO/IEC 9899:1999 standard.20115 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
20116 marked.20117 **Issue 7**

20118 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0024 [320] is applied.

20119 **NAME**

20120 acosh, acoshf, acoshl — inverse hyperbolic cosine functions

20121 **SYNOPSIS**

```
20122 #include <math.h>
20123 double acosh(double x);
20124 float acoshf(float x);
20125 long double acoshl(long double x);
```

20126 **DESCRIPTION**

20127 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 20128 conflict between the requirements described here and the ISO C standard is unintentional. This  
 20129 volume of POSIX.1-2008 defers to the ISO C standard.

20130 These functions shall compute the inverse hyperbolic cosine of their argument  $x$ .

20131 An application wishing to check for error situations should set *errno* to zero and call  
 20132 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 20133 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 20134 zero, an error has occurred.

20135 **RETURN VALUE**

20136 Upon successful completion, these functions shall return the inverse hyperbolic cosine of their  
 20137 argument.

20138 MX For finite values of  $x < 1$ , a domain error shall occur, and either a NaN (if supported), or an  
 20139 implementation-defined value shall be returned.

20140 MX If  $x$  is NaN, a NaN shall be returned.

20141 If  $x$  is +1, +0 shall be returned.

20142 If  $x$  is +Inf, +Inf shall be returned.

20143 If  $x$  is -Inf, a domain error shall occur, and a NaN shall be returned.

20144 **ERRORS**

20145 These functions shall fail if:

20146 MX Domain Error The  $x$  argument is finite and less than +1.0, or is -Inf.

20147 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 20148 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 20149 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 20150 shall be raised.

20151 **EXAMPLES**

20152 None.

20153 **APPLICATION USAGE**

20154 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 20155 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

20156 **RATIONALE**

20157 None.

20158 **FUTURE DIRECTIONS**

20159 None.

20160 **SEE ALSO**20161 *cosh()*, *feclearexcept()*, *fetestexcept()*20162 XBD Section 4.20 (on page 117), **<math.h>**20163 **CHANGE HISTORY**

20164 First released in Issue 4, Version 2.

20165 **Issue 5**

20166 Moved from X/OPEN UNIX extension to BASE.

20167 **Issue 6**20168 The *acosh()* function is no longer marked as an extension.20169 The *acoshf()* and *acoshl()* functions are added for alignment with the ISO/IEC 9899:1999  
20170 standard.20171 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
20172 revised to align with the ISO/IEC 9899:1999 standard.20173 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
20174 marked.20175 **Issue 7**

20176 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0025 [320] is applied.

20177 **NAME**

20178       acosl — arc cosine functions

20179 **SYNOPSIS**

20180       #include &lt;math.h&gt;

20181       long double acosl(long double x);

20182 **DESCRIPTION**20183       Refer to *acos()*.

**NAME**

aio\_cancel — cancel an asynchronous I/O request

**SYNOPSIS**

```
#include <aio.h>
```

```
int aio_cancel(int fildes, struct aiocb *aiocbp);
```

**DESCRIPTION**

The *aio\_cancel()* function shall attempt to cancel one or more asynchronous I/O requests currently outstanding against file descriptor *fildes*. The *aiocbp* argument points to the asynchronous I/O control block for a particular request to be canceled. If *aiocbp* is NULL, then all outstanding cancelable asynchronous I/O requests against *fildes* shall be canceled.

Normal asynchronous notification shall occur for asynchronous I/O operations that are successfully canceled. If there are requests that cannot be canceled, then the normal asynchronous completion process shall take place for those requests when they are completed.

For requested operations that are successfully canceled, the associated error status shall be set to [ECANCELED] and the return status shall be -1. For requested operations that are not successfully canceled, the *aiocbp* shall not be modified by *aio\_cancel()*.

If *aiocbp* is not NULL, then if *fildes* does not have the same value as the file descriptor with which the asynchronous operation was initiated, unspecified results occur.

Which operations are cancelable is implementation-defined.

**RETURN VALUE**

The *aio\_cancel()* function shall return the value AIO\_CANCELED if the requested operation(s) were canceled. The value AIO\_NOTCANCELED shall be returned if at least one of the requested operation(s) cannot be canceled because it is in progress. In this case, the state of the other operations, if any, referenced in the call to *aio\_cancel()* is not indicated by the return value of *aio\_cancel()*. The application may determine the state of affairs for these operations by using *aio\_error()*. The value AIO\_ALLDONE is returned if all of the operations have already completed. Otherwise, the function shall return -1 and set *errno* to indicate the error.

**ERRORS**

The *aio\_cancel()* function shall fail if:

[EBADF] The *fildes* argument is not a valid file descriptor.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*aio\_read()*, *aio\_write()*

XBD *<aio.h>*

20225 **CHANGE HISTORY**

20226 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

20227 **Issue 6**

20228 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
20229 implementation does not support the Asynchronous Input and Output option.

20230 The APPLICATION USAGE section is added.

20231 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/10 is applied, removing the words “to the  
20232 calling process” in the RETURN VALUE section. The term was unnecessary and precluded  
20233 threads.

20234 **Issue 7**

20235 The *aio\_cancel()* function is moved from the Asynchronous Input and Output option to the Base.

**NAME**

aio\_error — retrieve errors status for an asynchronous I/O operation

**SYNOPSIS**

```
#include <aio.h>
```

```
int aio_error(const struct aiocb *aiocbp);
```

**DESCRIPTION**

The *aio\_error()* function shall return the error status associated with the **aiocb** structure referenced by the *aiocbp* argument. The error status for an asynchronous I/O operation is the *errno* value that would be set by the corresponding *read()*, *write()*, *fdatasync()*, or *fsync()* operation. If the operation has not yet completed, then the error status shall be equal to [EINPROGRESS].

If the **aiocb** structure pointed to by *aiocbp* is not associated with an operation that has been scheduled, the results are undefined.

**RETURN VALUE**

If the asynchronous I/O operation has completed successfully, then 0 shall be returned. If the asynchronous operation has completed unsuccessfully, then the error status, as described for *read()*, *write()*, *fdatasync()*, and *fsync()*, shall be returned. If the asynchronous I/O operation has not yet completed, then [EINPROGRESS] shall be returned.

If the *aio\_error()* function fails, it shall return -1 and set *errno* to indicate the error.

**ERRORS**

The *aio\_error()* function may fail if:

[EINVAL] The *aiocbp* argument does not refer to an asynchronous operation whose return status has not yet been retrieved.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*aio\_cancel()*, *aio\_fsync()*, *aio\_read()*, *aio\_return()*, *aio\_write()*, *close()*, *exec*, *exit()*, *fork()*, *lio\_listio()*, *lseek()*, *read()*

XBD **<aio.h>**

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**Issue 6**

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Asynchronous Input and Output option.

The APPLICATION USAGE section is added.



20277 **Issue 7**

20278 Austin Group Interpretation 1003.1-2001 #045 is applied.

20279 SD5-XSH-ERN-148 is applied.

20280 The *aio\_error()* function is moved from the Asynchronous Input and Output option to the Base.

## NAME

**aiio\_fsync** — asynchronous file synchronization

## SYNOPSIS

```
#include <aiio.h>

int aiio_fsync(int op, struct aiocb *aiocbp);
```

## DESCRIPTION

The *aiio\_fsync()* function shall asynchronously perform a file synchronization operation, as specified by the *op* argument, for I/O operations associated with the file indicated by the file descriptor *aiio\_fildes* member of the **aiocb** structure referenced by the *aiocbp* argument and queued at the time of the call to *aiio\_fsync()*. The function call shall return when the synchronization request has been initiated or queued to the file or device (even when the data cannot be synchronized immediately).

If *op* is *O\_DSYNC*, all currently queued I/O operations shall be completed as if by a call to *fdatsync()*; that is, as defined for synchronized I/O data integrity completion.

If *op* is *O\_SYNC*, all currently queued I/O operations shall be completed as if by a call to *fsync()*; that is, as defined for synchronized I/O file integrity completion. If the *aiio\_fsync()* function fails, or if the operation queued by *aiio\_fsync()* fails, then outstanding I/O operations are not guaranteed to have been completed.

If *aiio\_fsync()* succeeds, then it is only the I/O that was queued at the time of the call to *aiio\_fsync()* that is guaranteed to be forced to the relevant completion state. The completion of subsequent I/O on the file descriptor is not guaranteed to be completed in a synchronized fashion.

The *aiocbp* argument refers to an asynchronous I/O control block. The *aiocbp* value may be used as an argument to *aiio\_error()* and *aiio\_return()* in order to determine the error status and return status, respectively, of the asynchronous operation while it is proceeding. When the request is queued, the error status for the operation is *[EINPROGRESS]*. When all data has been successfully transferred, the error status shall be reset to reflect the success or failure of the operation. If the operation does not complete successfully, the error status for the operation shall be set to indicate the error. The *aiio\_sigevent* member determines the asynchronous notification to occur as specified in [Section 2.4.1](#) (on page 488) when all operations have achieved synchronized I/O completion. All other members of the structure referenced by *aiocbp* are ignored. If the control block referenced by *aiocbp* becomes an illegal address prior to asynchronous I/O completion, then the behavior is undefined.

If the *aiio\_fsync()* function fails or *aiocbp* indicates an error condition, data is not guaranteed to have been successfully transferred.

## RETURN VALUE

The *aiio\_fsync()* function shall return the value 0 if the I/O operation is successfully queued; otherwise, the function shall return the value -1 and set *errno* to indicate the error.

## ERRORS

The *aiio\_fsync()* function shall fail if:

**[EAGAIN]** The requested asynchronous operation was not queued due to temporary resource limitations.

**[EBADF]** The *aiio\_fildes* member of the **aiocb** structure referenced by the *aiocbp* argument is not a valid file descriptor.

20325 SIO [EINVAL] This implementation does not support synchronized I/O for this file.

20326 FSC [EINVAL] The *aio\_fildes* member of the **aio\_cb** structure refers to a file on which an *fsync()*  
20327 operation is not possible.

20328 [EINVAL] A value of *op* other than O\_DSYNC or O\_SYNC was specified, or O\_DSYNC  
20329 was specified and the implementation does not provide runtime support for  
20330 the Synchronized Input and Output option, or O\_SYNC was specified and the  
20331 implementation does not provide runtime support for the File  
20332 Synchronization option.

20333 In the event that any of the queued I/O operations fail, *aio\_fsync()* shall return the error  
20334 condition defined for *read()* and *write()*. The error is returned in the error status for the  
20335 asynchronous operation, which can be retrieved using *aio\_error()*.

**EXAMPLES**

None.

**APPLICATION USAGE**

Note that even if the file descriptor is not open for writing, if there are any pending write  
requests on the underlying file, then that I/O will be completed prior to the return of a call to  
*aio\_error()* or *aio\_return()* indicating that the operation has completed.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO***aio\_error()*, *aio\_return()*, *fcntl()*, *fdatasync()*, *fsync()*, *open()*, *read()*, *write()*

XBD &lt;aio.h&gt;

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**Issue 6**

The [ENOSYS] error condition has been removed as stubs need not be provided if an  
implementation does not support the Asynchronous Input and Output option.

The APPLICATION USAGE section is added.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/11 is applied, removing the words “to the  
calling process” in the RETURN VALUE section. The term was unnecessary and precluded  
threads.

**Issue 7**

The *aio\_fsync()* function is moved from the Asynchronous Input and Output option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0026 [98] and XSH/TC1-2008/0027  
[98] are applied.

20362 **NAME**

20363 aio\_read — asynchronous read from a file

20364 **SYNOPSIS**

20365 #include <aio.h>

20366 int aio\_read(struct aiocb \*aiocbp);

20367 **DESCRIPTION**

20368 The *aio\_read()* function shall read *aiocbp->aio\_nbytes* from the file associated with  
20369 *aiocbp->aio\_fildes* into the buffer pointed to by *aiocbp->aio\_buf*. The function call shall return  
20370 when the read request has been initiated or queued to the file or device (even when the data  
20371 cannot be delivered immediately).

20372 PIO

If prioritized I/O is supported for this file, then the asynchronous operation shall be submitted  
at a priority equal to a base scheduling priority minus *aiocbp->aio\_reqprio*. If Thread Execution  
Scheduling is not supported, then the base scheduling priority is that of the calling process;

20375 PIO TPS

otherwise, the base scheduling priority is that of the calling thread.

20376 The *aiocbp* value may be used as an argument to *aio\_error()* and *aio\_return()* in order to  
20377 determine the error status and return status, respectively, of the asynchronous operation while it  
20378 is proceeding. If an error condition is encountered during queuing, the function call shall return  
20379 without having initiated or queued the request. The requested operation takes place at the  
20380 absolute position in the file as given by *aio\_offset*, as if *lseek()* were called immediately prior to  
20381 the operation with an *offset* equal to *aio\_offset* and a *whence* equal to *SEEK\_SET*. After a  
20382 successful call to enqueue an asynchronous I/O operation, the value of the file offset for the file  
20383 is unspecified.

20384 The *aio\_sigevent* member specifies the notification which occurs when the request is completed.

20385 The *aiocbp->aio\_lio\_opcode* field shall be ignored by *aio\_read()*.

20386 The *aiocbp* argument points to an **aiocb** structure. If the buffer pointed to by *aiocbp->aio\_buf* or  
20387 the control block pointed to by *aiocbp* becomes an illegal address prior to asynchronous I/O  
20388 completion, then the behavior is undefined.

20389 Simultaneous asynchronous operations using the same *aiocbp* produce undefined results.

20390 SIO

If synchronized I/O is enabled on the file associated with *aiocbp->aio\_fildes*, the behavior of this  
function shall be according to the definitions of synchronized I/O data integrity completion and  
synchronized I/O file integrity completion.

20393 For any system action that changes the process memory space while an asynchronous I/O is  
20394 outstanding to the address range being changed, the result of that action is undefined.

20395 For regular files, no data transfer shall occur past the offset maximum established in the open  
20396 file description associated with *aiocbp->aio\_fildes*.

20397 **RETURN VALUE**

20398 The *aio\_read()* function shall return the value zero if the I/O operation is successfully queued;  
20399 otherwise, the function shall return the value -1 and set *errno* to indicate the error.

20400 **ERRORS**

20401 The *aio\_read()* function shall fail if:

20402 [EAGAIN] The requested asynchronous I/O operation was not queued due to system  
20403 resource limitations.

20404 Each of the following conditions may be detected synchronously at the time of the call to  
20405 *aio\_read()*, or asynchronously. If any of the conditions below are detected synchronously, the

20406 *aio\_read()* function shall return `-1` and set *errno* to the corresponding value. If any of the  
 20407 conditions below are detected asynchronously, the return status of the asynchronous operation  
 20408 is set to `-1`, and the error status of the asynchronous operation is set to the corresponding value.

20409 [EBADF] The *aiocbp*→*aio\_fildes* argument is not a valid file descriptor open for reading.

20410 [EINVAL] The file offset value implied by *aiocbp*→*aio\_offset* would be invalid,  
 20411 PIO *aiocbp*→*aio\_reqprio* is not a valid value, or *aiocbp*→*aio\_nbytes* is an invalid  
 20412 value.

20413 In the case that the *aio\_read()* successfully queues the I/O operation but the operation is  
 20414 subsequently canceled or encounters an error, the return status of the asynchronous operation is  
 20415 one of the values normally returned by the *read()* function call. In addition, the error status of  
 20416 the asynchronous operation is set to one of the error statuses normally set by the *read()* function  
 20417 call, or one of the following values:

20418 [EBADF] The *aiocbp*→*aio\_fildes* argument is not a valid file descriptor open for reading.

20419 [ECANCELED] The requested I/O was canceled before the I/O completed due to an explicit  
 20420 *aio\_cancel()* request.

20421 [EINVAL] The file offset value implied by *aiocbp*→*aio\_offset* would be invalid.

20422 The following condition may be detected synchronously or asynchronously:

20423 [EOVERFLOW] The file is a regular file, *aiocbp*→*aio\_nbytes* is greater than 0, and the starting  
 20424 offset in *aiocbp*→*aio\_offset* is before the end-of-file and is at or beyond the  
 20425 offset maximum in the open file description associated with *aiocbp*→*aio\_fildes*.

## 20426 EXAMPLES

20427 None.

## 20428 APPLICATION USAGE

20429 None.

## 20430 RATIONALE

20431 None.

## 20432 FUTURE DIRECTIONS

20433 None.

## 20434 SEE ALSO

20435 *aio\_cancel()*, *aio\_error()*, *lio\_listio()*, *aio\_return()*, *aio\_write()*, *close()*, *exec*, *exit()*, *fork()*, *lseek()*,  
 20436 *read()*

20437 XBD <[aio.h](#)>

## 20438 CHANGE HISTORY

20439 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

20440 Large File Summit extensions are added.

## 20441 Issue 6

20442 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
 20443 implementation does not support the Asynchronous Input and Output option.

20444 The APPLICATION USAGE section is added.

20445 The following new requirements on POSIX implementations derive from alignment with the  
20446 Single UNIX Specification:

- 20447 • In the DESCRIPTION, text is added to indicate setting of the offset maximum in the open  
20448 file description. This change is to support large files.
- 20449 • In the ERRORS section, the [EOVERFLOW] condition is added. This change is to support  
20450 large files.

20451 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/12 is applied, rewording the  
20452 DESCRIPTION when prioritized I/O is supported to account for threads, and removing the  
20453 words “to the calling process” in the RETURN VALUE section.

20454 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/13 is applied, updating the [EINVAL]  
20455 error, so that detection of an [EINVAL] error for an invalid value of *aiocbp*→*aio\_reqprio* is only  
20456 required if the Prioritized Input and Output option is supported.

20457 **Issue 7**

20458 Austin Group Interpretation 1003.1-2001 #082 is applied.

20459 The *aio\_read()* function is moved from the Asynchronous Input and Output option to the Base.

20460 **NAME**

20461 aio\_return — retrieve return status of an asynchronous I/O operation

20462 **SYNOPSIS**

```
20463 #include <aio.h>
20464 ssize_t aio_return(struct aiocb *aiocbp);
```

20465 **DESCRIPTION**

20466 The *aio\_return()* function shall return the return status associated with the **aiocb** structure  
 20467 referenced by the *aiocbp* argument. The return status for an asynchronous I/O operation is the  
 20468 value that would be returned by the corresponding *read()*, *write()*, or *fsync()* function call. If the  
 20469 error status for the operation is equal to [EINPROGRESS], then the return status for the  
 20470 operation is undefined. The *aio\_return()* function may be called exactly once to retrieve the  
 20471 return status of a given asynchronous operation; thereafter, if the same **aiocb** structure is used in  
 20472 a call to *aio\_return()* or *aio\_error()*, an error may be returned. When the **aiocb** structure referred  
 20473 to by *aiocbp* is used to submit another asynchronous operation, then *aio\_return()* may be  
 20474 successfully used to retrieve the return status of that operation.

20475 **RETURN VALUE**

20476 If the asynchronous I/O operation has completed, then the return status, as described for *read()*,  
 20477 *write()*, and *fsync()*, shall be returned. If the asynchronous I/O operation has not yet completed,  
 20478 the results of *aio\_return()* are undefined.

20479 If the *aio\_return()* function fails, it shall return  $-1$  and set *errno* to indicate the error.

20480 **ERRORS**

20481 The *aio\_return()* function may fail if:

20482 [EINVAL] The *aiocbp* argument does not refer to an asynchronous operation whose  
 20483 return status has not yet been retrieved.

20484 **EXAMPLES**

20485 None.

20486 **APPLICATION USAGE**

20487 None.

20488 **RATIONALE**

20489 None.

20490 **FUTURE DIRECTIONS**

20491 None.

20492 **SEE ALSO**

20493 *aio\_cancel()*, *aio\_error()*, *aio\_fsync()*, *aio\_read()*, *aio\_write()*, *close()*, *exec*, *exit()*, *fork()*, *lio\_listio()*,  
 20494 *lseek()*, *read()*

20495 XBD <aio.h>

20496 **CHANGE HISTORY**

20497 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

20498 **Issue 6**

20499 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
 20500 implementation does not support the Asynchronous Input and Output option.

20501 The APPLICATION USAGE section is added.

20502 The [EINVAL] error condition is made optional. This is for consistency with the DESCRIPTION.

20503 **Issue 7**

20504 SD5-XSH-ERN-148 is applied.

20505 The *aio\_return()* function is moved from the Asynchronous Input and Output option to the Base.



**NAME**

aio\_suspend — wait for an asynchronous I/O request

**SYNOPSIS**

```
#include <aio.h>

int aio_suspend(const struct aiocb *const list[], int nent,
               const struct timespec *timeout);
```

**DESCRIPTION**

The *aio\_suspend()* function shall suspend the calling thread until at least one of the asynchronous I/O operations referenced by the *list* argument has completed, until a signal interrupts the function, or, if *timeout* is not NULL, until the time interval specified by *timeout* has passed. If any of the **aiocb** structures in the list correspond to completed asynchronous I/O operations (that is, the error status for the operation is not equal to [EINPROGRESS]) at the time of the call, the function shall return without suspending the calling thread. The *list* argument is an array of pointers to asynchronous I/O control blocks. The *nent* argument indicates the number of elements in the array. Each **aiocb** structure pointed to has been used in initiating an asynchronous I/O request via *aio\_read()*, *aio\_write()*, or *lio\_listio()*. This array may contain null pointers, which are ignored. If this array contains pointers that refer to **aiocb** structures that have not been used in submitting asynchronous I/O, the effect is undefined.

If the time interval indicated in the **timespec** structure pointed to by *timeout* passes before any of the I/O operations referenced by *list* are completed, then *aio\_suspend()* shall return with an error.

MON If the Monotonic Clock option is supported, the clock that shall be used to measure this time interval shall be the CLOCK\_MONOTONIC clock.

**RETURN VALUE**

If the *aio\_suspend()* function returns after one or more asynchronous I/O operations have completed, the function shall return zero. Otherwise, the function shall return a value of -1 and set *errno* to indicate the error.

The application may determine which asynchronous I/O completed by scanning the associated error and return status using *aio\_error()* and *aio\_return()*, respectively.

**ERRORS**

The *aio\_suspend()* function shall fail if:

[EAGAIN] No asynchronous I/O indicated in the list referenced by *list* completed in the time interval indicated by *timeout*.

[EINTR] A signal interrupted the *aio\_suspend()* function. Note that, since each asynchronous I/O operation may possibly provoke a signal when it completes, this error return may be caused by the completion of one (or more) of the very I/O operations being awaited.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

None.

20548 **FUTURE DIRECTIONS**

20549 None.

20550 **SEE ALSO**

20551 *aio\_read()*, *aio\_write()*, *lio\_listio()*

20552 XBD *<aio.h>*

20553 **CHANGE HISTORY**

20554 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

20555 **Issue 6**

20556 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
20557 implementation does not support the Asynchronous Input and Output option.

20558 The APPLICATION USAGE section is added.

20559 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that the  
20560 CLOCK\_MONOTONIC clock, if supported, is used.

20561 **Issue 7**

20562 The *aio\_suspend()* function is moved from the Asynchronous Input and Output option to the  
20563 Base.

20564 **NAME**

20565 aio\_write — asynchronous write to a file

20566 **SYNOPSIS**

20567 #include &lt;aio.h&gt;

20568 int aio\_write(struct aiocb \*aiocbp);

20569 **DESCRIPTION**

20570 The *aio\_write()* function shall write *aiocbp->aio\_nbytes* to the file associated with  
 20571 *aiocbp->aio\_fildes* from the buffer pointed to by *aiocbp->aio\_buf*. The function shall return when  
 20572 the write request has been initiated or, at a minimum, queued to the file or device.

20573 PIO If prioritized I/O is supported for this file, then the asynchronous operation shall be submitted  
 20574 at a priority equal to a base scheduling priority minus *aiocbp->aio\_reqprio*. If Thread Execution  
 20575 Scheduling is not supported, then the base scheduling priority is that of the calling process;  
 20576 PIO TPS otherwise, the base scheduling priority is that of the calling thread.

20577 The *aiocbp* argument may be used as an argument to *aio\_error()* and *aio\_return()* in order to  
 20578 determine the error status and return status, respectively, of the asynchronous operation while it  
 20579 is proceeding.

20580 The *aiocbp* argument points to an **aiocb** structure. If the buffer pointed to by *aiocbp->aio\_buf* or  
 20581 the control block pointed to by *aiocbp* becomes an illegal address prior to asynchronous I/O  
 20582 completion, then the behavior is undefined.

20583 If O\_APPEND is not set for the file descriptor *aio\_fildes*, then the requested operation shall take  
 20584 place at the absolute position in the file as given by *aio\_offset*, as if *lseek()* were called  
 20585 immediately prior to the operation with an *offset* equal to *aio\_offset* and a *whence* equal to  
 20586 SEEK\_SET. If O\_APPEND is set for the file descriptor, or if *aio\_fildes* is associated with a device  
 20587 that is incapable of seeking, write operations append to the file in the same order as the calls  
 20588 were made, except under circumstances described in Section 2.8.2. After a successful call to  
 20589 enqueue an asynchronous I/O operation, the value of the file offset for the file is unspecified.

20590 The *aio\_sigevent* member specifies the notification which occurs when the request is completed.

20591 The *aiocbp->aio\_lio\_opcode* field shall be ignored by *aio\_write()*.

20592 Simultaneous asynchronous operations using the same *aiocbp* produce undefined results.

20593 SIO If synchronized I/O is enabled on the file associated with *aiocbp->aio\_fildes*, the behavior of this  
 20594 function shall be according to the definitions of synchronized I/O data integrity completion, and  
 20595 synchronized I/O file integrity completion.

20596 For any system action that changes the process memory space while an asynchronous I/O is  
 20597 outstanding to the address range being changed, the result of that action is undefined.

20598 For regular files, no data transfer shall occur past the offset maximum established in the open  
 20599 file description associated with *aiocbp->aio\_fildes*.

20600 **RETURN VALUE**

20601 The *aio\_write()* function shall return the value zero if the I/O operation is successfully queued;  
 20602 otherwise, the function shall return the value -1 and set *errno* to indicate the error.

20603 **ERRORS**

20604 The *aio\_write()* function shall fail if:

20605 [EAGAIN] The requested asynchronous I/O operation was not queued due to system  
 20606 resource limitations.

20607 Each of the following conditions may be detected synchronously at the time of the call to

20608 *aio\_write()*, or asynchronously. If any of the conditions below are detected synchronously, the  
 20609 *aio\_write()* function shall return `-1` and set *errno* to the corresponding value. If any of the  
 20610 conditions below are detected asynchronously, the return status of the asynchronous operation  
 20611 shall be set to `-1`, and the error status of the asynchronous operation is set to the corresponding  
 20612 value.

20613 [EBADF] The *aioctx->aio\_fildes* argument is not a valid file descriptor open for writing.

20614 [EINVAL] The file offset value implied by *aioctx->aio\_offset* would be invalid,  
 20615 PIO *aioctx->aio\_reqprio* is not a valid value, or *aioctx->aio\_nbytes* is an invalid  
 20616 value.

20617 In the case that the *aio\_write()* successfully queues the I/O operation, the return status of the  
 20618 asynchronous operation shall be one of the values normally returned by the *write()* function call.  
 20619 If the operation is successfully queued but is subsequently canceled or encounters an error, the  
 20620 error status for the asynchronous operation contains one of the values normally set by the  
 20621 *write()* function call, or one of the following:

20622 [EBADF] The *aioctx->aio\_fildes* argument is not a valid file descriptor open for writing.

20623 [EINVAL] The file offset value implied by *aioctx->aio\_offset* would be invalid.

20624 [ECANCELED] The requested I/O was canceled before the I/O completed due to an explicit  
 20625 *aio\_cancel()* request.

20626 The following condition may be detected synchronously or asynchronously:

20627 [EFBIG] The file is a regular file, *aioctx->aio\_nbytes* is greater than 0, and the starting  
 20628 offset in *aioctx->aio\_offset* is at or beyond the offset maximum in the open file  
 20629 description associated with *aioctx->aio\_fildes*.

## 20630 EXAMPLES

20631 None.

## 20632 APPLICATION USAGE

20633 None.

## 20634 RATIONALE

20635 None.

## 20636 FUTURE DIRECTIONS

20637 None.

## 20638 SEE ALSO

20639 Section 2.8.2 (on page 503), *aio\_cancel()*, *aio\_error()*, *aio\_read()*, *aio\_return()*, *close()*, *exec*, *exit()*,  
 20640 *fork()*, *lio\_listio()*, *lseek()*, *write()*

20641 XBD <aio.h>

## 20642 CHANGE HISTORY

20643 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

20644 Large File Summit extensions are added.

## 20645 Issue 6

20646 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
 20647 implementation does not support the Asynchronous Input and Output option.

20648 The APPLICATION USAGE section is added.

20649 The following new requirements on POSIX implementations derive from alignment with the

- 20650 Single UNIX Specification:
- 20651 • In the DESCRIPTION, text is added to indicate that for regular files no data transfer occurs
  - 20652 past the offset maximum established in the open file description associated with
  - 20653 *aiocbp*→*aio\_fildes*.
  - 20654 • The [EFBIG] error is added as part of the large file support extensions.
- 20655 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/14 is applied, rewording the
- 20656 DESCRIPTION when prioritized I/O is supported to account for threads, and removing the
- 20657 words “to the calling process” in the RETURN VALUE section.
- 20658 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/15 is applied, updating the [EINVAL]
- 20659 error, so that detection of an [EINVAL] error for an invalid value of *aiocbp*→*aio\_reqprio* is only
- 20660 required if the Prioritized Input and Output option is supported.
- 20661 **Issue 7**
- 20662 Austin Group Interpretation 1003.1-2001 #082 is applied.
- 20663 The *aio\_write()* function is moved from the Asynchronous Input and Output option to the Base.
- 20664 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0028 [317] is applied.

20665 **NAME**

20666 alarm — schedule an alarm signal

20667 **SYNOPSIS**

20668 #include &lt;unistd.h&gt;

20669 unsigned alarm(unsigned *seconds*);20670 **DESCRIPTION**

20671 The *alarm()* function shall cause the system to generate a SIGALRM signal for the process after  
20672 the number of realtime seconds specified by *seconds* have elapsed. Processor scheduling delays  
20673 may prevent the process from handling the signal as soon as it is generated.

20674 If *seconds* is 0, a pending alarm request, if any, is canceled.

20675 Alarm requests are not stacked; only one SIGALRM generation can be scheduled in this manner.  
20676 If the SIGALRM signal has not yet been generated, the call shall result in rescheduling the time  
20677 at which the SIGALRM signal is generated.

20678 XSI Interactions between *alarm()* and *setitimer()* are unspecified.

20679 **RETURN VALUE**

20680 If there is a previous *alarm()* request with time remaining, *alarm()* shall return a non-zero value  
20681 that is the number of seconds until the previous request would have generated a SIGALRM  
20682 signal. Otherwise, *alarm()* shall return 0.

20683 **ERRORS**

20684 The *alarm()* function is always successful, and no return value is reserved to indicate an error.

20685 **EXAMPLES**

20686 None.

20687 **APPLICATION USAGE**

20688 The *fork()* function clears pending alarms in the child process. A new process image created by  
20689 one of the *exec* functions inherits the time left to an alarm signal in the image of the old process.

20690 Application developers should note that the type of the argument *seconds* and the return value of  
20691 *alarm()* is **unsigned**. That means that a Strictly Conforming POSIX System Interfaces  
20692 Application cannot pass a value greater than the minimum guaranteed value for {UINT\_MAX},  
20693 which the ISO C standard sets as 65 535, and any application passing a larger value is restricting  
20694 its portability. A different type was considered, but historical implementations, including those  
20695 with a 16-bit **int** type, consistently use either **unsigned** or **int**.

20696 Application developers should be aware of possible interactions when the same process uses  
20697 both the *alarm()* and *sleep()* functions.

20698 **RATIONALE**

20699 Many historical implementations (including Version 7 and System V) allow an alarm to occur up  
20700 to a second early. Other implementations allow alarms up to half a second or one clock tick  
20701 early or do not allow them to occur early at all. The latter is considered most appropriate, since it  
20702 gives the most predictable behavior, especially since the signal can always be delayed for an  
20703 indefinite amount of time due to scheduling. Applications can thus choose the *seconds* argument  
20704 as the minimum amount of time they wish to have elapse before the signal.

20705 The term “realtime” here and elsewhere (*sleep()*, *times()*) is intended to mean “wall clock” time  
20706 as common English usage, and has nothing to do with “realtime operating systems”. It is in  
20707 contrast to *virtual time*, which could be misinterpreted if just *time* were used.

20708 In some implementations, including 4.3 BSD, very large values of the *seconds* argument are  
20709 silently rounded down to an implementation-specific maximum value. This maximum is large

20710 enough (to the order of several months) that the effect is not noticeable.

20711 There were two possible choices for alarm generation in multi-threaded applications: generation  
20712 for the calling thread or generation for the process. The first option would not have been  
20713 particularly useful since the alarm state is maintained on a per-process basis and the alarm that  
20714 is established by the last invocation of *alarm()* is the only one that would be active.

20715 Furthermore, allowing generation of an asynchronous signal for a thread would have  
20716 introduced an exception to the overall signal model. This requires a compelling reason in order  
20717 to be justified.

#### 20718 FUTURE DIRECTIONS

20719 None.

#### 20720 SEE ALSO

20721 *alarm()*, *exec*, *fork()*, *getitimer()*, *pause()*, *sigaction()*, *sleep()*

20722 XBD [\*<signal.h>\*](#), [\*<unistd.h>\*](#)

#### 20723 CHANGE HISTORY

20724 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 20725 Issue 6

20726 The following new requirements on POSIX implementations derive from alignment with the  
20727 Single UNIX Specification:

- 20728 • The DESCRIPTION is updated to indicate that interactions with the *setitimer()*, *ualarm()*,  
20729 and *usleep()* functions are unspecified.

20730 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/16 is applied, replacing “an  
20731 implementation-defined maximum value” with “an implementation-specific maximum value”  
20732 in the RATIONALE.



**NAME**

alphasort, scandir — scan a directory

**SYNOPSIS**

```
#include <dirent.h>

int alphasort(const struct dirent **d1, const struct dirent **d2);
int scandir(const char *dir, struct dirent ***namelist,
            int (*sel)(const struct dirent *),
            int (*compar)(const struct dirent **, const struct dirent **));
```

**DESCRIPTION**

The *alphasort()* function can be used as the comparison function for the *scandir()* function to sort the directory entries, *d1* and *d2*, into alphabetical order. Sorting happens as if by calling the *strcoll()* function on the *d\_name* element of the **dirent** structures passed as the two parameters. If the *strcoll()* function fails, the return value of *alphasort()* is unspecified.

The *alphasort()* function shall not change the setting of *errno* if successful. Since no return value is reserved to indicate an error, an application wishing to check for error situations should set *errno* to 0, then call *alphasort()*, then check *errno*.

The *scandir()* function shall scan the directory *dir*, calling the function referenced by *sel* on each directory entry. Entries for which the function referenced by *sel* returns non-zero shall be stored in strings allocated as if by a call to *malloc()*, and sorted as if by a call to *qsort()* with the comparison function *compar*, except that *compar* need not provide total ordering. The strings are collected in array *namelist* which shall be allocated as if by a call to *malloc()*. If *sel* is a null pointer, all entries shall be selected. If the comparison function *compar* does not provide total ordering, the order in which the directory entries are stored is unspecified.

**RETURN VALUE**

Upon successful completion, the *alphasort()* function shall return an integer greater than, equal to, or less than 0, according to whether the name of the directory entry pointed to by *d1* is lexically greater than, equal to, or less than the directory pointed to by *d2* when both are interpreted as appropriate to the current locale. There is no return value reserved to indicate an error.

Upon successful completion, the *scandir()* function shall return the number of entries in the array and a pointer to the array through the parameter *namelist*. Otherwise, the *scandir()* function shall return -1.

**ERRORS**

The *scandir()* function shall fail if:

- |                |  |
|----------------|--|
| [EACCES]       | Search permission is denied for the component of the path prefix of <i>dir</i> or read permission is denied for <i>dir</i> . |
| [ELOOP]        | A loop exists in symbolic links encountered during resolution of the <i>dir</i> argument.                                    |
| [ENAMETOOLONG] | The length of a component of a pathname is longer than {NAME_MAX}.   |
| [ENOENT]       | A component of <i>dir</i> does not name an existing directory or <i>dir</i> is an empty string.                              |
| [ENOMEM]       | Insufficient storage space is available.   |



20776	[ENOTDIR]	A component of <i>dir</i> names an existing file that is neither a directory nor a symbolic link to a directory.
20777		
20778	[EOVERFLOW]	One of the values to be returned or passed to a callback function cannot be represented correctly.
20779		
20780	The <i>scandir()</i> function may fail if:	
20781	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the <i>dir</i> argument.
20782		
20783	[EMFILE]	All file descriptors available to the process are currently open.
20784	[ENAMETOOLONG]	
20785		The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.
20786		
20787		
20788	[ENFILE]	Too many files are currently open in the system.

**EXAMPLES**

An example to print the files in the current directory:

```
#include <dirent.h>
#include <stdio.h>
#include <stdlib.h>
...
struct dirent **namelist;
int i,n;

n = scandir(".", &namelist, 0, alphasort);
if (n < 0)
    perror("scandir");
else {
    for (i = 0; i < n; i++) {
        printf("%s\n", namelist[i]->d_name);
        free(namelist[i]);
    }
    free(namelist);
}
```

**APPLICATION USAGE**

If *dir* contains filenames that do not form character strings, or which contain characters outside the domain of the collating sequence of the current locale, the *alphasort()* function need not provide a total ordering. This condition is not possible if all filenames within the directory consist only of characters from the portable filename character set.

The *scandir()* function may allocate dynamic storage during its operation. If *scandir()* is forcibly terminated, such as by *longjmp()* or *siglongjmp()* being executed by the function pointed to by *sel* or *compar*, or by an interrupt routine, *scandir()* does not have a chance to free that storage, so it remains permanently allocated. A safe way to handle interrupts is to store the fact that an interrupt has occurred, then wait until *scandir()* returns to act on the interrupt.

For functions that allocate memory as if by *malloc()*, the application should release such memory when it is no longer required by a call to *free()*. For *scandir()*, this is *namelist* (including all of the individual strings in *namelist*).

20821 **RATIONALE**

20822 None.

20823 **FUTURE DIRECTIONS**

20824 None.

20825 **SEE ALSO**20826 *qsort()*, *strcoll()*

20827 XBD &lt;dirent.h&gt;

20828 **CHANGE HISTORY**

20829 First released in Issue 7.

20830 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0029 [324], XSH/TC1-2008/0030 [404],  
20831 XSH/TC1-2008/0031 [393], and XSH/TC1-2008/0032 [291] are applied.

20832 **NAME**

20833 asctime, asctime\_r — convert date and time to a string

20834 **SYNOPSIS**

```
20835 OB      #include <time.h>
20836         char *asctime(const struct tm *timeptr);
20837 OB CX    char *asctime_r(const struct tm *restrict tm, char *restrict buf);
```

20838 **DESCRIPTION**

20839 CX For *asctime()*: The functionality described on this reference page is aligned with the ISO C  
 20840 standard. Any conflict between the requirements described here and the ISO C standard is  
 20841 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

20842 The *asctime()* function shall convert the broken-down time in the structure pointed to by *timeptr*  
 20843 into a string in the form:

20844 Sun Sep 16 01:03:52 1973\n\0

20845 using the equivalent of the following algorithm:

```
20846 char *asctime(const struct tm *timeptr)
20847 {
20848     static char wday_name[7][3] = {
20849         "Sun", "Mon", "Tue", "Wed", "Thu", "Fri", "Sat"
20850     };
20851     static char mon_name[12][3] = {
20852         "Jan", "Feb", "Mar", "Apr", "May", "Jun",
20853         "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"
20854     };
20855     static char result[26];
20856     sprintf(result, "%.3s %.3s%3d %.2d:%.2d:%.2d %d\n",
20857         wday_name[timeptr->tm_wday],
20858         mon_name[timeptr->tm_mon],
20859         timeptr->tm_mday, timeptr->tm_hour,
20860         timeptr->tm_min, timeptr->tm_sec,
20861         1900 + timeptr->tm_year);
20862     return result;
20863 }
```

20864 However, the behavior is undefined if *timeptr->tm\_wday* or *timeptr->tm\_mon* are not within the  
 20865 normal ranges as defined in **<time.h>**, or if *timeptr->tm\_year* exceeds {INT\_MAX}-1990, or if the  
 20866 above algorithm would attempt to generate more than 26 bytes of output (including the  
 20867 terminating null).

20868 The **tm** structure is defined in the **<time.h>** header.

20869 CX The *asctime()*, *ctime()*, *gmtime()*, and *localtime()* functions shall return values in one of two static  
 20870 objects: a broken-down time structure and an array of type **char**. Execution of any of the  
 20871 functions may overwrite the information returned in either of these objects by any of the other  
 20872 functions.

20873 The *asctime()* function need not be thread-safe.

20874 The *asctime\_r()* function shall convert the broken-down time in the structure pointed to by *tm*  
 20875 into a string (of the same form as that returned by *asctime()*, and with the same undefined  
 20876 behavior when input or output is out of range) that is placed in the user-supplied buffer pointed

20877 to by *buf* (which shall contain at least 26 bytes) and then return *buf*.

#### 20878 RETURN VALUE

20879 CX Upon successful completion, *asctime()* shall return a pointer to the string. If the function is  
20880 unsuccessful, it shall return NULL.

20881 Upon successful completion, *asctime\_r()* shall return a pointer to a character string containing  
20882 the date and time. This string is pointed to by the argument *buf*. If the function is unsuccessful,  
20883 it shall return NULL.

#### 20884 ERRORS

20885 No errors are defined.

#### 20886 EXAMPLES

20887 None.

#### 20888 APPLICATION USAGE

20889 These functions are included only for compatibility with older implementations. They have  
20890 undefined behavior if the resulting string would be too long, so the use of these functions  
20891 should be discouraged. On implementations that do not detect output string length overflow, it  
20892 is possible to overflow the output buffers in such a way as to cause applications to fail, or  
20893 possible system security violations. Also, these functions do not support localized date and time  
20894 formats. To avoid these problems, applications should use *strftime()* to generate strings from  
20895 broken-down times.

20896 Values for the broken-down time structure can be obtained by calling *gmtime()* or *localtime()*.

20897 The *asctime\_r()* function is thread-safe and shall return values in a user-supplied buffer instead  
20898 of possibly using a static data area that may be overwritten by each call.

#### 20899 RATIONALE

20900 The standard developers decided to mark the *asctime()* and *asctime\_r()* functions obsolescent  
20901 even though *asctime()* is in the ISO C standard due to the possibility of buffer overflow. The  
20902 ISO C standard also provides the *strftime()* function which can be used to avoid these problems.

#### 20903 FUTURE DIRECTIONS

20904 These functions may be removed in a future version.

#### 20905 SEE ALSO

20906 *clock()*, *ctime()*, *difftime()*, *gmtime()*, *localtime()*, *mktime()*, *strftime()*, *strptime()*, *time()*, *utime()*

20907 XBD <time.h>

#### 20908 CHANGE HISTORY

20909 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 20910 Issue 5

20911 Normative text previously in the APPLICATION USAGE section is moved to the  
20912 DESCRIPTION.

20913 The *asctime\_r()* function is included for alignment with the POSIX Threads Extension.

20914 A note indicating that the *asctime()* function need not be reentrant is added to the  
20915 DESCRIPTION.

#### 20916 Issue 6

20917 The *asctime\_r()* function is marked as part of the Thread-Safe Functions option.

20918 Extensions beyond the ISO C standard are marked.

20919 The APPLICATION USAGE section is updated to include a note on the thread-safe function and

- 20920 its avoidance of possibly using a static data area.
- 20921 The DESCRIPTION of *asctime\_r()* is updated to describe the format of the string returned.
- 20922 The **restrict** keyword is added to the *asctime\_r()* prototype for alignment with the  
20923 ISO/IEC 9899:1999 standard
- 20924 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/17 is applied, adding the CX extension in  
20925 the RETURN VALUE section requiring that if the *asctime()* function is unsuccessful it returns  
20926 NULL.
- 20927 **Issue 7**
- 20928 Austin Group Interpretation 1003.1-2001 #053 is applied, marking these functions obsolescent.
- 20929 Austin Group Interpretation 1003.1-2001 #156 is applied.
- 20930 The *asctime\_r()* function is moved from the Thread-Safe Functions option to the Base.
- 20931 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0033 [86,429] is applied.

20932 **NAME**

20933 asin, asinf, asinl — arc sine function

20934 **SYNOPSIS**

```
20935 #include <math.h>
20936 double asin(double x);
20937 float asinf(float x);
20938 long double asinl(long double x);
```

20939 **DESCRIPTION**

20940 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 20941 conflict between the requirements described here and the ISO C standard is unintentional. This  
 20942 volume of POSIX.1-2008 defers to the ISO C standard.

20943 These functions shall compute the principal value of the arc sine of their argument  $x$ . The value  
 20944 of  $x$  should be in the range  $[-1,1]$ .

20945 An application wishing to check for error situations should set *errno* to zero and call  
 20946 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 20947 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 20948 zero, an error has occurred.

20949 **RETURN VALUE**

20950 Upon successful completion, these functions shall return the arc sine of  $x$ , in the range  
 20951  $[-\pi/2, \pi/2]$  radians.

20952 MX For finite values of  $x$  not in the range  $[-1,1]$ , a domain error shall occur, and either a NaN (if  
 20953 supported), or an implementation-defined value shall be returned.

20954 MX If  $x$  is NaN, a NaN shall be returned.

20955 If  $x$  is  $\pm 0$ ,  $x$  shall be returned.

20956 If  $x$  is  $\pm \text{Inf}$ , a domain error shall occur, and a NaN shall be returned.

20957 If  $x$  is subnormal, a range error may occur

20958 MXX and  $x$  should be returned.

20959 MX If  $x$  is not returned, *asin()*, *asinf()*, and *asinl()* shall return an implementation-defined value no  
 20960 greater in magnitude than DBL\_MIN, FLT\_MIN, and LDBL\_MIN, respectively.

20961 **ERRORS**

20962 These functions shall fail if:

20963 MX Domain Error The  $x$  argument is finite and is not in the range  $[-1,1]$ , or is  $\pm \text{Inf}$ .

20964 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 20965 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 20966 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 20967 shall be raised.

20968 These functions may fail if:

20969 MX Range Error The value of  $x$  is subnormal.

20970 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 20971 then *errno* shall be set to [ERANGE]. If the integer expression  
 20972 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 20973 floating-point exception shall be raised.

20974 **EXAMPLES**

20975 None.

20976 **APPLICATION USAGE**

20977 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 20978 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

20979 **RATIONALE**

20980 None.

20981 **FUTURE DIRECTIONS**

20982 None.

20983 **SEE ALSO**20984 *feclearexcept()*, *fetestexcept()*, *isnan()*, *sin()*20985 XBD Section 4.20 (on page 117), **<math.h>**20986 **CHANGE HISTORY**

20987 First released in Issue 1. Derived from Issue 1 of the SVID.

20988 **Issue 5**

20989 The DESCRIPTION is updated to indicate how an application should check for an error. This  
 20990 text was previously published in the APPLICATION USAGE section.

20991 **Issue 6**20992 The *asinf()* and *asinl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

20993 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
 20994 revised to align with the ISO/IEC 9899:1999 standard.

20995 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
 20996 marked.

20997 **Issue 7**

20998 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0034 [320] and XSH/TC1-2008/0035  
 20999 [68] are applied.

21000 **NAME**

21001 asinh, asinhf, asinhl — inverse hyperbolic sine functions

21002 **SYNOPSIS**

```
21003 #include <math.h>
21004 double asinh(double x);
21005 float asinhf(float x);
21006 long double asinhl(long double x);
```

21007 **DESCRIPTION**

21008 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 21009 conflict between the requirements described here and the ISO C standard is unintentional. This  
 21010 volume of POSIX.1-2008 defers to the ISO C standard.

21011 These functions shall compute the inverse hyperbolic sine of their argument  $x$ .

21012 An application wishing to check for error situations should set *errno* to zero and call  
 21013 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 21014 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 21015 zero, an error has occurred.

21016 **RETURN VALUE**

21017 Upon successful completion, these functions shall return the inverse hyperbolic sine of their  
 21018 argument.

21019 MX If  $x$  is NaN, a NaN shall be returned.

21020 If  $x$  is  $\pm 0$ , or  $\pm \text{Inf}$ ,  $x$  shall be returned.

21021 If  $x$  is subnormal, a range error may occur

21022 MXX and  $x$  should be returned.

21023 MX If  $x$  is not returned, *asinh*(), *asinhf*(), and *asinhl*() shall return an implementation-defined value  
 21024 no greater in magnitude than DBL\_MIN, FLT\_MIN, and LDBL\_MIN, respectively.

21025 **ERRORS**

21026 These functions may fail if:

21027 MX **Range Error** The value of  $x$  is subnormal.

21028 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 21029 then *errno* shall be set to [ERANGE]. If the integer expression  
 21030 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 21031 floating-point exception shall be raised.

21032 **EXAMPLES**

21033 None.

21034 **APPLICATION USAGE**

21035 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 21036 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

21037 **RATIONALE**

21038 None.

21039 **FUTURE DIRECTIONS**

21040 None.



21041 **SEE ALSO**21042 *feclearexcept()*, *fetestexcept()*, *sinh()*21043 XBD Section 4.20 (on page 117), **<math.h>**21044 **CHANGE HISTORY**

21045 First released in Issue 4, Version 2.

21046 **Issue 5**

21047 Moved from X/OPEN UNIX extension to BASE.

21048 **Issue 6**21049 The *asinh()* function is no longer marked as an extension.21050 The *asinhf()* and *asinhll()* functions are added for alignment with the ISO/IEC 9899:1999  
21051 standard.21052 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
21053 revised to align with the ISO/IEC 9899:1999 standard.21054 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
21055 marked.21056 **Issue 7**

21057 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0036 [68] is applied.

21058 **NAME**

21059        asinl — arc sine function

21060 **SYNOPSIS**21061        `#include <math.h>`21062        `long double asinl(long double x);`21063 **DESCRIPTION**21064        Refer to *asin()*.

21065 **NAME**

21066       assert — insert program diagnostics

21067 **SYNOPSIS**

21068       #include &lt;assert.h&gt;

21069       void assert(scalar expression);

21070 **DESCRIPTION**

21071 **CX**       The functionality described on this reference page is aligned with the ISO C standard. Any  
 21072 conflict between the requirements described here and the ISO C standard is unintentional. This  
 21073 volume of POSIX.1-2008 defers to the ISO C standard.

21074       The *assert()* macro shall insert diagnostics into programs; it shall expand to a **void** expression.  
 21075 When it is executed, if *expression* (which shall have a **scalar** type) is false (that is, compares equal  
 21076 to 0), *assert()* shall write information about the particular call that failed on *stderr* and shall call  
 21077 *abort()*.

21078       The information written about the call that failed shall include the text of the argument, the  
 21079 name of the source file, the source file line number, and the name of the enclosing function; the  
 21080 latter are, respectively, the values of the preprocessing macros `__FILE__` and `__LINE__` and of  
 21081 the identifier `__func__`.

21082       Forcing a definition of the name `NDEBUG`, either from the compiler command line or with the  
 21083 preprocessor control statement **#define** `NDEBUG` ahead of the **#include** `<assert.h>` statement,  
 21084 shall stop assertions from being compiled into the program.

21085 **RETURN VALUE**21086       The *assert()* macro shall not return a value.21087 **ERRORS**

21088       No errors are defined.

21089 **EXAMPLES**

21090       None.

21091 **APPLICATION USAGE**

21092       None.

21093 **RATIONALE**

21094       None.

21095 **FUTURE DIRECTIONS**

21096       None.

21097 **SEE ALSO**21098       *abort()*, *stdin*21099       XBD `<assert.h>`21100 **CHANGE HISTORY**

21101       First released in Issue 1. Derived from Issue 1 of the SVID.

21102 **Issue 6**

21103       The prototype for the *expression* argument to *assert()* is changed from **int** to **scalar** for alignment  
 21104 with the ISO/IEC 9899:1999 standard.

21105       The DESCRIPTION of *assert()* is updated for alignment with the ISO/IEC 9899:1999 standard.

21106 **NAME**

21107 atan, atanf, atanl — arc tangent function

21108 **SYNOPSIS**

```
21109 #include <math.h>
21110 double atan(double x);
21111 float atanf(float x);
21112 long double atanl(long double x);
```

21113 **DESCRIPTION**

21114 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 21115 conflict between the requirements described here and the ISO C standard is unintentional. This  
 21116 volume of POSIX.1-2008 defers to the ISO C standard.

21117 These functions shall compute the principal value of the arc tangent of their argument  $x$ .

21118 An application wishing to check for error situations should set *errno* to zero and call  
 21119 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 21120 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 21121 zero, an error has occurred.

21122 **RETURN VALUE**

21123 Upon successful completion, these functions shall return the arc tangent of  $x$  in the range  
 21124  $[-\pi/2, \pi/2]$  radians.

21125 MX If  $x$  is NaN, a NaN shall be returned.

21126 If  $x$  is  $\pm 0$ ,  $x$  shall be returned.

21127 If  $x$  is  $\pm \text{Inf}$ ,  $\pm \pi/2$  shall be returned.

21128 If  $x$  is subnormal, a range error may occur

21129 MXX and  $x$  should be returned.

21130 MX If  $x$  is not returned, *atan()*, *atanf()*, and *atanl()* shall return an implementation-defined value no  
 21131 greater in magnitude than DBL\_MIN, FLT\_MIN, and LDBL\_MIN, respectively.

21132 **ERRORS**

21133 These functions may fail if:

21134 MX **Range Error** The value of  $x$  is subnormal.

21135 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 21136 then *errno* shall be set to [ERANGE]. If the integer expression  
 21137 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 21138 floating-point exception shall be raised.

21139 **EXAMPLES**

21140 None.

21141 **APPLICATION USAGE**

21142 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 21143 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

21144 **RATIONALE**

21145 None.

21146 **FUTURE DIRECTIONS**

21147 None.

21148 **SEE ALSO**21149 *atan2()*, *feclearexcept()*, *fetestexcept()*, *isnan()*, *tan()*21150 XBD [Section 4.20](#) (on page 117), **<math.h>**21151 **CHANGE HISTORY**

21152 First released in Issue 1. Derived from Issue 1 of the SVID.

21153 **Issue 5**21154 The DESCRIPTION is updated to indicate how an application should check for an error. This  
21155 text was previously published in the APPLICATION USAGE section.21156 **Issue 6**21157 The *atanf()* and *atanl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.21158 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
21159 revised to align with the ISO/IEC 9899:1999 standard.21160 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
21161 marked.21162 **Issue 7**

21163 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0037 [68] is applied.

21164 **NAME**

21165 atan2, atan2f, atan2l — arc tangent functions

21166 **SYNOPSIS**

```
21167 #include <math.h>
21168 double atan2(double y, double x);
21169 float atan2f(float y, float x);
21170 long double atan2l(long double y, long double x);
```

21171 **DESCRIPTION**

21172 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 21173 conflict between the requirements described here and the ISO C standard is unintentional. This  
 21174 volume of POSIX.1-2008 defers to the ISO C standard.

21175 These functions shall compute the principal value of the arc tangent of  $y/x$ , using the signs of  
 21176 both arguments to determine the quadrant of the return value.

21177 An application wishing to check for error situations should set *errno* to zero and call  
 21178 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 21179 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 21180 zero, an error has occurred.

21181 **RETURN VALUE**

21182 Upon successful completion, these functions shall return the arc tangent of  $y/x$  in the range  
 21183  $[-\pi, \pi]$  radians.

21184 If  $y$  is  $\pm 0$  and  $x$  is  $< 0$ ,  $\pm\pi$  shall be returned.

21185 If  $y$  is  $\pm 0$  and  $x$  is  $> 0$ ,  $\pm 0$  shall be returned.

21186 If  $y$  is  $< 0$  and  $x$  is  $\pm 0$ ,  $-\pi/2$  shall be returned.

21187 If  $y$  is  $> 0$  and  $x$  is  $\pm 0$ ,  $\pi/2$  shall be returned.

21188 If  $x$  is 0, a pole error shall not occur.

21189 MX If either  $x$  or  $y$  is NaN, a NaN shall be returned.

21190 If the correct value would cause underflow, a range error may occur, and *atan*(), *atan2f*(), and  
 21191 *atan2l*() shall return an implementation-defined value no greater in magnitude than DBL\_MIN,  
 21192 FLT\_MIN, and LDBL\_MIN, respectively.

21193 MXX If the IEC 60559 Floating-Point option is supported,  $y/x$  should be returned.

21194 MX If  $y$  is  $\pm 0$  and  $x$  is  $-0$ ,  $\pm\pi$  shall be returned.

21195 If  $y$  is  $\pm 0$  and  $x$  is  $+0$ ,  $\pm 0$  shall be returned.

21196 For finite values of  $\pm y > 0$ , if  $x$  is  $-\text{Inf}$ ,  $\pm\pi$  shall be returned.

21197 For finite values of  $\pm y > 0$ , if  $x$  is  $+\text{Inf}$ ,  $\pm 0$  shall be returned.

21198 For finite values of  $x$ , if  $y$  is  $\pm\text{Inf}$ ,  $\pm\pi/2$  shall be returned.

21199 If  $y$  is  $\pm\text{Inf}$  and  $x$  is  $-\text{Inf}$ ,  $\pm 3\pi/4$  shall be returned.

21200 If  $y$  is  $\pm\text{Inf}$  and  $x$  is  $+\text{Inf}$ ,  $\pm\pi/4$  shall be returned.

21201 If both arguments are 0, a domain error shall not occur.

21202 **ERRORS**

21203 These functions may fail if:

21204 MX **Range Error** The result underflows.

21205 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 21206 then *errno* shall be set to [ERANGE]. If the integer expression  
 21207 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 21208 floating-point exception shall be raised.

21209 **EXAMPLES**21210 **Converting Cartesian to Polar Coordinates System**

21211 The function below uses *atan2()* to convert a 2d vector expressed in cartesian coordinates (*x,y*) to  
 21212 the polar coordinates (*rho,theta*). There are other ways to compute the angle *theta*, using *asin()*  
 21213 *acos()*, or *atan()*. However, *atan2()* presents here two advantages:

- 21214 • The angle's quadrant is automatically determined.
- 21215 • The singular cases (0,*y*) are taken into account.

21216 Finally, this example uses *hypot()* rather than *sqrt()* since it is better for special cases; see *hypot()*  
 21217 for more information.

```
21218 #include <math.h>
21219 void
21220 cartesian_to_polar(const double x, const double y,
21221                  double *rho, double *theta
21222                  )
21223 {
21224     *rho    = hypot (x,y); /* better than sqrt(x*x+y*y) */
21225     *theta  = atan2 (y,x);
21226 }
```

21227 **APPLICATION USAGE**

21228 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 21229 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

21230 **RATIONALE**

21231 None.

21232 **FUTURE DIRECTIONS**

21233 None.

21234 **SEE ALSO**21235 *acos()*, *asin()*, *atan()*, *feclearexcept()*, *fetestexcept()*, *hypot()*, *isnan()*, *sqrt()*, *tan()*21236 XBD Section 4.20 (on page 117), **<math.h>**21237 **CHANGE HISTORY**

21238 First released in Issue 1. Derived from Issue 1 of the SVID.

21239 **Issue 5**

21240 The DESCRIPTION is updated to indicate how an application should check for an error. This  
 21241 text was previously published in the APPLICATION USAGE section.

21242 **Issue 6**

21243 The *atan2f()* and *atan2l()* functions are added for alignment with the ISO/IEC 9899:1999  
21244 standard.

21245 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
21246 revised to align with the ISO/IEC 9899:1999 standard, and the IEC 60559:1989 standard  
21247 floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

21248 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/18 is applied, adding to the EXAMPLES  
21249 section.

21250 **Issue 7**

21251 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0038 [68,428] is applied.



21252 **NAME**

21253       atanf — arc tangent function

21254 **SYNOPSIS**

21255       #include &lt;math.h&gt;

21256       float atanf(float x);

21257 **DESCRIPTION**21258       Refer to *atan()*.

21259 **NAME**

21260 atanh, atanhf, atanhf — inverse hyperbolic tangent functions

21261 **SYNOPSIS**

```
21262 #include <math.h>
21263 double atanh(double x);
21264 float atanhf(float x);
21265 long double atanhf(long double x);
```

21266 **DESCRIPTION**

21267 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 21268 conflict between the requirements described here and the ISO C standard is unintentional. This  
 21269 volume of POSIX.1-2008 defers to the ISO C standard.

21270 These functions shall compute the inverse hyperbolic tangent of their argument  $x$ .

21271 An application wishing to check for error situations should set *errno* to zero and call  
 21272 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 21273 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 21274 zero, an error has occurred.

21275 **RETURN VALUE**

21276 Upon successful completion, these functions shall return the inverse hyperbolic tangent of their  
 21277 argument.

21278 If  $x$  is  $\pm 1$ , a pole error shall occur, and *atanh*(), *atanhf*(), and *atanf*() shall return the value of the  
 21279 macro HUGE\_VAL, HUGE\_VALF, and HUGE\_VALL, respectively, with the same sign as the  
 21280 correct value of the function.

21281 MX For finite  $|x| > 1$ , a domain error shall occur, and either a NaN (if supported), or an  
 21282 implementation-defined value shall be returned.

21283 MX If  $x$  is NaN, a NaN shall be returned.

21284 If  $x$  is  $\pm 0$ ,  $x$  shall be returned.

21285 If  $x$  is  $\pm \text{Inf}$ , a domain error shall occur, and a NaN shall be returned.

21286 If  $x$  is subnormal, a range error may occur

21287 MXX and  $x$  should be returned.

21288 MX If  $x$  is not returned, *atanh*(), *atanhf*(), and *atanf*() shall return an implementation-defined value  
 21289 no greater in magnitude than DBL\_MIN, FLT\_MIN, and LDBL\_MIN, respectively.

21290 **ERRORS**

21291 These functions shall fail if:

21292 MX Domain Error The  $x$  argument is finite and not in the range  $[-1, 1]$ , or is  $\pm \text{Inf}$ .

21293 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 21294 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 21295 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 21296 shall be raised.

21297 Pole Error The  $x$  argument is  $\pm 1$ .

21298 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 21299 then *errno* shall be set to [ERANGE]. If the integer expression  
 21300 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the divide-by-zero  
 21301 floating-point exception shall be raised.

21302 These functions may fail if:

21303	MX	<b>Range Error</b>	The value of $x$ is subnormal.
21304			If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
21305			then <i>errno</i> shall be set to [ERANGE]. If the integer expression
21306			$(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the underflow
21307			floating-point exception shall be raised.

#### 21308 EXAMPLES

21309 None.

#### 21310 APPLICATION USAGE

21311 On error, the expressions  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$  and  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$  are independent of each other, but at least one of them must be non-zero.

#### 21313 RATIONALE

21314 None.

#### 21315 FUTURE DIRECTIONS

21316 None.

#### 21317 SEE ALSO

21318 *feclearexcept()*, *fetestexcept()*, *tanh()*

21319 XBD Section 4.20 (on page 117), **<math.h>**

#### 21320 CHANGE HISTORY

21321 First released in Issue 4, Version 2.

#### 21322 Issue 5

21323 Moved from X/OPEN UNIX extension to BASE.

#### 21324 Issue 6

21325 The *atanh()* function is no longer marked as an extension.

21326 The *atanhf()* and *atanhl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

21328 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

21330 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

#### 21332 Issue 7

21333 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0039 [320] and XSH/TC1-2008/0040 [680] are applied.

21335 **NAME**

21336       atanl — arc tangent function

21337 **SYNOPSIS**

21338       #include &lt;math.h&gt;

21339       long double atanl(long double x);

21340 **DESCRIPTION**21341       Refer to *atan()*.

21342 **NAME**

21343       atexit — register a function to run at process termination

21344 **SYNOPSIS**

21345       #include &lt;stdlib.h&gt;

21346       int atexit(void (\*func)(void));

21347 **DESCRIPTION**

21348 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 21349 conflict between the requirements described here and the ISO C standard is unintentional. This  
 21350 volume of POSIX.1-2008 defers to the ISO C standard.

21351       The *atexit()* function shall register the function pointed to by *func*, to be called without  
 21352 arguments at normal program termination. At normal program termination, all functions  
 21353 registered by the *atexit()* function shall be called, in the reverse order of their registration, except  
 21354 that a function is called after any previously registered functions that had already been called at  
 21355 the time it was registered. Normal termination occurs either by a call to *exit()* or a return from  
 21356 *main()*.

21357       At least 32 functions can be registered with *atexit()*.

21358 CX       After a successful call to any of the *exec* functions, any functions previously registered by *atexit()*  
 21359 shall no longer be registered.

21360 **RETURN VALUE**21361       Upon successful completion, *atexit()* shall return 0; otherwise, it shall return a non-zero value.21362 **ERRORS**

21363       No errors are defined.

21364 **EXAMPLES**

21365       None.

21366 **APPLICATION USAGE**21367       The functions registered by a call to *atexit()* must return to ensure that all registered functions  
 21368 are called.

21369       The application should call *sysconf()* to obtain the value of {ATEXIT\_MAX}, the number of  
 21370 functions that can be registered. There is no way for an application to tell how many functions  
 21371 have already been registered with *atexit()*.

21372       Since the behavior is undefined if the *exit()* function is called more than once, portable  
 21373 applications calling *atexit()* must ensure that the *exit()* function is not called at normal process  
 21374 termination when all functions registered by the *atexit()* function are called.

21375       All functions registered by the *atexit()* function are called at normal process termination, which  
 21376 occurs by a call to the *exit()* function or a return from *main()* or on the last thread termination,  
 21377 when the behavior is as if the implementation called *exit()* with a zero argument at thread  
 21378 termination time.

21379       If, at normal process termination, a function registered by the *atexit()* function is called and a  
 21380 portable application needs to stop further *exit()* processing, it must call the *\_exit()* function or  
 21381 the *\_Exit()* function or one of the functions which cause abnormal process termination.

21382 **RATIONALE**

21383       None.

21384 **FUTURE DIRECTIONS**

21385 None.

21386 **SEE ALSO**21387 *exec, exit(), sysconf()*

21388 XBD &lt;stdlib.h&gt;

21389 **CHANGE HISTORY**

21390 First released in Issue 4. Derived from the ANSI C standard.

21391 **Issue 6**

21392 Extensions beyond the ISO C standard are marked.

21393 The DESCRIPTION is updated for alignment with the ISO/IEC 9899:1999 standard.

21394 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/19 is applied, adding further clarification  
21395 to the APPLICATION USAGE section.

21396 **NAME**

21397        atof — convert a string to a double-precision number

21398 **SYNOPSIS**

21399        #include &lt;stdlib.h&gt;

21400        double atof(const char \*str);

21401 **DESCRIPTION**

21402 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 21403 conflict between the requirements described here and the ISO C standard is unintentional. This  
 21404 volume of POSIX.1-2008 defers to the ISO C standard.

21405        The call *atof(str)* shall be equivalent to:

21406        strtod(str, (char \*\*)NULL),

21407 except that the handling of errors may differ. If the value cannot be represented, the behavior is  
 21408 undefined.

21409 **RETURN VALUE**21410        The *atof()* function shall return the converted value if the value can be represented.21411 **ERRORS**

21412        No errors are defined.

21413 **EXAMPLES**

21414        None.

21415 **APPLICATION USAGE**

21416        The *atof()* function is subsumed by *strtod()* but is retained because it is used extensively in  
 21417 existing code. If the number is not known to be in range, *strtod()* should be used because *atof()*  
 21418 is not required to perform any error checking.

21419 **RATIONALE**

21420        None.

21421 **FUTURE DIRECTIONS**

21422        None.

21423 **SEE ALSO**21424        *strtod()*

21425        XBD &lt;stdlib.h&gt;

21426 **CHANGE HISTORY**

21427        First released in Issue 1. Derived from Issue 1 of the SVID.

21428 **NAME**

21429       atoi — convert a string to an integer

21430 **SYNOPSIS**

21431       #include &lt;stdlib.h&gt;

21432       int atoi(const char \*str);

21433 **DESCRIPTION**

21434 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
21435       conflict between the requirements described here and the ISO C standard is unintentional. This  
21436       volume of POSIX.1-2008 defers to the ISO C standard.

21437       The call *atoi(str)* shall be equivalent to:

21438       (int) strtol(str, (char \*\*)NULL, 10)

21439       except that the handling of errors may differ. If the value cannot be represented, the behavior is  
21440       undefined.

21441 **RETURN VALUE**21442       The *atoi()* function shall return the converted value if the value can be represented.21443 **ERRORS**

21444       No errors are defined.

21445 **EXAMPLES**21446       **Converting an Argument**

21447       The following example checks for proper usage of the program. If there is an argument and the  
21448       decimal conversion of this argument (obtained using *atoi()*) is greater than 0, then the program  
21449       has a valid number of minutes to wait for an event.

```
21450       #include <stdlib.h>
21451       #include <stdio.h>
21452       ...
21453       int minutes_to_event;
21454       ...
21455       if (argc < 2 || ((minutes_to_event = atoi (argv[1]))) <= 0) {
21456           fprintf(stderr, "Usage: %s minutes\n", argv[0]); exit(1);
21457       }
21458       ...
```

21459 **APPLICATION USAGE**

21460       The *atoi()* function is subsumed by *strtol()* but is retained because it is used extensively in  
21461       existing code. If the number is not known to be in range, *strtol()* should be used because *atoi()* is  
21462       not required to perform any error checking.

21463 **RATIONALE**

21464       None.

21465 **FUTURE DIRECTIONS**

21466       None.

21467 **SEE ALSO**21468       *strtol()*

21469       XBD &lt;stdlib.h&gt;



21470	<b>CHANGE HISTORY</b>
21471	First released in Issue 1. Derived from Issue 1 of the SVID.

21472 **NAME**21473        `atol`, `atoll` — convert a string to a long integer21474 **SYNOPSIS**21475        `#include <stdlib.h>`21476        `long atol(const char *nptr);` |21477        `long long atoll(const char *nptr);`21478 **DESCRIPTION**

21479 CX        The functionality described on this reference page is aligned with the ISO C standard. Any  
 21480        conflict between the requirements described here and the ISO C standard is unintentional. This  
 21481        volume of POSIX.1-2008 defers to the ISO C standard.

21482        Except as noted below, the call `atol(nptr)` shall be equivalent to: |21483        `strtol(nptr, (char **)NULL, 10)` |21484        Except as noted below, the call to `atoll(nptr)` shall be equivalent to: |21485        `strtoll(nptr, (char **)NULL, 10)`

21486        The handling of errors may differ. If the value cannot be represented, the behavior is undefined. |

21487 **RETURN VALUE**

21488        These functions shall return the converted value if the value can be represented.

21489 **ERRORS**

21490        No errors are defined.

21491 **EXAMPLES**

21492        None.

21493 **APPLICATION USAGE**

21494        If the number is not known to be in range, `strtol()` or `strtoll()` should be used because `atol()` and |  
 21495        `atoll()` are not required to perform any error checking.

21496 **RATIONALE**

21497        None.

21498 **FUTURE DIRECTIONS**

21499        None.

21500 **SEE ALSO**21501        [\*strtol\(\)\*](#)21502        XBD [\*<stdlib.h>\*](#)21503 **CHANGE HISTORY**

21504        First released in Issue 1. Derived from Issue 1 of the SVID.

21505 **Issue 6**21506        The `atoll()` function is added for alignment with the ISO/IEC 9899:1999 standard.21507 **Issue 7**21508        SD5-XSH-ERN-61 is applied, correcting the DESCRIPTION of `atoll()`.

21509 **NAME**21510        **basename** — return the last component of a pathname21511 **SYNOPSIS**

```
21512 XSI      #include <libgen.h>
21513          char *basename(char *path);
```

21514 **DESCRIPTION**

21515        The *basename()* function shall take the pathname pointed to by *path* and return a pointer to the  
 21516        final component of the pathname, deleting any trailing '/' characters.

21517        If the string pointed to by *path* consists entirely of the '/' character, *basename()* shall return a  
 21518        pointer to the string "/". If the string pointed to by *path* is exactly "/", it is implementation-  
 21519        defined whether '/' or "/" is returned.

21520        If *path* is a null pointer or points to an empty string, *basename()* shall return a pointer to the  
 21521        string ".".

21522        The *basename()* function may modify the string pointed to by *path*, and may return a pointer to |  
 21523        internal storage. The returned pointer might be invalidated or the storage might be overwritten |  
 21524        by a subsequent call to *basename()*. The returned pointer might also be invalidated if the calling +  
 21525        thread is terminated.

21526        The *basename()* function need not be thread-safe.

21527 **RETURN VALUE**

21528        The *basename()* function shall return a pointer to the final component of *path*.

21529 **ERRORS**

21530        No errors are defined.

21531 **EXAMPLES**21532        **Using basename()**

21533        The following program fragment returns a pointer to the value *lib*, which is the base name of  
 21534        */usr/lib*.

```
21535      #include <libgen.h>
21536      ...
21537      char name[] = "/usr/lib";
21538      char *base;

21539      base = basename(name);
21540      ...
```

21541        **Sample Input and Output Strings for the basename() and dirname() Functions and the**  
 21542        **basename and dirname Utilities**

basename() and dirname() Functions path Argument	String Returned by basename()	String Returned by dirname()	basename and dirname Utilities string Operand	Output Written by basename Utility	Output Written by dirname Utility
"usr"	"usr"	". "	usr	usr	.
"usr/ "	"usr"	". "	usr/	usr	.
" "	". "	". "	" "	. or empty string	.
" / "	" / "	" / "	/	/	/
" // "	" / " or " // "	" / " or " // "	//	/ or //	/ or //
" /// "	" / "	" / "	///	/	/
" /usr/ "	"usr"	" / "	/usr/	usr	/
" /usr/lib"	"lib"	" /usr"	/usr/lib	lib	/usr
" //usr//lib// "	"lib"	" //usr"	//usr//lib//	lib	//usr
" /home//dwc// test"	"test"	" /home//dwc"	/home//dwc// test	test	/home//dwc

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO***dirname()*

XBD &lt;libgen.h&gt;

XCU *basename***CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

Normative text previously in the APPLICATION USAGE section is moved to the DESCRIPTION.

A note indicating that this function need not be reentrant is added to the DESCRIPTION.

**Issue 6**

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/20 is applied, changing the DESCRIPTION to make it clear that the string referenced is the string pointed to by *path*.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #156 is applied.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0041 [75] is applied.

21582 **NAME**

21583 bind — bind a name to a socket

21584 **SYNOPSIS**

21585 #include &lt;sys/socket.h&gt;

```
21586 int bind(int socket, const struct sockaddr *address,
21587         socklen_t address_len);
```

21588 **DESCRIPTION**

21589 The *bind()* function shall assign a local socket address *address* to a socket identified by descriptor  
 21590 *socket* that has no local socket address assigned. Sockets created with the *socket()* function are  
 21591 initially unnamed; they are identified only by their address family.

21592 The *bind()* function takes the following arguments:

21593 *socket* Specifies the file descriptor of the socket to be bound.

21594 *address* Points to a **sockaddr** structure containing the address to be bound to the  
 21595 socket. The length and format of the address depend on the address family of  
 21596 the socket.

21597 *address\_len* Specifies the length of the **sockaddr** structure pointed to by the *address*  
 21598 argument.

21599 The socket specified by *socket* may require the process to have appropriate privileges to use the  
 21600 *bind()* function.

21601 If the address family of the socket is AF\_UNIX and the pathname in *address* names a symbolic  
 21602 link, *bind()* shall fail and set *errno* to [EADDRINUSE].

21603 If the socket address cannot be assigned immediately and O\_NONBLOCK is set for the file  
 21604 descriptor for the socket, *bind()* shall fail and set *errno* to [EINPROGRESS], but the assignment  
 21605 request shall not be aborted, and the assignment shall be completed asynchronously. Subsequent  
 21606 calls to *bind()* for the same socket, before the assignment is completed, shall fail and set *errno* to  
 21607 [EALREADY].

21608 When the assignment has been performed asynchronously, *pselect()*, *select()*, and *poll()* shall  
 21609 indicate that the file descriptor for the socket is ready for reading and writing.

21610 **RETURN VALUE**

21611 Upon successful completion, *bind()* shall return 0; otherwise, -1 shall be returned and *errno* set  
 21612 to indicate the error.

21613 **ERRORS**

21614 The *bind()* function shall fail if:

21615 [EADDRINUSE] The specified address is already in use.

21616 [EADDRNOTAVAIL]

21617 The specified address is not available from the local machine.

21618 [EAFNOSUPPORT]

21619 The specified address is not a valid address for the address family of the  
 21620 specified socket.

21621 [EALREADY] An assignment request is already in progress for the specified socket.

21622 [EBADF] The *socket* argument is not a valid file descriptor.

21623	[EINPROGRESS]	O_NONBLOCK is set for the file descriptor for the socket and the assignment cannot be immediately performed; the assignment shall be performed asynchronously.
21624		
21625		
21626	[EINVAL]	The socket is already bound to an address, and the protocol does not support binding to a new address; or the socket has been shut down.
21627		
21628	[ENOBUFS]	Insufficient resources were available to complete the call.
21629	[ENOTSOCK]	The <i>socket</i> argument does not refer to a socket.
21630	[EOPNOTSUPP]	The socket type of the specified socket does not support binding to an address.
21631	If the address family of the socket is AF_UNIX, then <i>bind()</i> shall fail if:	
21632	[EACCES]	A component of the path prefix denies search permission, or the requested name requires writing in a directory with a mode that denies write permission.
21633		
21634		
21635	[EDESTADDRREQ] or [EISDIR]	
21636		The <i>address</i> argument is a null pointer.
21637	[EIO]	An I/O error occurred.
21638	[ELOOP]	A loop exists in symbolic links encountered during resolution of the pathname in <i>address</i> .
21639		
21640	[ENAMETOOLONG]	
21641		The length of a component of a pathname is longer than {NAME_MAX}.
21642	[ENOENT]	A component of the path prefix of the pathname in <i>address</i> does not name an existing file or the pathname is an empty string.
21643		
21644	[ENOENT] or [ENOTDIR]	
21645		The pathname in <i>address</i> contains at least one non- <i>&lt;slash&gt;</i> character and ends with one or more trailing <i>&lt;slash&gt;</i> characters. If the pathname without the trailing <i>&lt;slash&gt;</i> characters would name an existing file, an [ENOENT] error shall not occur.
21646		
21647		
21648		
21649	[ENOTDIR]	A component of the path prefix of the pathname in <i>address</i> names an existing file that is neither a directory nor a symbolic link to a directory, or the pathname in <i>address</i> contains at least one non- <i>&lt;slash&gt;</i> character and ends with one or more trailing <i>&lt;slash&gt;</i> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.
21650		
21651		
21652		
21653		
21654		
21655	[EROFS]	The name would reside on a read-only file system.
21656	The <i>bind()</i> function may fail if:	
21657	[EACCES]	The specified address is protected and the current user does not have permission to bind to it.
21658		
21659	[EINVAL]	The <i>address_len</i> argument is not a valid length for the address family.
21660	[EISCONN]	The socket is already connected.
21661	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the pathname in <i>address</i> .
21662		

21663 [ENAMETOOLONG]  
 21664 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
 21665 symbolic link produced an intermediate result with a length that exceeds  
 21666 {PATH\_MAX}.

## 21667 EXAMPLES

21668 The following code segment shows how to create a socket and bind it to a name in the AF\_UNIX  
 21669 domain.

```
21670 #define MY_SOCKET_PATH "/somepath"
21671 int sfd;
21672 struct sockaddr_un my_addr;
21673 sfd = socket(AF_UNIX, SOCK_STREAM, 0);
21674 if (sfd == -1)
21675     /* Handle error */;
21676 memset(&my_addr, '\0', sizeof(struct sockaddr_un));
21677     /* Clear structure */
21678 my_addr.sun_family = AF_UNIX;
21679 strncpy(my_addr.sun_path, MY_SOCKET_PATH, sizeof(my_addr.sun_path) - 1);
21680 if (bind(sfd, (struct sockaddr *) &my_addr,
21681         sizeof(struct sockaddr_un)) == -1)
21682     /* Handle error */;
```

## 21683 APPLICATION USAGE

21684 An application program can retrieve the assigned socket name with the *getsockname()* function.

## 21685 RATIONALE

21686 None.

## 21687 FUTURE DIRECTIONS

21688 None.

## 21689 SEE ALSO

21690 *connect()*, *getsockname()*, *listen()*, *socket()*

21691 XBD <sys/socket.h>

## 21692 CHANGE HISTORY

21693 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

### 21694 Issue 7

21695 Austin Group Interpretation 1003.1-2001 #044 is applied, changing the “may fail” [ENOBUFFS]  
 21696 error to become a “shall fail” error.

21697 Austin Group Interpretation 1003.1-2001 #143 is applied.

21698 SD5-XSH-ERN-185 is applied.

21699 An example is added.

21700 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0042 [146], XSH/TC1-2008/0043 [146],  
 21701 and XSH/TC1-2008/0044 [324] are applied.

## NAME

bsearch — binary search a sorted table

## SYNOPSIS

```
#include <stdlib.h>
```

```
void *bsearch(const void *key, const void *base, size_t nel,
             size_t width, int (*compar)(const void *, const void *));
```

## DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The `bsearch()` function shall search an array of *nel* objects, the initial element of which is pointed to by *base*, for an element that matches the object pointed to by *key*. The size of each element in the array is specified by *width*. If the *nel* argument has the value zero, the comparison function pointed to by *compar* shall not be called and no match shall be found.

The comparison function pointed to by *compar* shall be called with two arguments that point to the *key* object and to an array element, in that order.

The application shall ensure that the comparison function pointed to by *compar* does not alter the contents of the array. The implementation may reorder elements of the array between calls to the comparison function, but shall not alter the contents of any individual element.

The implementation shall ensure that the first argument is always a pointer to the key.

When the same objects (consisting of width bytes, irrespective of their current positions in the array) are passed more than once to the comparison function, the results shall be consistent with one another. That is, the same object shall always compare the same way with the key.

The application shall ensure that the function returns an integer less than, equal to, or greater than 0 if the *key* object is considered, respectively, to be less than, to match, or to be greater than the array element. The application shall ensure that the array consists of all the elements that compare less than, all the elements that compare equal to, and all the elements that compare greater than the *key* object, in that order.

## RETURN VALUE

The *bsearch()* function shall return a pointer to a matching member of the array, or a null pointer if no match is found. If two or more members compare equal, which member is returned is unspecified.

## ERRORS

No errors are defined.

## EXAMPLES

The example below searches a table containing pointers to nodes consisting of a string and its length. The table is ordered alphabetically on the string in the node pointed to by each entry.

The code fragment below reads in strings and either finds the corresponding node and prints out the string and its length, or prints an error message.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
```

```
#define TABSIZE 1000
```

[illegible]



```

21746     char *string;
21747     int length;
21748 };
21749 struct node table[TABSIZE];    /* Table to be searched. */
21750     .
21751     .
21752     .
21753 {
21754     struct node *node_ptr, node;
21755     /* Routine to compare 2 nodes. */
21756     int node_compare(const void *, const void *);
21757     .
21758     .
21759     .
21760     while (scanf("%ms", &node.string) != EOF) {
21761         node_ptr = (struct node *)bsearch((void *)&node,
21762             (void *)table, TABSIZE,
21763             sizeof(struct node), node_compare);
21764         if (node_ptr != NULL) {
21765             (void)printf("string = %20s, length = %d\n",
21766                 node_ptr->string, node_ptr->length);
21767         } else {
21768             (void)printf("not found: %s\n", node.string);
21769         }
21770         free(node.string);
21771     }
21772 }
21773 /*
21774     This routine compares two nodes based on an
21775     alphabetical ordering of the string field.
21776 */
21777 int
21778 node_compare(const void *node1, const void *node2)
21779 {
21780     return strcoll(((const struct node *)node1)->string,
21781         ((const struct node *)node2)->string);
21782 }

```

## APPLICATION USAGE

The pointers to the key and the element at the base of the table should be of type pointer-to-element.

The comparison function need not compare every byte, so arbitrary data may be contained in the elements in addition to the values being compared.

In practice, the array is usually sorted according to the comparison function.

## RATIONALE

The requirement that the second argument (hereafter referred to as *p*) to the comparison function is a pointer to an element of the array implies that for every call all of the following expressions are non-zero:

```

21793 ( (char *)p - (char *)base ) % width == 0
21794 (char *)p >= (char *)base

```

21795           (char \*)p < (char \*)base + nel \* width

21796 **FUTURE DIRECTIONS**

21797           None.

21798 **SEE ALSO**

21799           *hcreate()*, *lsearch()*, *qsort()*, *tdelete()*

21800           XBD <stdlib.h>

21801 **CHANGE HISTORY**

21802           First released in Issue 1. Derived from Issue 1 of the SVID.

21803 **Issue 6**

21804           The normative text is updated to avoid use of the term “must” for application requirements.

21805           IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/11 is applied, adding to the  
21806           DESCRIPTION the last sentence of the first non-shaded paragraph, and the following three  
21807           paragraphs. The RATIONALE section is also updated. These changes are for alignment with the  
21808           ISO C standard.

21809 **Issue 7**

21810           The EXAMPLES section is revised.

21811 **NAME**

21812       btowc — single byte to wide character conversion

21813 **SYNOPSIS**

21814       #include &lt;stdio.h&gt;

21815       #include &lt;wchar.h&gt;

21816       wint\_t btowc(int c);

21817 **DESCRIPTION**

21818 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 21819       conflict between the requirements described here and the ISO C standard is unintentional. This  
 21820       volume of POSIX.1-2008 defers to the ISO C standard.

21821       The *btowc()* function shall determine whether *c* constitutes a valid (one-byte) character in the  
 21822       initial shift state.

21823       The behavior of this function shall be affected by the *LC\_CTYPE* category of the current locale.

21824 **RETURN VALUE**

21825       The *btowc()* function shall return WEOF if *c* has the value EOF or if (**unsigned char**) *c* does not  
 21826       constitute a valid (one-byte) character in the initial shift state. Otherwise, it shall return the  
 21827       wide-character representation of that character.

21828 CX       In the POSIX locale, *btowc()* shall not return WEOF if *c* has a value in the range 0 to 255 +  
 21829       inclusive.

21830 **ERRORS**

21831       No errors are defined.

21832 **EXAMPLES**

21833       None.

21834 **APPLICATION USAGE**

21835       None.

21836 **RATIONALE**

21837       None.

21838 **FUTURE DIRECTIONS**

21839       None.

21840 **SEE ALSO**21841       *wctob()*

21842       XBD &lt;stdio.h&gt;, &lt;wchar.h&gt;

21843 **CHANGE HISTORY**

21844       First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
 21845       (E).

21846 **NAME**

21847 cabs, cabsf, cabsl — return a complex absolute value

21848 **SYNOPSIS**

```
21849 #include <complex.h>
21850 double cabs(double complex z);
21851 float cabsf(float complex z);
21852 long double cabsl(long double complex z);
```

21853 **DESCRIPTION**

21854 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
21855 conflict between the requirements described here and the ISO C standard is unintentional. This  
21856 volume of POSIX.1-2008 defers to the ISO C standard.

21857 These functions shall compute the complex absolute value (also called norm, modulus, or  
21858 magnitude) of *z*.

21859 **RETURN VALUE**

21860 These functions shall return the complex absolute value.

21861 **ERRORS**

21862 No errors are defined.

21863 **EXAMPLES**

21864 None.

21865 **APPLICATION USAGE**

21866 None.

21867 **RATIONALE**

21868 None.

21869 **FUTURE DIRECTIONS**

21870 None.

21871 **SEE ALSO**21872 XBD [<complex.h>](#)21873 **CHANGE HISTORY**

21874 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

21875 **NAME**

21876 cacos, cacosf, cacosl — complex arc cosine functions

21877 **SYNOPSIS**

```
21878 #include <complex.h>
21879 double complex cacos(double complex z);
21880 float complex cacosf(float complex z);
21881 long double complex cacosl(long double complex z);
```

21882 **DESCRIPTION**

21883 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 21884 conflict between the requirements described here and the ISO C standard is unintentional. This  
 21885 volume of POSIX.1-2008 defers to the ISO C standard.

21886 These functions shall compute the complex arc cosine of  $z$ , with branch cuts outside the interval  
 21887  $[-1, +1]$  along the real axis.

21888 **RETURN VALUE**

21889 These functions shall return the complex arc cosine value, in the range of a strip mathematically  
 21890 unbounded along the imaginary axis and in the interval  $[0, \pi]$  along the real axis.

21891 **ERRORS**

21892 No errors are defined.

21893 **EXAMPLES**

21894 None.

21895 **APPLICATION USAGE**

21896 None.

21897 **RATIONALE**

21898 None.

21899 **FUTURE DIRECTIONS**

21900 None.

21901 **SEE ALSO**

21902 [ccos\(\)](#)  
 21903 XBD [<complex.h>](#)

21904 **CHANGE HISTORY**

21905 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

21906 **NAME**

21907 cacosh, cacoshf, cacoshl — complex arc hyperbolic cosine functions

21908 **SYNOPSIS**21909 `#include <complex.h>`21910 `double complex cacosh(double complex z);`21911 `float complex cacoshf(float complex z);`21912 `long double complex cacoshl(long double complex z);`21913 **DESCRIPTION**

21914 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
21915 conflict between the requirements described here and the ISO C standard is unintentional. This  
21916 volume of POSIX.1-2008 defers to the ISO C standard.

21917 These functions shall compute the complex arc hyperbolic cosine of  $z$ , with a branch cut at  
21918 values less than 1 along the real axis.

21919 **RETURN VALUE**

21920 These functions shall return the complex arc hyperbolic cosine value, in the range of a half-strip  
21921 of non-negative values along the real axis and in the interval  $[-i\pi, +i\pi]$  along the imaginary axis.

21922 **ERRORS**

21923 No errors are defined.

21924 **EXAMPLES**

21925 None.

21926 **APPLICATION USAGE**

21927 None.

21928 **RATIONALE**

21929 None.

21930 **FUTURE DIRECTIONS**

21931 None.

21932 **SEE ALSO**21933 [ccosh\(\)](#)21934 XBD [<complex.h>](#)21935 **CHANGE HISTORY**

21936 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

21937 **NAME**

21938       cacosl — complex arc cosine functions

21939 **SYNOPSIS**

21940       #include &lt;complex.h&gt;

21941       long double complex cacosl(long double complex z);

21942 **DESCRIPTION**21943       Refer to *acos()*.

21944 **NAME**

21945        calloc — a memory allocator

21946 **SYNOPSIS**

21947        #include &lt;stdlib.h&gt;

21948        void \*calloc(size\_t *nelem*, size\_t *elsize*);21949 **DESCRIPTION**

21950 CX        The functionality described on this reference page is aligned with the ISO C standard. Any  
 21951        conflict between the requirements described here and the ISO C standard is unintentional. This  
 21952        volume of POSIX.1-2008 defers to the ISO C standard.

21953        The *calloc()* function shall allocate unused space for an array of *nelem* elements each of whose  
 21954        size in bytes is *elsize*. The space shall be initialized to all bits 0.

21955        The order and contiguity of storage allocated by successive calls to *calloc()* is unspecified. The  
 21956        pointer returned if the allocation succeeds shall be suitably aligned so that it may be assigned to  
 21957        a pointer to any type of object and then used to access such an object or an array of such objects  
 21958        in the space allocated (until the space is explicitly freed or reallocated). Each such allocation shall  
 21959        yield a pointer to an object disjoint from any other object. The pointer returned shall point to the  
 21960        start (lowest byte address) of the allocated space. If the space cannot be allocated, a null pointer  
 21961        shall be returned. If the size of the space requested is 0, the behavior is implementation-defined:  
 21962        either a null pointer shall be returned, or the behavior shall be as if the size were some non-zero  
 21963        value, except that the behavior is undefined if the returned pointer is used to access an object.

21964 **RETURN VALUE**

21965        Upon successful completion with both *nelem* and *elsize* non-zero, *calloc()* shall return a pointer to  
 21966        the allocated space. If either *nelem* or *elsize* is 0, then either:

- 21967 CX        • A null pointer shall be returned and *errno* may be set to an implementation-defined value,  
 21968                or
- 21969        • A pointer to the allocated space shall be returned. The application shall ensure that the  
 21970        pointer is not used to access an object.

21971 CX        Otherwise, it shall return a null pointer and set *errno* to indicate the error.

21972 **ERRORS**21973        The *calloc()* function shall fail if:

21974 CX        [ENOMEM]        Insufficient memory is available.

21975 **EXAMPLES**

21976        None.

21977 **APPLICATION USAGE**

21978        There is now no requirement for the implementation to support the inclusion of &lt;malloc.h&gt;.

21979 **RATIONALE**

21980        None.

21981 **FUTURE DIRECTIONS**

21982        None.

21983 **SEE ALSO**21984        *free()*, *malloc()*, *realloc()*

21985        XBD &lt;stdlib.h&gt;



21986 **CHANGE HISTORY**

21987 First released in Issue 1. Derived from Issue 1 of the SVID.

21988 **Issue 6**

21989 Extensions beyond the ISO C standard are marked.

21990 The following new requirements on POSIX implementations derive from alignment with the  
21991 Single UNIX Specification:

- 21992
- The setting of *errno* and the [ENOMEM] error condition are mandatory if an insufficient
- 
- 21993 memory condition occurs.

21994 **NAME**

21995       carg, cargf, cargl — complex argument functions

21996 **SYNOPSIS**

21997       #include &lt;complex.h&gt;

21998       double carg(double complex z);

21999       float cargf(float complex z);

22000       long double cargl(long double complex z);

22001 **DESCRIPTION**

22002 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
22003       conflict between the requirements described here and the ISO C standard is unintentional. This  
22004       volume of POSIX.1-2008 defers to the ISO C standard.

22005       These functions shall compute the argument (also called phase angle) of  $z$ , with a branch cut  
22006       along the negative real axis.

22007 **RETURN VALUE**22008       These functions shall return the value of the argument in the interval  $[-\pi, +\pi]$ .22009 **ERRORS**

22010       No errors are defined.

22011 **EXAMPLES**

22012       None.

22013 **APPLICATION USAGE**

22014       None.

22015 **RATIONALE**

22016       None.

22017 **FUTURE DIRECTIONS**

22018       None.

22019 **SEE ALSO**22020        *cimag()*, *conj()*, *cproj()*22021       XBD **<complex.h>**22022 **CHANGE HISTORY**

22023       First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

22024 **NAME**

22025        casin, casinf, casinl — complex arc sine functions

22026 **SYNOPSIS**

```
22027        #include <complex.h>
22028        double complex casin(double complex z);
22029        float complex casinf(float complex z);
22030        long double complex casinl(long double complex z);
```

22031 **DESCRIPTION**

22032 CX        The functionality described on this reference page is aligned with the ISO C standard. Any  
 22033            conflict between the requirements described here and the ISO C standard is unintentional. This  
 22034            volume of POSIX.1-2008 defers to the ISO C standard.

22035        These functions shall compute the complex arc sine of  $z$ , with branch cuts outside the interval  
 22036         $[-1, +1]$  along the real axis.

22037 **RETURN VALUE**

22038        These functions shall return the complex arc sine value, in the range of a strip mathematically  
 22039        unbounded along the imaginary axis and in the interval  $[-\pi/2, +\pi/2]$  along the real axis.

22040 **ERRORS**

22041        No errors are defined.

22042 **EXAMPLES**

22043        None.

22044 **APPLICATION USAGE**

22045        None.

22046 **RATIONALE**

22047        None.

22048 **FUTURE DIRECTIONS**

22049        None.

22050 **SEE ALSO**22051        [csin\(\)](#)22052        XBD [<complex.h>](#)22053 **CHANGE HISTORY**

22054        First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

22055 **NAME**

22056       c sinh, c sinh f, c sinh l — complex arc hyperbolic sine functions

22057 **SYNOPSIS**

22058       #include &lt;complex.h&gt;

22059       double complex c sinh(double complex z);

22060       float complex c sinh f(float complex z);

22061       long double complex c sinh l(long double complex z);

22062 **DESCRIPTION**

22063 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
22064       conflict between the requirements described here and the ISO C standard is unintentional. This  
22065       volume of POSIX.1-2008 defers to the ISO C standard.

22066       These functions shall compute the complex arc hyperbolic sine of  $z$ , with branch cuts outside the  
22067       interval  $[-i, +i]$  along the imaginary axis.

22068 **RETURN VALUE**

22069       These functions shall return the complex arc hyperbolic sine value, in the range of a strip  
22070       mathematically unbounded along the real axis and in the interval  $[-i\pi/2, +i\pi/2]$  along the  
22071       imaginary axis.

22072 **ERRORS**

22073       No errors are defined.

22074 **EXAMPLES**

22075       None.

22076 **APPLICATION USAGE**

22077       None.

22078 **RATIONALE**

22079       None.

22080 **FUTURE DIRECTIONS**

22081       None.

22082 **SEE ALSO**

22083       c sinh()

22084       XBD &lt;complex.h&gt;

22085 **CHANGE HISTORY**

22086       First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

22087 **NAME**

22088        casinl — complex arc sine functions

22089 **SYNOPSIS**22090        `#include <complex.h>`22091        `long double complex casinl(long double complex z);`22092 **DESCRIPTION**22093        Refer to *casin()*.

22094 **NAME**

22095       catan, catanf, catanl — complex arc tangent functions

22096 **SYNOPSIS**

```
22097       #include <complex.h>
22098       double complex catan(double complex z);
22099       float complex catanf(float complex z);
22100       long double complex catanl(long double complex z);
```

22101 **DESCRIPTION**

22102 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
22103       conflict between the requirements described here and the ISO C standard is unintentional. This  
22104       volume of POSIX.1-2008 defers to the ISO C standard.

22105       These functions shall compute the complex arc tangent of  $z$ , with branch cuts outside the  
22106       interval  $[-i, +i]$  along the imaginary axis.

22107 **RETURN VALUE**

22108       These functions shall return the complex arc tangent value, in the range of a strip  
22109       mathematically unbounded along the imaginary axis and in the interval  $[-\pi/2, +\pi/2]$  along the  
22110       real axis.

22111 **ERRORS**

22112       No errors are defined.

22113 **EXAMPLES**

22114       None.

22115 **APPLICATION USAGE**

22116       None.

22117 **RATIONALE**

22118       None.

22119 **FUTURE DIRECTIONS**

22120       None.

22121 **SEE ALSO**

22122       ctan()

22123       XBD &lt;complex.h&gt;

22124 **CHANGE HISTORY**

22125       First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

22126 **NAME**

22127       catanh, catanhf, catanhl — complex arc hyperbolic tangent functions

22128 **SYNOPSIS**

22129       #include &lt;complex.h&gt;

22130       double complex catanh(double complex z);

22131       float complex catanhf(float complex z);

22132       long double complex catanhl(long double complex z);

22133 **DESCRIPTION**

22134 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 22135       conflict between the requirements described here and the ISO C standard is unintentional. This  
 22136       volume of POSIX.1-2008 defers to the ISO C standard.

22137       These functions shall compute the complex arc hyperbolic tangent of  $z$ , with branch cuts outside  
 22138       the interval  $[-1, +1]$  along the real axis.

22139 **RETURN VALUE**

22140       These functions shall return the complex arc hyperbolic tangent value, in the range of a strip  
 22141       mathematically unbounded along the real axis and in the interval  $[-i\pi/2, +i\pi/2]$  along the  
 22142       imaginary axis.

22143 **ERRORS**

22144       No errors are defined.

22145 **EXAMPLES**

22146       None.

22147 **APPLICATION USAGE**

22148       None.

22149 **RATIONALE**

22150       None.

22151 **FUTURE DIRECTIONS**

22152       None.

22153 **SEE ALSO**

22154       ctanh()

22155       XBD &lt;complex.h&gt;

22156 **CHANGE HISTORY**

22157       First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

22158 **NAME**

22159       catanl — complex arc tangent functions

22160 **SYNOPSIS**

22161       #include &lt;complex.h&gt;

22162       long double complex catanl(long double complex z);

22163 **DESCRIPTION**22164       Refer to *catan()*.



22165 **NAME**

22166 catclose — close a message catalog descriptor

22167 **SYNOPSIS**

22168 #include &lt;nl\_types.h&gt;

22169 int catclose(nl\_catd catd);

22170 **DESCRIPTION**

22171 The *catclose()* function shall close the message catalog identified by *catd*. If a file descriptor is  
22172 used to implement the type **nl\_catd**, that file descriptor shall be closed.

22173 **RETURN VALUE**

22174 Upon successful completion, *catclose()* shall return 0; otherwise, -1 shall be returned, and *errno*  
22175 set to indicate the error.

22176 **ERRORS**22177 The *catclose()* function may fail if:

22178 [EBADF] The catalog descriptor is not valid.

22179 [EINTR] The *catclose()* function was interrupted by a signal.22180 **EXAMPLES**

22181 None.

22182 **APPLICATION USAGE**

22183 None.

22184 **RATIONALE**

22185 None.

22186 **FUTURE DIRECTIONS**

22187 None.

22188 **SEE ALSO**22189 *catgets()*, *catopen()*

22190 XBD &lt;nl\_types.h&gt;

22191 **CHANGE HISTORY**

22192 First released in Issue 2.

22193 **Issue 7**22194 The *catclose()* function is moved from the XSI option to the Base.

22195 **NAME**

22196 catgets — read a program message

22197 **SYNOPSIS**

22198 #include &lt;nl\_types.h&gt;

22199 char \*catgets(nl\_catd catd, int set\_id, int msg\_id, const char \*s);

22200 **DESCRIPTION**

22201 The *catgets()* function shall attempt to read message *msg\_id*, in set *set\_id*, from the message  
22202 catalog identified by *catd*. The *catd* argument is a message catalog descriptor returned from an  
22203 earlier call to *catopen()*. The results are undefined if *catd* is not a value returned by *catopen()* for  
22204 a message catalog still open in the process. The *s* argument points to a default message string  
22205 which shall be returned by *catgets()* if it cannot retrieve the identified message.

22206 The *catgets()* function need not be thread-safe.22207 **RETURN VALUE**

22208 If the identified message is retrieved successfully, *catgets()* shall return a pointer to an internal  
22209 buffer area containing the null-terminated message string. If the call is unsuccessful for any  
22210 reason, *s* shall be returned and *errno* shall be set to indicate the error.

22211 **ERRORS**22212 The *catgets()* function shall fail if:

22213 [EINTR] The read operation was terminated due to the receipt of a signal, and no data  
22214 was transferred.

22215 [ENOMSG] The message identified by *set\_id* and *msg\_id* is not in the message catalog.

22216 The *catgets()* function may fail if:

22217 [EBADF] The *catd* argument is not a valid message catalog descriptor open for reading.

22218 [EBADMSG] The message identified by *set\_id* and *msg\_id* in the specified message catalog  
22219 did not satisfy implementation-defined security criteria.

22220 [EINVAL] The message catalog identified by *catd* is corrupted.

22221 **EXAMPLES**

22222 None.

22223 **APPLICATION USAGE**

22224 None.

22225 **RATIONALE**

22226 None.

22227 **FUTURE DIRECTIONS**

22228 None.

22229 **SEE ALSO**22230 *catclose()*, *catopen()*

22231 XBD &lt;nl\_types.h&gt;

22232 **CHANGE HISTORY**

22233 First released in Issue 2.

22234 **Issue 5**

22235 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

22236 **Issue 6**

22237 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

22238 **Issue 7**

22239 Austin Group Interpretation 1003.1-2001 #044 is applied, changing the “may fail” [EINTR] and  
22240 [ENOMSG] errors to become “shall fail” errors, updating the RETURN VALUE section, and  
22241 updating the DESCRIPTION to note that: “The results are undefined if *catd* is not a value  
22242 returned by *catopen()* for a message catalog still open in the process.

22243 Austin Group Interpretation 1003.1-2001 #148 is applied, adding

22244 The *catgets()* function is moved from the XSI option to the Base.

22245 **NAME**

22246 catopen — open a message catalog

22247 **SYNOPSIS**

22248 #include &lt;nl\_types.h&gt;

22249 nl\_catd catopen(const char \*name, int oflag);

22250 **DESCRIPTION**

22251 The *catopen()* function shall open a message catalog and return a message catalog descriptor.  
 22252 The *name* argument specifies the name of the message catalog to be opened. If *name* contains a  
 22253 `'/'`, then *name* specifies a pathname for the message catalog. Otherwise, the environment  
 22254 variable *NLSPATH* is used with *name* substituted for the `%N` conversion specification (see XBD  
 22255 Chapter 8, on page 173); if *NLSPATH* exists in the environment when the process starts, then if  
 22256 the process has appropriate privileges, the behavior of *catopen()* is undefined. If *NLSPATH* does  
 22257 not exist in the environment, or if a message catalog cannot be found in any of the components  
 22258 specified by *NLSPATH*, then an implementation-defined default path shall be used. This default  
 22259 may be affected by the setting of *LC\_MESSAGES* if the value of *oflag* is *NL\_CAT\_LOCALE*, or  
 22260 the *LANG* environment variable if *oflag* is 0.

22261 A message catalog descriptor shall remain valid in a process until that process closes it, or a  
 22262 successful call to one of the *exec* functions. A change in the setting of the *LC\_MESSAGES*  
 22263 category may invalidate existing open catalogs.

22264 If a file descriptor is used to implement message catalog descriptors, the *FD\_CLOEXEC* flag  
 22265 shall be set; see <fcntl.h>.

22266 If the value of the *oflag* argument is 0, the *LANG* environment variable is used to locate the  
 22267 catalog without regard to the *LC\_MESSAGES* category. If the *oflag* argument is  
 22268 *NL\_CAT\_LOCALE*, the *LC\_MESSAGES* category is used to locate the message catalog (see XBD  
 22269 Section 8.2, on page 174).

22270 **RETURN VALUE**

22271 Upon successful completion, *catopen()* shall return a message catalog descriptor for use on  
 22272 subsequent calls to *catgets()* and *catclose()*. Otherwise, *catopen()* shall return (*nl\_catd*) -1 and set  
 22273 *errno* to indicate the error.

22274 **ERRORS**22275 The *catopen()* function may fail if:

22276 [EACCES] Search permission is denied for the component of the path prefix of the  
 22277 message catalog or read permission is denied for the message catalog.

22278 [EMFILE] All file descriptors available to the process are currently open.

22279 [ENAMETOOLONG]  
 22280 The length of a component of a pathname is longer than {NAME\_MAX}.

22281 [ENAMETOOLONG]  
 22282 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
 22283 symbolic link produced an intermediate result with a length that exceeds  
 22284 {PATH\_MAX}.

22285 [ENFILE] Too many files are currently open in the system.

22286 [ENOENT] The message catalog does not exist or the *name* argument points to an empty  
 22287 string.

22288 [ENOMEM] Insufficient storage space is available.

22289 [ENOTDIR] A component of the path prefix of the message catalog names an existing file  
 22290 that is neither a directory nor a symbolic link to a directory, or the pathname  
 22291 of the message catalog contains at least one non-`<slash>` character and ends  
 22292 with one or more trailing `<slash>` characters and the last pathname  
 22293 component names an existing file that is neither a directory nor a symbolic  
 22294 link to a directory.

**EXAMPLES**

22295 None.  
 22296

**APPLICATION USAGE**

22298 Some implementations of `catopen()` use `malloc()` to allocate space for internal buffer areas. The  
 22299 `catopen()` function may fail if there is insufficient storage space available to accommodate these  
 22300 buffers.

22301 Conforming applications must assume that message catalog descriptors are not valid after a call  
 22302 to one of the `exec` functions.

22303 Application developers should be aware that guidelines for the location of message catalogs  
 22304 have not yet been developed. Therefore they should take care to avoid conflicting with catalogs  
 22305 used by other applications and the standard utilities.

22306 To be sure that messages produced by an application running with appropriate privileges cannot +  
 22307 be used by an attacker setting an unexpected value for `NLSPATH` in the environment to confuse +  
 22308 a system administrator, such applications should use pathnames containing a `' / '` to get defined +  
 22309 behavior when using `catopen()` to open a message catalog.

**RATIONALE**

22310 None.  
 22311

**FUTURE DIRECTIONS**

22312 None.  
 22313

**SEE ALSO**

22314 `catclose()`, `catgets()`  
 22315

22316 XBD Chapter 8 (on page 173), `<fcntl.h>`, `<nl_types.h>`,

**CHANGE HISTORY**

22317 First released in Issue 2.  
 22318

**Issue 7**

22319 Austin Group Interpretation 1003.1-2001 #143 is applied.  
 22320

22321 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

22322 The `catopen()` function is moved from the XSI option to the Base.

22323 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0045 [324] is applied.

22324 **NAME**

22325       cbrt, cbrtf, cbrtl — cube root functions

22326 **SYNOPSIS**

22327       #include &lt;math.h&gt;

22328       double cbrt(double x);

22329       float cbrtf(float x);

22330       long double cbrtl(long double x);

22331 **DESCRIPTION**

22332 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
22333       conflict between the requirements described here and the ISO C standard is unintentional. This  
22334       volume of POSIX.1-2008 defers to the ISO C standard.

22335       These functions shall compute the real cube root of their argument *x*.22336 **RETURN VALUE**22337       Upon successful completion, these functions shall return the cube root of *x*.22338 MX       If *x* is NaN, a NaN shall be returned.22339       If *x* is  $\pm 0$  or  $\pm \text{Inf}$ , *x* shall be returned.22340 **ERRORS**

22341       No errors are defined.

22342 **EXAMPLES**

22343       None.

22344 **APPLICATION USAGE**

22345       None.

22346 **RATIONALE**

22347       For some applications, a true cube root function, which returns negative results for negative  
22348       arguments, is more appropriate than *pow(x, 1.0/3.0)*, which returns a NaN for *x* less than 0.

22349 **FUTURE DIRECTIONS**

22350       None.

22351 **SEE ALSO**22352       XBD [<math.h>](#)22353 **CHANGE HISTORY**

22354       First released in Issue 4, Version 2.

22355 **Issue 5**

22356       Moved from X/OPEN UNIX extension to BASE.

22357 **Issue 6**22358       The *cbrt()* function is no longer marked as an extension.22359       The *cbrtf()* and *cbrtl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

22360       The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
22361       revised to align with the ISO/IEC 9899:1999 standard.

22362       IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
22363       marked.

22364 **NAME**

22365 ccos, ccosf, ccosl — complex cosine functions

22366 **SYNOPSIS**

22367 #include &lt;complex.h&gt;

22368 double complex ccos(double complex z);

22369 float complex ccosf(float complex z);

22370 long double complex ccosl(long double complex z);

22371 **DESCRIPTION**

22372 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
22373 conflict between the requirements described here and the ISO C standard is unintentional. This  
22374 volume of POSIX.1-2008 defers to the ISO C standard.

22375 These functions shall compute the complex cosine of z.

22376 **RETURN VALUE**

22377 These functions shall return the complex cosine value.

22378 **ERRORS**

22379 No errors are defined.

22380 **EXAMPLES**

22381 None.

22382 **APPLICATION USAGE**

22383 None.

22384 **RATIONALE**

22385 None.

22386 **FUTURE DIRECTIONS**

22387 None.

22388 **SEE ALSO**22389 [ccos\(\)](#)22390 XBD [<complex.h>](#)22391 **CHANGE HISTORY**

22392 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

22393 **NAME**

22394 ccosh, ccoshf, ccoshl — complex hyperbolic cosine functions

22395 **SYNOPSIS**

22396 #include &lt;complex.h&gt;

22397 double complex ccosh(double complex z);

22398 float complex ccoshf(float complex z);

22399 long double complex ccoshl(long double complex z);

22400 **DESCRIPTION**

22401 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
22402 conflict between the requirements described here and the ISO C standard is unintentional. This  
22403 volume of POSIX.1-2008 defers to the ISO C standard.

22404 These functions shall compute the complex hyperbolic cosine of *z*.22405 **RETURN VALUE**

22406 These functions shall return the complex hyperbolic cosine value.

22407 **ERRORS**

22408 No errors are defined.

22409 **EXAMPLES**

22410 None.

22411 **APPLICATION USAGE**

22412 None.

22413 **RATIONALE**

22414 None.

22415 **FUTURE DIRECTIONS**

22416 None.

22417 **SEE ALSO**22418 *cacosh()*

22419 XBD &lt;complex.h&gt;

22420 **CHANGE HISTORY**

22421 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.



22422 **NAME**

22423       ccosl — complex cosine functions

22424 **SYNOPSIS**

22425       #include &lt;complex.h&gt;

22426       long double complex ccosl(long double complex z);

22427 **DESCRIPTION**22428       Refer to *ccos()*.

22429 **NAME**

22430       ceil, ceilf, ceill — ceiling value function

22431 **SYNOPSIS**

```
22432       #include <math.h>
22433       double ceil(double x);
22434       float ceilf(float x);
22435       long double ceill(long double x);
```

22436 **DESCRIPTION**

22437 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 22438       conflict between the requirements described here and the ISO C standard is unintentional. This  
 22439       volume of POSIX.1-2008 defers to the ISO C standard.

22440       These functions shall compute the smallest integral value not less than *x*.22441 **RETURN VALUE**22442 MX       The result shall have the same sign as *x*.

22443       Upon successful completion, *ceil()*, *ceilf()*, and *ceill()* shall return the smallest integral value not  
 22444       less than *x*, expressed as a type **double**, **float**, or **long double**, respectively.

22445 MX       If *x* is NaN, a NaN shall be returned.22446       If *x* is  $\pm 0$  or  $\pm \text{Inf}$ , *x* shall be returned.22447 **ERRORS**

22448       No errors are defined.

22449 **EXAMPLES**

22450       None.

22451 **APPLICATION USAGE**

22452       The integral value returned by these functions need not be expressible as an **intmax\_t**. The  
 22453       return value should be tested before assigning it to an integer type to avoid the undefined  
 22454       results of an integer overflow.

22455       These functions may raise the inexact floating-point exception if the result differs in value from  
 22456       the argument.

22457 **RATIONALE**

22458       None.

22459 **FUTURE DIRECTIONS**

22460       None.

22461 **SEE ALSO**22462       *feclearexcept()*, *fetestexcept()*, *floor()*, *isnan()*22463       XBD Section 4.20 (on page 117), **<math.h>**22464 **CHANGE HISTORY**

22465       First released in Issue 1. Derived from Issue 1 of the SVID.

22466 **Issue 5**

22467       The DESCRIPTION is updated to indicate how an application should check for an error. This  
 22468       text was previously published in the APPLICATION USAGE section.

**Issue 6**

The *ceilf()* and *ceil()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

**Issue 7**

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0046 [346] is applied.

22477 **NAME**

22478 cexp, cexpf, cexpl — complex exponential functions

22479 **SYNOPSIS**

22480 #include &lt;complex.h&gt;

22481 double complex cexp(double complex z);

22482 float complex cexpf(float complex z);

22483 long double complex cexpl(long double complex z);

22484 **DESCRIPTION**

22485 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
22486 conflict between the requirements described here and the ISO C standard is unintentional. This  
22487 volume of POSIX.1-2008 defers to the ISO C standard.

22488 These functions shall compute the complex exponent of  $z$ , defined as  $e^z$ .22489 **RETURN VALUE**22490 These functions shall return the complex exponential value of  $z$ .22491 **ERRORS**

22492 No errors are defined.

22493 **EXAMPLES**

22494 None.

22495 **APPLICATION USAGE**

22496 None.

22497 **RATIONALE**

22498 None.

22499 **FUTURE DIRECTIONS**

22500 None.

22501 **SEE ALSO**22502 [clog\(\)](#)22503 XBD [<complex.h>](#)22504 **CHANGE HISTORY**

22505 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

22506 **NAME**

22507 cfgetispeed — get input baud rate

22508 **SYNOPSIS**

22509 #include &lt;termios.h&gt;

22510 speed\_t cfgetispeed(const struct termios \*termios\_p);

22511 **DESCRIPTION**22512 The *cfgetispeed()* function shall extract the input baud rate from the **termios** structure to which  
22513 the *termios\_p* argument points.22514 This function shall return exactly the value in the **termios** data structure, without interpretation.22515 **RETURN VALUE**22516 Upon successful completion, *cfgetispeed()* shall return a value of type **speed\_t** representing the  
22517 input baud rate.22518 **ERRORS**

22519 No errors are defined.

22520 **EXAMPLES**

22521 None.

22522 **APPLICATION USAGE**

22523 None.

22524 **RATIONALE**22525 The term “baud” is used historically here, but is not technically correct. This is properly “bits per  
22526 second”, which may not be the same as baud. However, the term is used because of the  
22527 historical usage and understanding.22528 The *cfgetispeed()*, *cfsetispeed()*, *cfsetospeed()*, and *cfsetispeed()* functions do not take arguments as  
22529 numbers, but rather as symbolic names. There are two reasons for this:

- 22530 1. Historically, numbers were not used because of the way the rate was stored in the data
- 
- 22531 structure. This is retained even though a function is now used.
- 
- 22532 2. More importantly, only a limited set of possible rates is at all portable, and this constrains
- 
- 22533 the application to that set.

22534 There is nothing to prevent an implementation accepting as an extension a number (such as 126),  
22535 and since the encoding of the Bxxx symbols is not specified, this can be done to avoid  
22536 introducing ambiguity.22537 Setting the input baud rate to zero was a mechanism to allow for split baud rates. Clarifications  
22538 in this volume of POSIX.1-2008 have made it possible to determine whether split rates are  
22539 supported and to support them without having to treat zero as a special case. Since this  
22540 functionality is also confusing, it has been declared obsolescent. The 0 argument referred to is  
22541 the literal constant 0, not the symbolic constant B0. This volume of POSIX.1-2008 does not  
22542 preclude B0 from being defined as the value 0; in fact, implementations would likely benefit  
22543 from the two being equivalent. This volume of POSIX.1-2008 does not fully specify whether the  
22544 previous *cfsetispeed()* value is retained after a *tcgetattr()* as the actual value or as zero. Therefore,  
22545 conforming applications should always set both the input speed and output speed when setting  
22546 either.22547 In historical implementations, the baud rate information is traditionally kept in **c\_cflag**.  
22548 Applications should be written to presume that this might be the case (and thus not blindly copy  
22549 **c\_cflag**), but not to rely on it in case it is in some other field of the structure. Setting the **c\_cflag**  
22550 field absolutely after setting a baud rate is a non-portable action because of this. In general, the

22551 unused parts of the flag fields might be used by the implementation and should not be blindly  
22552 copied from the descriptions of one terminal device to another.

22553 **FUTURE DIRECTIONS**

22554 None.

22555 **SEE ALSO**

22556 [\*cfgetospeed\(\)\*](#), [\*cfsetispeed\(\)\*](#), [\*cfsetospeed\(\)\*](#), [\*tcgetattr\(\)\*](#)

22557 XBD [Chapter 11](#) (on page 199), [\*\*<termios.h>\*\*](#)

22558 **CHANGE HISTORY**

22559 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

22560 **NAME**

22561 cfgetospeed — get output baud rate

22562 **SYNOPSIS**

22563 #include &lt;termios.h&gt;

22564 speed\_t cfgetospeed(const struct termios \*termios\_p);

22565 **DESCRIPTION**22566 The *cfgetospeed()* function shall extract the output baud rate from the **termios** structure to which  
22567 the *termios\_p* argument points.22568 This function shall return exactly the value in the **termios** data structure, without interpretation.22569 **RETURN VALUE**22570 Upon successful completion, *cfgetospeed()* shall return a value of type **speed\_t** representing the  
22571 output baud rate.22572 **ERRORS**

22573 No errors are defined.

22574 **EXAMPLES**

22575 None.

22576 **APPLICATION USAGE**

22577 None.

22578 **RATIONALE**22579 Refer to *cfgetispeed()*.22580 **FUTURE DIRECTIONS**

22581 None.

22582 **SEE ALSO**22583 *cfgetispeed()*, *cfsetispeed()*, *cfsetospeed()*, *tcgetattr()*

22584 XBD Chapter 11 (on page 199), &lt;termios.h&gt;

22585 **CHANGE HISTORY**

22586 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

22587 **NAME**

22588 cfsetispeed — set input baud rate

22589 **SYNOPSIS**

22590 #include &lt;termios.h&gt;

22591 int cfsetispeed(struct termios \*termios\_p, speed\_t speed);

22592 **DESCRIPTION**22593 The *cfsetispeed()* function shall set the input baud rate stored in the structure pointed to by  
22594 *termios\_p* to *speed*.22595 There shall be no effect on the baud rates set in the hardware until a subsequent successful call  
22596 to *tcsetattr()* with the same **termios** structure. Similarly, errors resulting from attempts to set  
22597 baud rates not supported by the terminal device need not be detected until the *tcsetattr()*  
22598 function is called.22599 **RETURN VALUE**22600 Upon successful completion, *cfsetispeed()* shall return 0; otherwise, -1 shall be returned, and  
22601 *errno* may be set to indicate the error.22602 **ERRORS**22603 The *cfsetispeed()* function may fail if:22604 [EINVAL] The *speed* value is not a valid baud rate.22605 [EINVAL] The value of *speed* is outside the range of possible speed values as specified in  
22606 <termios.h>.22607 **EXAMPLES**

22608 None.

22609 **APPLICATION USAGE**

22610 None.

22611 **RATIONALE**22612 Refer to *cfgetispeed()*.22613 **FUTURE DIRECTIONS**

22614 None.

22615 **SEE ALSO**22616 *cfgetispeed()*, *cfgetospeed()*, *cfsetospeed()*, *tcsetattr()*

22617 XBD Chapter 11 (on page 199), &lt;termios.h&gt;

22618 **CHANGE HISTORY**

22619 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

22620 **Issue 6**22621 The following new requirements on POSIX implementations derive from alignment with the  
22622 Single UNIX Specification:

- 22623
- The optional setting of *errno* and the [EINVAL] error conditions are added.



22624 **NAME**

22625 cfsetospeed — set output baud rate

22626 **SYNOPSIS**

22627 #include &lt;termios.h&gt;

22628 int cfsetospeed(struct termios \*termios\_p, speed\_t speed);

22629 **DESCRIPTION**22630 The *cfsetospeed()* function shall set the output baud rate stored in the structure pointed to by  
22631 *termios\_p* to *speed*.22632 There shall be no effect on the baud rates set in the hardware until a subsequent successful call  
22633 to *tcsetattr()* with the same **termios** structure. Similarly, errors resulting from attempts to set  
22634 baud rates not supported by the terminal device need not be detected until the *tcsetattr()*  
22635 function is called.22636 **RETURN VALUE**22637 Upon successful completion, *cfsetospeed()* shall return 0; otherwise, it shall return -1 and *errno*  
22638 may be set to indicate the error.22639 **ERRORS**22640 The *cfsetospeed()* function may fail if:22641 [EINVAL] The *speed* value is not a valid baud rate.22642 [EINVAL] The value of *speed* is outside the range of possible speed values as specified in  
22643 <termios.h>.22644 **EXAMPLES**

22645 None.

22646 **APPLICATION USAGE**

22647 None.

22648 **RATIONALE**22649 Refer to *cfgetispeed()*.22650 **FUTURE DIRECTIONS**

22651 None.

22652 **SEE ALSO**22653 *cfgetispeed()*, *cfgetospeed()*, *cfsetispeed()*, *tcsetattr()*

22654 XBD Chapter 11 (on page 199), &lt;termios.h&gt;

22655 **CHANGE HISTORY**

22656 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

22657 **Issue 6**22658 The following new requirements on POSIX implementations derive from alignment with the  
22659 Single UNIX Specification:

- 22660
- The optional setting of *errno* and the [EINVAL] error conditions are added.

22661 **NAME**

22662 chdir — change working directory

22663 **SYNOPSIS**

22664 #include &lt;unistd.h&gt;

22665 int chdir(const char \*path);

22666 **DESCRIPTION**

22667 The *chdir()* function shall cause the directory named by the pathname pointed to by the *path*  
22668 argument to become the current working directory; that is, the starting point for path searches  
22669 for pathnames not beginning with '/ '.

22670 **RETURN VALUE**

22671 Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned, the current  
22672 working directory shall remain unchanged, and *errno* shall be set to indicate the error.

22673 **ERRORS**22674 The *chdir()* function shall fail if:

22675 [EACCES] Search permission is denied for any component of the pathname.

22676 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*  
22677 argument.

22678 [ENAMETOOLONG]

22679 The length of a component of a pathname is longer than {NAME\_MAX}.

22680 [ENOENT] A component of *path* does not name an existing directory or *path* is an empty  
22681 string.

22682 [ENOTDIR] A component of the pathname names an existing file that is neither a directory  
22683 nor a symbolic link to a directory.

22684 The *chdir()* function may fail if:

22685 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
22686 resolution of the *path* argument.

22687 [ENAMETOOLONG]

22688 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
22689 symbolic link produced an intermediate result with a length that exceeds  
22690 {PATH\_MAX}.

22691 **EXAMPLES**22692 **Changing the Current Working Directory**

22693 The following example makes the value pointed to by **directory**, **/tmp**, the current working  
22694 directory.

22695 #include &lt;unistd.h&gt;

22696 ...

22697 char \*directory = "/tmp";

22698 int ret;

22699 ret = chdir (directory);

22700 **APPLICATION USAGE**

22701 None.

22702 **RATIONALE**22703 The *chdir()* function only affects the working directory of the current process.22704 **FUTURE DIRECTIONS**

22705 None.

22706 **SEE ALSO**22707 *getcwd()*

22708 XBD &lt;unistd.h&gt;

22709 **CHANGE HISTORY**

22710 First released in Issue 1. Derived from Issue 1 of the SVID.

22711 **Issue 6**

22712 The APPLICATION USAGE section is added.

22713 The following new requirements on POSIX implementations derive from alignment with the  
22714 Single UNIX Specification:

- 22715 • The [ELOOP] mandatory error condition is added.
- 22716 • A second [ENAMETOOLONG] is added as an optional error condition.

22717 The following changes were made to align with the IEEE P1003.1a draft standard:

- 22718 • The [ELOOP] optional error condition is added.

22719 **Issue 7**

22720 Austin Group Interpretation 1003.1-2001 #143 is applied.

22721 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0047 [324] is applied.

22722 **NAME**

22723 chmod, fchmodat — change mode of a file |

22724 **SYNOPSIS**

22725 #include &lt;sys/stat.h&gt;

22726 int chmod(const char \*path, mode\_t mode);

22727 OH #include &lt;fcntl.h&gt; +

22728 int fchmodat(int fd, const char \*path, mode\_t mode, int flag);

22729 **DESCRIPTION**

22730 XSI The *chmod()* function shall change S\_ISUID, S\_ISGID, S\_ISVTX, and the file permission bits of  
 22731 the file named by the pathname pointed to by the *path* argument to the corresponding bits in the  
 22732 *mode* argument. The application shall ensure that the effective user ID of the process matches the  
 22733 owner of the file or the process has appropriate privileges in order to do this.

22734 XSI S\_ISUID, S\_ISGID, S\_ISVTX, and the file permission bits are described in <sys/stat.h>.

22735 If the calling process does not have appropriate privileges, and if the group ID of the file does  
 22736 not match the effective group ID or one of the supplementary group IDs and if the file is a  
 22737 regular file, bit S\_ISGID (set-group-ID on execution) in the file's mode shall be cleared upon  
 22738 successful return from *chmod()*.

22739 Additional implementation-defined restrictions may cause the S\_ISUID and S\_ISGID bits in  
 22740 *mode* to be ignored.

22741 Upon successful completion, *chmod()* shall mark for update the last file status change timestamp  
 22742 of the file.

22743 The *fchmodat()* function shall be equivalent to the *chmod()* function except in the case where *path*  
 22744 specifies a relative path. In this case the file to be changed is determined relative to the directory  
 22745 associated with the file descriptor *fd* instead of the current working directory. If the access mode  
 22746 of the open file description associated with the file descriptor is not O\_SEARCH, the function  
 22747 shall check whether directory searches are permitted using the current permissions of the  
 22748 directory underlying the file descriptor. If the access mode is O\_SEARCH, the function shall not  
 22749 perform the check.

22750 Values for *flag* are constructed by a bitwise-inclusive OR of flags from the following list, defined  
 22751 in <fcntl.h>:

22752 AT\_SYMLINK\_NOFOLLOW

22753 If *path* names a symbolic link, then the mode of the symbolic link is changed.

22754 If *fchmodat()* is passed the special value AT\_FDCWD in the *fd* parameter, the current working  
 22755 directory shall be used. If also *flag* is zero, the behavior shall be identical to a call to *chmod()*.

22756 **RETURN VALUE**

22757 Upon successful completion, these functions shall return 0. Otherwise, these functions shall  
 22758 return -1 and set *errno* to indicate the error. If -1 is returned, no change to the file mode occurs.

22759 **ERRORS**

22760 These functions shall fail if:

22761 [EACCES] Search permission is denied on a component of the path prefix.

22762 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*  
 22763 argument.

22764	[ENAMETOOLONG]	
22765		The length of a component of a pathname is longer than {NAME_MAX}.
22766	[ENOENT]	A component of <i>path</i> does not name an existing file or <i>path</i> is an empty string.
22767	[ENOTDIR]	A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>path</i> argument contains at least one non- <code>&lt;slash&gt;</code> character and ends with one or more trailing <code>&lt;slash&gt;</code> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.
22768		
22769		
22770		
22771		
22772	[EPERM]	The effective user ID does not match the owner of the file and the process does not have appropriate privileges.
22773		
22774	[EROFS]	The named file resides on a read-only file system.
22775	The <i>fchmodat()</i> function shall fail if:	
22776	[EACCES]	The access mode of the open file description associated with <i>fd</i> is not O_SEARCH and the permissions of the directory underlying <i>fd</i> do not permit directory searches.
22777		
22778		
22779	[EBADF]	The <i>path</i> argument does not specify an absolute path and the <i>fd</i> argument is neither AT_FDCWD nor a valid file descriptor open for reading or searching.
22780		
22781	[ENOTDIR]	The <i>path</i> argument is not an absolute path and <i>fd</i> is a file descriptor associated with a non-directory file.
22782		
22783	These functions may fail if:	
22784	[EINTR]	A signal was caught during execution of the function.
22785	[EINVAL]	The value of the <i>mode</i> argument is invalid.
22786	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the <i>path</i> argument.
22787		
22788	[ENAMETOOLONG]	
22789		The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.
22790		
22791		
22792	The <i>fchmodat()</i> function may fail if:	
22793	[EINVAL]	The value of the <i>flag</i> argument is invalid.
22794	[EOPNOTSUPP]	The AT_SYMLINK_NOFOLLOW bit is set in the <i>flag</i> argument, <i>path</i> names a symbolic link, and the system does not support changing the mode of a symbolic link.
22795		
22796		

## 22797 EXAMPLES

**Setting Read Permissions for User, Group, and Others**

The following example sets read permissions for the owner, group, and others.

```
#include <sys/stat.h>
const char *path;
...
chmod(path, S_IRUSR|S_IRGRP|S_IROTH);
```

**Setting Read, Write, and Execute Permissions for the Owner Only**

The following example sets read, write, and execute permissions for the owner, and no permissions for group and others.

```
#include <sys/stat.h>
const char *path;
...
chmod(path, S_IRWXU);
```

**Setting Different Permissions for Owner, Group, and Other**

The following example sets owner permissions for `CHANGEFILE` to read, write, and execute, group permissions to read and execute, and other permissions to read.

```
#include <sys/stat.h>
#define CHANGEFILE "/etc/myfile"
...
chmod(CHANGEFILE, S_IRWXU|S_IRGRP|S_IXGRP|S_IROTH);
```

**Setting and Checking File Permissions**

The following example sets the file permission bits for a file named `/home/cnd/mod1`, then calls the `stat()` function to verify the permissions.

```
#include <sys/types.h>
#include <sys/stat.h>
int status;
struct stat buffer
...
chmod("/home/cnd/mod1", S_IRWXU|S_IRWXG|S_IROTH|S_IWOTH);
status = stat("/home/cnd/mod1", &buffer);
```

**APPLICATION USAGE**

In order to ensure that the `S_ISUID` and `S_ISGID` bits are set, an application requiring this should use `stat()` after a successful `chmod()` to verify this.

Any file descriptors currently open by any process on the file could possibly become invalid if the mode of the file is changed to a value which would deny access to that process. One situation where this could occur is on a stateless file system. This behavior will not occur in a conforming environment.

**RATIONALE**

This volume of POSIX.1-2008 specifies that the S\_ISGID bit is cleared by *chmod()* on a regular file under certain conditions. This is specified on the assumption that regular files may be executed, and the system should prevent users from making executable *setgid()* files perform with privileges that the caller does not have. On implementations that support execution of other file types, the S\_ISGID bit should be cleared for those file types under the same circumstances.

Implementations that use the S\_ISUID bit to indicate some other function (for example, mandatory record locking) on non-executable files need not clear this bit on writing. They should clear the bit for executable files and any other cases where the bit grants special powers to processes that change the file contents. Similar comments apply to the S\_ISGID bit.

The purpose of the *fchmodat()* function is to enable changing the mode of files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *chmod()*, resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *fchmodat()* function it can be guaranteed that the changed file is located relative to the desired directory. Some implementations might allow changing the mode of symbolic links. This is not supported by the interfaces in the POSIX specification. Systems with such support provide an interface named *lchmod()*. To support such implementations *fchmodat()* has a *flag* parameter.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*access()*, *chown()*, *exec*, *fstatat()*, *fstatvfs()*, *mkdir()*, *mkfifo()*, *mknod()*, *open()*

XBD [\*<fcntl.h>\*](#), [\*<sys/stat.h>\*](#), [\*<sys/types.h>\*](#)

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 6**

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include [\*<sys/types.h>\*](#) has been removed. Although [\*<sys/types.h>\*](#) was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The [EINVAL] and [EINTR] optional error conditions are added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

The normative text is updated to avoid use of the term “must” for application requirements.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #143 is applied.

The *fchmodat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Changes are made related to support for finegrained timestamps.

Changes are made to allow a directory to be opened for searching.

The [ENOTDIR] error condition is clarified to cover the condition where the last component of a

22878            pathname exists but is not a directory or a symbolic link to a directory.

22879            POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0048 [300], XSH/TC1-2008/0049 [461],

22880            XSH/TC1-2008/0050 [324], XSH/TC1-2008/0051 [278], and XSH/TC1-2008/0052 [278] are

22881            applied.



## 22882 NAME

22883 chown, fchownat — change owner and group of a file |

## 22884 SYNOPSIS

22885 #include &lt;unistd.h&gt;

22886 int chown(const char \*path, uid\_t owner, gid\_t group);

22887 OH #include &lt;fcntl.h&gt; +

22888 +

22889 int fchownat(int fd, const char \*path, uid\_t owner, gid\_t group,  
22890 int flag);

## 22891 DESCRIPTION

22892 The *chown()* function shall change the user and group ownership of a file.22893 The *path* argument points to a pathname naming a file. The user ID and group ID of the named  
22894 file shall be set to the numeric values contained in *owner* and *group*, respectively.22895 Only processes with an effective user ID equal to the user ID of the file or with appropriate  
22896 privileges may change the ownership of a file. If `_POSIX_CHOWN_RESTRICTED` is in effect for  
22897 *path*:

- 22898
- Changing the user ID is restricted to processes with appropriate privileges.
  - Changing the group ID is permitted to a process with an effective user ID equal to the user  
22899 ID of the file, but without appropriate privileges, if and only if *owner* is equal to the file's  
22900 user ID or `(uid_t)-1` and *group* is equal either to the calling process' effective group ID or to  
22901 one of its supplementary group IDs.

22903 If the specified file is a regular file, one or more of the `S_IXUSR`, `S_IXGRP`, or `S_IXOTH` bits of  
22904 the file mode are set, and the process does not have appropriate privileges, the set-user-ID  
22905 (`S_ISUID`) and set-group-ID (`S_ISGID`) bits of the file mode shall be cleared upon successful  
22906 return from *chown()*. If the specified file is a regular file, one or more of the `S_IXUSR`, `S_IXGRP`,  
22907 or `S_IXOTH` bits of the file mode are set, and the process has appropriate privileges, it is  
22908 implementation-defined whether the set-user-ID and set-group-ID bits are altered. If the *chown()*  
22909 function is successfully invoked on a file that is not a regular file and one or more of the  
22910 `S_IXUSR`, `S_IXGRP`, or `S_IXOTH` bits of the file mode are set, the set-user-ID and set-group-ID  
22911 bits may be cleared.22912 If *owner* or *group* is specified as `(uid_t)-1` or `(gid_t)-1`, respectively, the corresponding ID of the  
22913 file shall not be changed. -22914 Upon successful completion, *chown()* shall mark for update the last file status change timestamp |  
22915 of the file, except that if *owner* is `(uid_t)-1` and *group* is `(gid_t)-1`, the file status change |  
22916 timestamp need not be marked for update.22917 The *fchownat()* function shall be equivalent to the *chown()* and *lchown()* functions except in the  
22918 case where *path* specifies a relative path. In this case the file to be changed is determined relative  
22919 to the directory associated with the file descriptor *fd* instead of the current working directory. If |  
22920 the access mode of the open file description associated with the file descriptor is not |  
22921 `O_SEARCH`, the function shall check whether directory searches are permitted using the current |  
22922 permissions of the directory underlying the file descriptor. If the access mode is `O_SEARCH`, the |  
22923 function shall not perform the check.22924 Values for *flag* are constructed by a bitwise-inclusive OR of flags from the following list, defined  
22925 in `<fcntl.h>`:

22926 AT\_SYMLINK\_NOFOLLOW

22927 If *path* names a symbolic link, ownership of the symbolic link is changed.

22928 If *fchownat()* is passed the special value AT\_FDCWD in the *fd* parameter, the current working  
22929 directory shall be used and the behavior shall be identical to a call to *chown()* or *lchown()*  
22930 respectively, depending on whether or not the AT\_SYMLINK\_NOFOLLOW bit is set in the *flag*  
22931 argument.

## 22932 RETURN VALUE

22933 Upon successful completion, these functions shall return 0. Otherwise, these functions shall  
22934 return -1 and set *errno* to indicate the error. If -1 is returned, no changes are made in the user ID  
22935 and group ID of the file.

## 22936 ERRORS

22937 These functions shall fail if:

22938 [EACCES] Search permission is denied on a component of the path prefix.

22939 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*  
22940 argument.

22941 [ENAMETOOLONG]

22942 The length of a component of a pathname is longer than {NAME\_MAX}.

22943 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.

22944 [ENOTDIR] A component of the path prefix names an existing file that is neither a  
22945 directory nor a symbolic link to a directory, or the *path* argument contains at  
22946 least one non-*<slash>* character and ends with one or more trailing *<slash>*  
22947 characters and the last pathname component names an existing file that is  
22948 neither a directory nor a symbolic link to a directory.

22949 [EPERM] The effective user ID does not match the owner of the file, or the calling  
22950 process does not have appropriate privileges and  
22951 \_POSIX\_CHOWN\_RESTRICTED indicates that such privilege is required.

22952 [EROFS] The named file resides on a read-only file system.

22953 The *fchownat()* function shall fail if:

22954 [EACCES] The access mode of the open file description associated with *fd* is not  
22955 O\_SEARCH and the permissions of the directory underlying *fd* do not permit  
22956 directory searches.

22957 [EBADF] The *path* argument does not specify an absolute path and the *fd* argument is  
22958 neither AT\_FDCWD nor a valid file descriptor open for reading or searching.

22959 [ENOTDIR] The *path* argument is not an absolute path and *fd* is a file descriptor associated  
22960 with a non-directory file.

22961 These functions may fail if:

22962 [EIO] An I/O error occurred while reading or writing to the file system.

22963 [EINTR] The *chown()* function was interrupted by a signal which was caught.

22964 [EINVAL] The owner or group ID supplied is not a value supported by the  
22965 implementation.

22966 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
 22967 resolution of the *path* argument.

22968 [ENAMETOOLONG]  
 22969 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
 22970 symbolic link produced an intermediate result with a length that exceeds  
 22971 {PATH\_MAX}.

22972 The *fchownat()* function may fail if:

22973 [EINVAL] The value of the *flag* argument is not valid.

## 22974 EXAMPLES

22975 None.

## 22976 APPLICATION USAGE

22977 Although *chown()* can be used on some implementations by the file owner to change the owner  
 22978 and group to any desired values, the only portable use of this function is to change the group of  
 22979 a file to the effective GID of the calling process or to a member of its group set.

## 22980 RATIONALE

22981 System III and System V allow a user to give away files; that is, the owner of a file may change  
 22982 its user ID to anything. This is a serious problem for implementations that are intended to meet  
 22983 government security regulations. Version 7 and 4.3 BSD permit only the superuser to change the  
 22984 user ID of a file. Some government agencies (usually not ones concerned directly with security)  
 22985 find this limitation too confining. This volume of POSIX.1-2008 uses *may* to permit secure  
 22986 implementations while not disallowing System V.

22987 System III and System V allow the owner of a file to change the group ID to anything. Version 7  
 22988 permits only the superuser to change the group ID of a file. 4.3 BSD permits the owner to  
 22989 change the group ID of a file to its effective group ID or to any of the groups in the list of  
 22990 supplementary group IDs, but to no others.

22991 The POSIX.1-1990 standard requires that the *chown()* function invoked by a non-appropriate  
 22992 privileged process clear the S\_ISGID and the S\_ISUID bits for regular files, and permits them to  
 22993 be cleared for other types of files. This is so that changes in accessibility do not accidentally  
 22994 cause files to become security holes. Unfortunately, requiring these bits to be cleared on non-  
 22995 executable data files also clears the mandatory file locking bit (shared with S\_ISGID), which is  
 22996 an extension on many implementations (it first appeared in System V). These bits should only be  
 22997 required to be cleared on regular files that have one or more of their execute bits set.

22998 The purpose of the *fchownat()* function is to enable changing ownership of files in directories  
 22999 other than the current working directory without exposure to race conditions. Any part of the  
 23000 path of a file could be changed in parallel to a call to *chown()* or *lchown()*, resulting in  
 23001 unspecified behavior. By opening a file descriptor for the target directory and using the  
 23002 *fchownat()* function it can be guaranteed that the changed file is located relative to the desired  
 23003 directory.

## 23004 FUTURE DIRECTIONS

23005 None.

## 23006 SEE ALSO

23007 *chmod()*, *fpathconf()*, *lchown()*

23008 XBD <fcntl.h>, <sys/types.h>, <unistd.h>

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 6**

The following changes are made for alignment with the ISO POSIX-1: 1996 standard:

- The wording describing the optional dependency on `_POSIX_CHOWN_RESTRICTED` is restored.
- The `[EPERM]` error is restored as an error dependent on `_POSIX_CHOWN_RESTRICTED`. This is since its operand is a pathname and applications should be aware that the error may not occur for that pathname if the file system does not support `_POSIX_CHOWN_RESTRICTED`.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The value for *owner* of `(uid_t)-1` allows the use of `-1` by the owner of a file to change the group ID only. A corresponding change is made for group.
- The `[ELOOP]` mandatory error condition is added.
- The `[EIO]` and `[EINTR]` optional error conditions are added.
- A second `[ENAMETOOLONG]` is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Clarification is added that the `S_ISUID` and `S_ISGID` bits do not need to be cleared when the process has appropriate privileges.
- The `[ELOOP]` optional error condition is added.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #143 is applied.

The `fchownat()` function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Changes are made related to support for finegrained timestamps.

Changes are made to allow a directory to be opened for searching.

The `[ENOTDIR]` error condition is clarified to cover the condition where the last component of a pathname exists but is not a directory or a symbolic link to a directory.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0053 [461], XSH/TC1-2008/0054 [324], XSH/TC1-2008/0055 [278], and XSH/TC1-2008/0056 [278] are applied.

23043 **NAME**

23044 cimag, cimagf, cimagl — complex imaginary functions

23045 **SYNOPSIS**

```
23046 #include <complex.h>
23047 double cimag(double complex z);
23048 float cimagf(float complex z);
23049 long double cimagl(long double complex z);
```

23050 **DESCRIPTION**

23051 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 23052 conflict between the requirements described here and the ISO C standard is unintentional. This  
 23053 volume of POSIX.1-2008 defers to the ISO C standard.

23054 These functions shall compute the imaginary part of *z*.23055 **RETURN VALUE**

23056 These functions shall return the imaginary part value (as a real).

23057 **ERRORS**

23058 No errors are defined.

23059 **EXAMPLES**

23060 None.

23061 **APPLICATION USAGE**23062 For a variable *z* of complex type:23063 `z == creal(z) + cimag(z)*I`23064 **RATIONALE**

23065 None.

23066 **FUTURE DIRECTIONS**

23067 None.

23068 **SEE ALSO**23069 *carg()*, *conj()*, *cproj()*, *creal()*23070 XBD **<complex.h>**23071 **CHANGE HISTORY**

23072 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

23073 **NAME**

23074 clearerr — clear indicators on a stream

23075 **SYNOPSIS**

23076 #include &lt;stdio.h&gt;

23077 void clearerr(FILE \*stream);

23078 **DESCRIPTION**

23079 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
23080 conflict between the requirements described here and the ISO C standard is unintentional. This  
23081 volume of POSIX.1-2008 defers to the ISO C standard.

23082 The *clearerr()* function shall clear the end-of-file and error indicators for the stream to which  
23083 *stream* points.

23084 CX The *clearerr()* function shall not change the setting of *errno* if *stream* is valid.

23085 **RETURN VALUE**23086 The *clearerr()* function shall not return a value.23087 **ERRORS**

23088 No errors are defined.

23089 **EXAMPLES**

23090 None.

23091 **APPLICATION USAGE**

23092 None.

23093 **RATIONALE**

23094 None.

23095 **FUTURE DIRECTIONS**

23096 None.

23097 **SEE ALSO**

23098 XBD &lt;stdio.h&gt;

23099 **CHANGE HISTORY**

23100 First released in Issue 1. Derived from Issue 1 of the SVID.

23101 **Issue 7**

23102 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0057 [401] is applied.

23103 **NAME**

23104 clock — report CPU time used

23105 **SYNOPSIS**

23106 #include &lt;time.h&gt;

23107 clock\_t clock(void);

23108 **DESCRIPTION**

23109 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 23110 conflict between the requirements described here and the ISO C standard is unintentional. This  
 23111 volume of POSIX.1-2008 defers to the ISO C standard.

23112 The *clock()* function shall return the implementation's best approximation to the processor time  
 23113 used by the process since the beginning of an implementation-defined era related only to the  
 23114 process invocation.

23115 **RETURN VALUE**

23116 To determine the time in seconds, the value returned by *clock()* should be divided by the value  
 23117 XSI of the macro `CLOCKS_PER_SEC`. `CLOCKS_PER_SEC` is defined to be one million in `<time.h>`.  
 23118 If the processor time used is not available or its value cannot be represented, the function shall  
 23119 return the value `(clock_t)-1`.

23120 **ERRORS**

23121 No errors are defined.

23122 **EXAMPLES**

23123 None.

23124 **APPLICATION USAGE**

23125 In programming environments where `clock_t` is a 32-bit integer type and `CLOCKS_PER_SEC` is  
 23126 one million, *clock()* will start failing in less than 36 minutes of processor time for signed `clock_t`,  
 23127 or 72 minutes for unsigned `clock_t`. Applications intended to be portable to such environments  
 23128 should use *times()* instead (or *clock\_gettime()* with `CLOCK_PROCESS_CPUTIME_ID`, if  
 23129 supported). +

23130 In order to measure the time spent in a program, *clock()* should be called at the start of the  
 23131 program and its return value subtracted from the value returned by subsequent calls. The value  
 23132 returned by *clock()* is defined for compatibility across systems that have clocks with different  
 23133 resolutions. The resolution on any particular system need not be to microsecond accuracy. -

23134 **RATIONALE**

23135 None.

23136 **FUTURE DIRECTIONS**

23137 None.

23138 **SEE ALSO**

23139 *asctime()*, *clock\_getres()*, *ctime()*, *difftime()*, *gmtime()*, *localtime()*, *mktime()*, *strftime()*, *strptime()*,  
 23140 *time()*, *times()*, *utime()*

23141 XBD `<time.h>`23142 **CHANGE HISTORY**

23143 First released in Issue 1. Derived from Issue 1 of the SVID.



23144 **NAME**23145 clock\_getcpuclockid — access a process CPU-time clock (**ADVANCED REALTIME**)23146 **SYNOPSIS**

```
23147 CPT    #include <time.h>
23148      int clock_getcpuclockid(pid_t pid, clockid_t *clock_id);
```

23149 **DESCRIPTION**

23150 The *clock\_getcpuclockid()* function shall return the clock ID of the CPU-time clock of the process  
 23151 specified by *pid*. If the process described by *pid* exists and the calling process has permission, the  
 23152 clock ID of this clock shall be returned in *clock\_id*.

23153 If *pid* is zero, the *clock\_getcpuclockid()* function shall return the clock ID of the CPU-time clock of  
 23154 the process making the call, in *clock\_id*.

23155 The conditions under which one process has permission to obtain the CPU-time clock ID of  
 23156 other processes are implementation-defined.

23157 **RETURN VALUE**

23158 Upon successful completion, *clock\_getcpuclockid()* shall return zero; otherwise, an error number  
 23159 shall be returned to indicate the error.

23160 **ERRORS**

23161 The *clock\_getcpuclockid()* function shall fail if:

23162 [EPERM] The requesting process does not have permission to access the CPU-time clock  
 23163 for the process.

23164 The *clock\_getcpuclockid()* function may fail if:

23165 [ESRCH] No process can be found corresponding to the process specified by *pid*.

23166 **EXAMPLES**

23167 None.

23168 **APPLICATION USAGE**

23169 The *clock\_getcpuclockid()* function is part of the Process CPU-Time Clocks option and need not be  
 23170 provided on all implementations.

23171 **RATIONALE**

23172 None.

23173 **FUTURE DIRECTIONS**

23174 None.

23175 **SEE ALSO**

23176 *clock\_getres()*, *timer\_create()*

23177 XBD **<time.h>**

23178 **CHANGE HISTORY**

23179 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

23180 In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.



23181 **NAME**

23182 clock\_getres, clock\_gettime, clock\_settime — clock and timer functions

23183 **SYNOPSIS**

```

23184 CX    #include <time.h>

23185    int clock_getres(clockid_t clock_id, struct timespec *res);
23186    int clock_gettime(clockid_t clock_id, struct timespec *tp);
23187    int clock_settime(clockid_t clock_id, const struct timespec *tp);

```

23188 **DESCRIPTION**

23189 The *clock\_getres()* function shall return the resolution of any clock. Clock resolutions are  
 23190 implementation-defined and cannot be set by a process. If the argument *res* is not NULL, the  
 23191 resolution of the specified clock shall be stored in the location pointed to by *res*. If *res* is NULL,  
 23192 the clock resolution is not returned. If the *time* argument of *clock\_settime()* is not a multiple of *res*,  
 23193 then the value is truncated to a multiple of *res*.

23194 The *clock\_gettime()* function shall return the current value *tp* for the specified clock, *clock\_id*.

23195 The *clock\_settime()* function shall set the specified clock, *clock\_id*, to the value specified by *tp*.  
 23196 Time values that are between two consecutive non-negative integer multiples of the resolution of  
 23197 the specified clock shall be truncated down to the smaller multiple of the resolution.

23198 A clock may be system-wide (that is, visible to all processes) or per-process (measuring time that  
 23199 is meaningful only within a process). All implementations shall support a *clock\_id* of  
 23200 CLOCK\_REALTIME as defined in **<time.h>**. This clock represents the clock measuring real time  
 23201 for the system. For this clock, the values returned by *clock\_gettime()* and specified by  
 23202 *clock\_settime()* represent the amount of time (in seconds and nanoseconds) since the Epoch. An  
 23203 implementation may also support additional clocks. The interpretation of time values for these  
 23204 clocks is unspecified.

23205 If the value of the CLOCK\_REALTIME clock is set via *clock\_settime()*, the new value of the clock  
 23206 shall be used to determine the time of expiration for absolute time services based upon the  
 23207 CLOCK\_REALTIME clock. This applies to the time at which armed absolute timers expire. If the  
 23208 absolute time requested at the invocation of such a time service is before the new value of the  
 23209 clock, the time service shall expire immediately as if the clock had reached the requested time  
 23210 normally.

23211 Setting the value of the CLOCK\_REALTIME clock via *clock\_settime()* shall have no effect on  
 23212 threads that are blocked waiting for a relative time service based upon this clock, including the  
 23213 *nanosleep()* function; nor on the expiration of relative timers based upon this clock.  
 23214 Consequently, these time services shall expire when the requested relative interval elapses,  
 23215 independently of the new or old value of the clock.

23216 MON If the Monotonic Clock option is supported, all implementations shall support a *clock\_id* of  
 23217 CLOCK\_MONOTONIC defined in **<time.h>**. This clock represents the monotonic clock for the  
 23218 system. For this clock, the value returned by *clock\_gettime()* represents the amount of time (in  
 23219 seconds and nanoseconds) since an unspecified point in the past (for example, system start-up  
 23220 time, or the Epoch). This point does not change after system start-up time. The value of the  
 23221 CLOCK\_MONOTONIC clock cannot be set via *clock\_settime()*. This function shall fail if it is  
 23222 invoked with a *clock\_id* argument of CLOCK\_MONOTONIC.

23223 The effect of setting a clock via *clock\_settime()* on armed per-process timers associated with a  
 23224 clock other than CLOCK\_REALTIME is implementation-defined.

23225 If the value of the CLOCK\_REALTIME clock is set via *clock\_settime()*, the new value of the clock  
 23226 shall be used to determine the time at which the system shall awaken a thread blocked on an

absolute *clock\_nanosleep()* call based upon the `CLOCK_REALTIME` clock. If the absolute time requested at the invocation of such a time service is before the new value of the clock, the call shall return immediately as if the clock had reached the requested time normally.

Setting the value of the `CLOCK_REALTIME` clock via *clock\_settime()* shall have no effect on any thread that is blocked on a relative *clock\_nanosleep()* call. Consequently, the call shall return when the requested relative interval elapses, independently of the new or old value of the clock.

Appropriate privileges to set a particular clock are implementation-defined.

CPT If `_POSIX_CPUTIME` is defined, implementations shall support clock ID values obtained by invoking *clock\_getcpuclockid()*, which represent the CPU-time clock of a given process. Implementations shall also support the special `clockid_t` value `CLOCK_PROCESS_CPUTIME_ID`, which represents the CPU-time clock of the calling process when invoking one of the *clock\_\**() or *timer\_\**() functions. For these clock IDs, the values returned by *clock\_gettime()* and specified by *clock\_settime()* represent the amount of execution time of the process associated with the clock. Changing the value of a CPU-time clock via *clock\_settime()* shall have no effect on the behavior of the sporadic server scheduling policy (see [Scheduling Policies](#)).

TCT If `_POSIX_THREAD_CPUTIME` is defined, implementations shall support clock ID values obtained by invoking *pthread\_getcpuclockid()*, which represent the CPU-time clock of a given thread. Implementations shall also support the special `clockid_t` value `CLOCK_THREAD_CPUTIME_ID`, which represents the CPU-time clock of the calling thread when invoking one of the *clock\_\**() or *timer\_\**() functions. For these clock IDs, the values returned by *clock\_gettime()* and specified by *clock\_settime()* shall represent the amount of execution time of the thread associated with the clock. Changing the value of a CPU-time clock via *clock\_settime()* shall have no effect on the behavior of the sporadic server scheduling policy (see [Scheduling Policies](#)).

## RETURN VALUE

A return value of 0 shall indicate that the call succeeded. A return value of `-1` shall indicate that an error occurred, and *errno* shall be set to indicate the error.

## ERRORS

The *clock\_getres()*, *clock\_gettime()*, and *clock\_settime()* functions shall fail if:

[EINVAL] The *clock\_id* argument does not specify a known clock.

The *clock\_gettime()* function shall fail if:

[EOVERFLOW] The number of seconds will not fit in an object of type `time_t`.

The *clock\_settime()* function shall fail if:

[EINVAL] The *tp* argument to *clock\_settime()* is outside the range for the given clock ID.

[EINVAL] The *tp* argument specified a nanosecond value less than zero or greater than or equal to 1 000 million.

MON [EINVAL] The value of the *clock\_id* argument is `CLOCK_MONOTONIC`.

The *clock\_settime()* function may fail if:

[EPERM] The requesting process does not have appropriate privileges to set the specified clock.

**EXAMPLES**

None.

**APPLICATION USAGE**

Note that the absolute value of the monotonic clock is meaningless (because its origin is arbitrary), and thus there is no need to set it. Furthermore, realtime applications can rely on the fact that the value of this clock is never set and, therefore, that time intervals measured with this clock will not be affected by calls to *clock\_settime()*.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

Scheduling Policies (on page 506), *clock\_getcpuclockid()*, *clock\_nanosleep()*, *ctime()*, *mq\_receive()*, *mq\_send()*, *nanosleep()*, *pthread\_mutex\_timedlock()*, *sem\_timedwait()*, *time()*, *timer\_create()*, *timer\_getoverrun()*

XBD **<time.h>**

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**Issue 6**

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Timers option.

The APPLICATION USAGE section is added.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Clarification is added of the effect of resetting the clock resolution.

CPU-time clocks and the *clock\_getcpuclockid()* function are added for alignment with IEEE Std 1003.1d-1999.

The following changes are added for alignment with IEEE Std 1003.1j-2000:

- The DESCRIPTION is updated as follows:
  - The value returned by *clock\_gettime()* for CLOCK\_MONOTONIC is specified.
  - The *clock\_settime()* function failing for CLOCK\_MONOTONIC is specified.
  - The effects of *clock\_settime()* on the *clock\_nanosleep()* function with respect to CLOCK\_REALTIME are specified.
- An [EINVAL] error is added to the ERRORS section, indicating that *clock\_settime()* fails for CLOCK\_MONOTONIC.
- The APPLICATION USAGE section notes that the CLOCK\_MONOTONIC clock need not and shall not be set by *clock\_settime()* since the absolute value of the CLOCK\_MONOTONIC clock is meaningless.
- The *clock\_nanosleep()*, *mq\_timedreceive()*, *mq\_timedsend()*, *pthread\_mutex\_timedlock()*, *sem\_timedwait()*, *timer\_create()*, and *timer\_settime()* functions are added to the SEE ALSO section.

23308 **Issue 7**

23309       Functionality relating to the Clock Selection option is moved to the Base.

23310       The *clock\_getres()*, *clock\_gettime()*, and *clock\_settime()* functions are moved from the Timers  
23311       option to the Base.

23312       POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0058 [106] is applied.

23313 **NAME**

23314 clock\_nanosleep — high resolution sleep with specifiable clock

23315 **SYNOPSIS**

```
23316 CX      #include <time.h>
23317
23317      int clock_nanosleep(clockid_t clock_id, int flags,
23318                          const struct timespec *rqtp, struct timespec *rmtp);
```

23319 **DESCRIPTION**

23320 If the flag `TIMER_ABSTIME` is not set in the *flags* argument, the *clock\_nanosleep()* function shall  
23321 cause the current thread to be suspended from execution until either the time interval specified  
23322 by the *rqtp* argument has elapsed, or a signal is delivered to the calling thread and its action is to  
23323 invoke a signal-catching function, or the process is terminated. The clock used to measure the  
23324 time shall be the clock specified by *clock\_id*.

23325 If the flag `TIMER_ABSTIME` is set in the *flags* argument, the *clock\_nanosleep()* function shall  
23326 cause the current thread to be suspended from execution until either the time value of the clock  
23327 specified by *clock\_id* reaches the absolute time specified by the *rqtp* argument, or a signal is  
23328 delivered to the calling thread and its action is to invoke a signal-catching function, or the  
23329 process is terminated. If, at the time of the call, the time value specified by *rqtp* is less than or  
23330 equal to the time value of the specified clock, then *clock\_nanosleep()* shall return immediately  
23331 and the calling process shall not be suspended.

23332 The suspension time caused by this function may be longer than requested because the  
23333 argument value is rounded up to an integer multiple of the sleep resolution, or because of the  
23334 scheduling of other activity by the system. But, except for the case of being interrupted by a  
23335 signal, the suspension time for the relative *clock\_nanosleep()* function (that is, with the  
23336 `TIMER_ABSTIME` flag not set) shall not be less than the time interval specified by *rqtp*, as  
23337 measured by the corresponding clock. The suspension for the absolute *clock\_nanosleep()* function  
23338 (that is, with the `TIMER_ABSTIME` flag set) shall be in effect at least until the value of the  
23339 corresponding clock reaches the absolute time specified by *rqtp*, except for the case of being  
23340 interrupted by a signal.

23341 The use of the *clock\_nanosleep()* function shall have no effect on the action or blockage of any  
23342 signal.

23343 The *clock\_nanosleep()* function shall fail if the *clock\_id* argument refers to the CPU-time clock of  
23344 the calling thread. It is unspecified whether *clock\_id* values of other CPU-time clocks are allowed.

23345 **RETURN VALUE**

23346 If the *clock\_nanosleep()* function returns because the requested time has elapsed, its return value  
23347 shall be zero.

23348 If the *clock\_nanosleep()* function returns because it has been interrupted by a signal, it shall  
23349 return the corresponding error value. For the relative *clock\_nanosleep()* function, if the *rmtp*  
23350 argument is non-NULL, the **timespec** structure referenced by it shall be updated to contain the  
23351 amount of time remaining in the interval (the requested time minus the time actually slept). The  
23352 *rqtp* and *rmtp* arguments can point to the same object. If the *rmtp* argument is NULL, the  
23353 remaining time is not returned. The absolute *clock\_nanosleep()* function has no effect on the  
23354 structure referenced by *rmtp*.

23355 If *clock\_nanosleep()* fails, it shall return the corresponding error value.

**ERRORS**

The *clock\_nanosleep()* function shall fail if:

- [EINTR] The *clock\_nanosleep()* function was interrupted by a signal.
- [EINVAL] The *rqtp* argument specified a nanosecond value less than zero or greater than or equal to 1 000 million; or the `TIMER_ABSTIME` flag was specified in *flags* and the *rqtp* argument is outside the range for the clock specified by *clock\_id*; or the *clock\_id* argument does not specify a known clock, or specifies the CPU-time clock of the calling thread.
- [ENOTSUP] The *clock\_id* argument specifies a clock for which *clock\_nanosleep()* is not supported, such as a CPU-time clock.

**EXAMPLES**

None.

**APPLICATION USAGE**

Calling *clock\_nanosleep()* with the value `TIMER_ABSTIME` not set in the *flags* argument and with a *clock\_id* of `CLOCK_REALTIME` is equivalent to calling *nanosleep()* with the same *rqtp* and *rmtpt* arguments.

**RATIONALE**

The *nanosleep()* function specifies that the system-wide clock `CLOCK_REALTIME` is used to measure the elapsed time for this time service. However, with the introduction of the monotonic clock `CLOCK_MONOTONIC` a new relative sleep function is needed to allow an application to take advantage of the special characteristics of this clock.

There are many applications in which a process needs to be suspended and then activated multiple times in a periodic way; for example, to poll the status of a non-interrupting device or to refresh a display device. For these cases, it is known that precise periodic activation cannot be achieved with a relative *sleep()* or *nanosleep()* function call. Suppose, for example, a periodic process that is activated at time  $T_0$ , executes for a while, and then wants to suspend itself until time  $T_0+T$ , the period being  $T$ . If this process wants to use the *nanosleep()* function, it must first call *clock\_gettime()* to get the current time, then calculate the difference between the current time and  $T_0+T$  and, finally, call *nanosleep()* using the computed interval. However, the process could be preempted by a different process between the two function calls, and in this case the interval computed would be wrong; the process would wake up later than desired. This problem would not occur with the absolute *clock\_nanosleep()* function, since only one function call would be necessary to suspend the process until the desired time. In other cases, however, a relative sleep is needed, and that is why both functionalities are required.

Although it is possible to implement periodic processes using the timers interface, this implementation would require the use of signals, and the reservation of some signal numbers. In this regard, the reasons for including an absolute version of the *clock\_nanosleep()* function in POSIX.1-2008 are the same as for the inclusion of the relative *nanosleep()*.

It is also possible to implement precise periodic processes using *pthread\_cond\_timedwait()*, in which an absolute timeout is specified that takes effect if the condition variable involved is never signaled. However, the use of this interface is unnatural, and involves performing other operations on mutexes and condition variables that imply an unnecessary overhead. Furthermore, *pthread\_cond\_timedwait()* is not available in implementations that do not support threads.

Although the interface of the relative and absolute versions of the new high resolution sleep service is the same *clock\_nanosleep()* function, the *rmtpt* argument is only used in the relative sleep. This argument is needed in the relative *clock\_nanosleep()* function to reissue the function

23403 call if it is interrupted by a signal, but it is not needed in the absolute *clock\_nanosleep()* function  
23404 call; if the call is interrupted by a signal, the absolute *clock\_nanosleep()* function can be invoked  
23405 again with the same *rqtp* argument used in the interrupted call.

#### 23406 FUTURE DIRECTIONS

23407 None.

#### 23408 SEE ALSO

23409 *clock\_getres()*, *nanosleep()*, *pthread\_cond\_timedwait()*, *sleep()*

23410 XBD <time.h>

#### 23411 CHANGE HISTORY

23412 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

#### 23413 Issue 7

23414 The *clock\_nanosleep()* function is moved from the Clock Selection option to the Base.



23415 **NAME**

23416 clock\_gettime — clock and timer functions

23417 **SYNOPSIS**

```
23418 CX      #include <time.h>
23419      int clock_gettime(clockid_t clock_id, const struct timespec *tp);
```

23420 **DESCRIPTION**

23421 Refer to *clock\_getres()*.



23422 **NAME**

23423 clog, clogf, clogl — complex natural logarithm functions

23424 **SYNOPSIS**

```
23425 #include <complex.h>
23426 double complex clog(double complex z);
23427 float complex clogf(float complex z);
23428 long double complex clogl(long double complex z);
```

23429 **DESCRIPTION**

23430 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 23431 conflict between the requirements described here and the ISO C standard is unintentional. This  
 23432 volume of POSIX.1-2008 defers to the ISO C standard.

23433 These functions shall compute the complex natural (base  $e$ ) logarithm of  $z$ , with a branch cut  
 23434 along the negative real axis.

23435 **RETURN VALUE**

23436 These functions shall return the complex natural logarithm value, in the range of a strip  
 23437 mathematically unbounded along the real axis and in the interval  $[-i\pi, +i\pi]$  along the imaginary  
 23438 axis.

23439 **ERRORS**

23440 No errors are defined.

23441 **EXAMPLES**

23442 None.

23443 **APPLICATION USAGE**

23444 None.

23445 **RATIONALE**

23446 None.

23447 **FUTURE DIRECTIONS**

23448 None.

23449 **SEE ALSO**23450 [cexp\(\)](#)23451 XBD [<complex.h>](#)23452 **CHANGE HISTORY**

23453 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

23454 **NAME**

23455 close — close a file descriptor

23456 **SYNOPSIS**

23457 #include &lt;unistd.h&gt;

23458 int close(int *fildes*);23459 **DESCRIPTION**

23460 The *close()* function shall deallocate the file descriptor indicated by *fildes*. To deallocate means to  
 23461 make the file descriptor available for return by subsequent calls to *open()* or other functions that  
 23462 allocate file descriptors. All outstanding record locks owned by the process on the file associated  
 23463 with the file descriptor shall be removed (that is, unlocked).

23464 If *close()* is interrupted by a signal that is to be caught, it shall return  $-1$  with *errno* set to [EINTR]  
 23465 and the state of *fildes* is unspecified. If an I/O error occurred while reading from or writing to  
 23466 the file system during *close()*, it may return  $-1$  with *errno* set to [EIO]; if this error is returned, the  
 23467 state of *fildes* is unspecified.

23468 When all file descriptors associated with a pipe or FIFO special file are closed, any data  
 23469 remaining in the pipe or FIFO shall be discarded.

23470 When all file descriptors associated with an open file description have been closed, the open file  
 23471 description shall be freed.

23472 If the link count of the file is 0, when all file descriptors associated with the file are closed, the  
 23473 space occupied by the file shall be freed and the file shall no longer be accessible.

23474 OB XSR If a STREAMS-based *fildes* is closed and the calling process was previously registered to receive  
 23475 a SIGPOLL signal for events associated with that STREAM, the calling process shall be  
 23476 unregistered for events associated with the STREAM. The last *close()* for a STREAM shall cause  
 23477 the STREAM associated with *fildes* to be dismantled. If O\_NONBLOCK is not set and there have  
 23478 been no signals posted for the STREAM, and if there is data on the module's write queue, *close()*  
 23479 shall wait for an unspecified time (for each module and driver) for any output to drain before  
 23480 dismantling the STREAM. The time delay can be changed via an I\_SETCLTIME *ioctl()* request. If  
 23481 the O\_NONBLOCK flag is set, or if there are any pending signals, *close()* shall not wait for  
 23482 output to drain, and shall dismantle the STREAM immediately.

23483 If the implementation supports STREAMS-based pipes, and *fildes* is associated with one end of a  
 23484 pipe, the last *close()* shall cause a hangup to occur on the other end of the pipe. In addition, if the  
 23485 other end of the pipe has been named by *fattach()*, then the last *close()* shall force the named end  
 23486 to be detached by *fdetach()*. If the named end has no open file descriptors associated with it and  
 23487 gets detached, the STREAM associated with that end shall also be dismantled.

23488 XSI If *fildes* refers to the master side of a pseudo-terminal, and this is the last close, a SIGHUP signal  
 23489 shall be sent to the controlling process, if any, for which the slave side of the pseudo-terminal is  
 23490 the controlling terminal. It is unspecified whether closing the master side of the pseudo-terminal  
 23491 flushes all queued input and output.

23492 OB XSR If *fildes* refers to the slave side of a STREAMS-based pseudo-terminal, a zero-length message  
 23493 may be sent to the master.

23494 When there is an outstanding cancelable asynchronous I/O operation against *fildes* when *close()*  
 23495 is called, that I/O operation may be canceled. An I/O operation that is not canceled completes  
 23496 as if the *close()* operation had not yet occurred. All operations that are not canceled shall  
 23497 complete as if the *close()* blocked until the operations completed. The *close()* operation itself  
 23498 need not block awaiting such I/O completion. Whether any I/O operation is canceled, and  
 23499 which I/O operation may be canceled upon *close()*, is implementation-defined.

23500 SHM If a memory mapped file or a shared memory object remains referenced at the last close (that is,  
 23501 a process has it mapped), then the entire contents of the memory object shall persist until the  
 23502 SHM memory object becomes unreferenced. If this is the last close of a memory mapped file or a  
 23503 shared memory object and the close results in the memory object becoming unreferenced, and  
 23504 the memory object has been unlinked, then the memory object shall be removed.

23505 If *fildest* refers to a socket, *close()* shall cause the socket to be destroyed. If the socket is in  
 23506 connection-mode, and the *SO\_LINGER* option is set for the socket with non-zero linger time,  
 23507 and the socket has untransmitted data, then *close()* shall block for up to the current linger  
 23508 interval until all data is transmitted.

#### 23509 RETURN VALUE

23510 Upon successful completion, 0 shall be returned; otherwise, -1 shall be returned and *errno* set to  
 23511 indicate the error.

#### 23512 ERRORS

23513 The *close()* function shall fail if:

23514 [EBADF] The *fildest* argument is not a open file descriptor.

23515 [EINTR] The *close()* function was interrupted by a signal.

23516 The *close()* function may fail if:

23517 [EIO] An I/O error occurred while reading from or writing to the file system.

#### 23518 EXAMPLES

##### 23519 Reassigning a File Descriptor

23520 The following example closes the file descriptor associated with standard output for the current  
 23521 process, re-assigns standard output to a new file descriptor, and closes the original file descriptor  
 23522 to clean up. This example assumes that the file descriptor 0 (which is the descriptor for standard  
 23523 input) is not closed.

```
23524 #include <unistd.h>
23525 ...
23526 int pfd;
23527 ...
23528 close(1);
23529 dup(pfd);
23530 close(pfd);
23531 ...
```

23532 Incidentally, this is exactly what could be achieved using:

```
23533 dup2(pfd, 1);
23534 close(pfd);
```

## Closing a File Descriptor

In the following example, *close()* is used to close a file descriptor after an unsuccessful attempt is made to associate that file descriptor with a stream.

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>

#define LOCKFILE "/etc/ptmp"
...
int pfd;
FILE *fpfd;
...
if ((fpfd = fdopen (pfd, "w")) == NULL) {
    close(pfd);
    unlink(LOCKFILE);
    exit(1);
}
...
```

## APPLICATION USAGE

An application that had used the *stdio* routine *fopen()* to open a file should use the corresponding *fclose()* routine rather than *close()*. Once a file is closed, the file descriptor no longer exists, since the integer corresponding to it no longer refers to a file.

Implementations may use file descriptors that must be inherited into child processes for the child process to remain conforming, such as for message catalog or tracing purposes. Therefore, an application that calls *close()* on an arbitrary integer risks non-conforming behavior, and *close()* can only portably be used on file descriptor values that the application has obtained through explicit actions, as well as the three file descriptors corresponding to the standard file streams. In multi-threaded parent applications, the practice of calling *close()* in a loop after *fork()* and before an *exec* call in order to avoid a race condition of leaking an unintended file descriptor into a child process, is therefore unsafe, and the race should instead be combatted by opening all file descriptors with the *FD\_CLOEXEC* bit set unless the file descriptor is intended to be inherited across *exec*.

Usage of *close()* on file descriptors *STDIN\_FILENO*, *STDOUT\_FILENO*, or *STDERR\_FILENO* + should immediately be followed by an operation to reopen these file descriptors. Unexpected + behavior will result if any of these file descriptors is left in a closed state (for example, an + [EBADF] error from *perror()*) or if an unrelated *open()* or similar call later in the application + accidentally allocates a file to one of these well-known file descriptors. Furthermore, a *close()* + followed by a reopen operation (e.g., *open()*, *dup()*, etc.) is not atomic; *dup2()* should be used to + change standard file descriptors.

## RATIONALE

The use of interruptible device close routines should be discouraged to avoid problems with the implicit closes of file descriptors by *exec* and *exit()*. This volume of POSIX.1-2008 only intends to permit such behavior by specifying the [EINTR] error condition.

Note that the requirement for *close()* on a socket to block for up to the current linger interval is not conditional on the *O\_NONBLOCK* setting.

The standard developers rejected a proposal to add *closefrom()* to the standard. Because the standard permits implementations to use inherited file descriptors as a means of providing a conforming environment for the child process, it is not possible to standardize an interface that

23582 closes arbitrary file descriptors above a certain value while still guaranteeing a conforming  
23583 environment.

#### 23584 FUTURE DIRECTIONS

23585 None.

#### 23586 SEE ALSO

23587 Section 2.6 (on page 500), *dup()*, *exec*, *exit()*, *fattach()*, *fclose()*, *fdetach()*, *fopen()*, *fork()*, *ioctl()*,  
23588 *open()*, *perror()*, *unlink()*

23589 XBD <unistd.h>

#### 23590 CHANGE HISTORY

23591 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 23592 Issue 5

23593 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension.

#### 23594 Issue 6

23595 The DESCRIPTION related to a STREAMS-based file or pseudo-terminal is marked as part of  
23596 the XSI STREAMS Option Group.

23597 The following new requirements on POSIX implementations derive from alignment with the  
23598 Single UNIX Specification:

- 23599 • The [EIO] error condition is added as an optional error.
- 23600 • The DESCRIPTION is updated to describe the state of the *fildev* file descriptor as  
23601 unspecified if an I/O error occurs and an [EIO] error condition is returned.

23602 Text referring to sockets is added to the DESCRIPTION.

23603 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that  
23604 shared memory objects and memory mapped files (and not typed memory objects) are the types  
23605 of memory objects to which the paragraph on last closes applies.

23606 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/12 is applied, correcting the XSH shaded  
23607 text relating to the master side of a pseudo-terminal. The reason for the change is that the  
23608 behavior of pseudo-terminals and regular terminals should be as much alike as possible in this  
23609 case; the change achieves that and matches historical behavior.

#### 23610 Issue 7

23611 Functionality relating to the XSI STREAMS option is marked obsolescent.

23612 Functionality relating to the Asynchronous Input and Output and Memory Mapped Files  
23613 options is moved to the Base.

23614 Austin Group Interpretation 1003.1-2001 #139 is applied, clarifying that the requirement for  
23615 *close()* on a socket to block for up to the current linger interval is not conditional on the  
23616 O\_NONBLOCK setting.

23617 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0059 [419], XSH/TC1-2008/0060 [149],  
23618 and XSH/TC1-2008/0061 [149] are applied.

23619 **NAME**

23620       closedir — close a directory stream

23621 **SYNOPSIS**

23622       #include &lt;dirent.h&gt;

23623       int closedir(DIR \*dirp);

23624 **DESCRIPTION**

23625       The *closedir()* function shall close the directory stream referred to by the argument *dirp*. Upon  
23626       return, the value of *dirp* may no longer point to an accessible object of the type **DIR**. If a file  
23627       descriptor is used to implement type **DIR**, that file descriptor shall be closed.

23628 **RETURN VALUE**

23629       Upon successful completion, *closedir()* shall return 0; otherwise, -1 shall be returned and *errno*  
23630       set to indicate the error.

23631 **ERRORS**23632       The *closedir()* function may fail if:23633       [EBADF]       The *dirp* argument does not refer to an open directory stream.23634       [EINTR]       The *closedir()* function was interrupted by a signal.23635 **EXAMPLES**23636       **Closing a Directory Stream**23637       The following program fragment demonstrates how the *closedir()* function is used.

```
23638       ...
23639       DIR *dir;
23640       struct dirent *dp;
23641       ...
23642       if ((dir = opendir(".")) == NULL) {
23643       ...
23644       }
23645       while ((dp = readdir (dir)) != NULL) {
23646       ...
23647       }
23648       closedir(dir);
23649       ...
```

23650 **APPLICATION USAGE**

23651       None.

23652 **RATIONALE**

23653       None.

23654 **FUTURE DIRECTIONS**

23655       None.

23656 **SEE ALSO**23657       *dirfd()*, *fdopendir()*

23658       XBD &lt;dirent.h&gt;

23659 **CHANGE HISTORY**

23660 First released in Issue 2.

23661 **Issue 6**23662 In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.23663 The following new requirements on POSIX implementations derive from alignment with the  
23664 Single UNIX Specification:

- 23665 • The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was  
23666 required for conforming implementations of previous POSIX specifications, it was not  
23667 required for UNIX applications.
- 23668 • The [EINTR] error condition is added as an optional error condition.

23669 **NAME**

23670 closelog, openlog, setlogmask, syslog — control system log

23671 **SYNOPSIS**

```

23672 XSI #include <syslog.h>
23673
23673 void closelog(void);
23674 void openlog(const char *ident, int logopt, int facility);
23675 int setlogmask(int maskpri);
23676 void syslog(int priority, const char *message, ... /* arguments */);

```

23677 **DESCRIPTION**

23678 The *syslog()* function shall send a message to an implementation-defined logging facility, which  
 23679 may log it in an implementation-defined system log, write it to the system console, forward it to  
 23680 a list of users, or forward it to the logging facility on another host over the network. The logged  
 23681 message shall include a message header and a message body. The message header contains at  
 23682 least a timestamp and a tag string.

23683 The message body is generated from the *message* and following arguments in the same manner  
 23684 as if these were arguments to *printf()*, except that the additional conversion specification *%m*  
 23685 shall be recognized; it shall convert no arguments, shall cause the output of the error message  
 23686 string associated with the value of *errno* on entry to *syslog()*, and may be mixed with argument  
 23687 specifications of the "*%n\$*" form. If a complete conversion specification with the *m* conversion  
 23688 specifier character is not just *%m*, the behavior is undefined. A trailing <newline> may be added  
 23689 if needed.

23690 Values of the *priority* argument are formed by OR'ing together a severity-level value and an  
 23691 optional facility value. If no facility value is specified, the current default facility value is used.

23692 Possible values of severity level include:

23693	LOG_EMERG	A panic condition.
23694	LOG_ALERT	A condition that should be corrected immediately, such as a corrupted system
23695		database.
23696	LOG_CRIT	Critical conditions, such as hard device errors.
23697	LOG_ERR	Errors.
23698	LOG_WARNING	
23699		Warning messages.
23700	LOG_NOTICE	Conditions that are not error conditions, but that may require special
23701		handling.
23702	LOG_INFO	Informational messages.
23703	LOG_DEBUG	Messages that contain information normally of use only when debugging a
23704		program.

23705 The facility indicates the application or system component generating the message. Possible  
 23706 facility values include:

23707	LOG_USER	Messages generated by arbitrary processes. This is the default facility
23708		identifier if none is specified.
23709	LOG_LOCAL0	Reserved for local use.



23710	LOG_LOCAL1	Reserved for local use.
23711	LOG_LOCAL2	Reserved for local use.
23712	LOG_LOCAL3	Reserved for local use.
23713	LOG_LOCAL4	Reserved for local use.
23714	LOG_LOCAL5	Reserved for local use.
23715	LOG_LOCAL6	Reserved for local use.
23716	LOG_LOCAL7	Reserved for local use.

23717 The *openlog()* function shall set process attributes that affect subsequent calls to *syslog()*. The  
 23718 *ident* argument is a string that is prepended to every message. The *logopt* argument indicates  
 23719 logging options. Values for *logopt* are constructed by a bitwise-inclusive OR of zero or more of  
 23720 the following:

23721	LOG_PID	Log the process ID with each message. This is useful for identifying specific
23722		processes.
23723	LOG_CONS	Write messages to the system console if they cannot be sent to the logging
23724		facility. The <i>syslog()</i> function ensures that the process does not acquire the
23725		console as a controlling terminal in the process of writing the message.
23726	LOG_NDELAY	Open the connection to the logging facility immediately. Normally the open is
23727		delayed until the first message is logged. This is useful for programs that need
23728		to manage the order in which file descriptors are allocated.
23729	LOG_ODELAY	Delay open until <i>syslog()</i> is called.
23730	LOG_NOWAIT	Do not wait for child processes that may have been created during the course
23731		of logging the message. This option should be used by processes that enable
23732		notification of child termination using SIGCHLD, since <i>syslog()</i> may
23733		otherwise block waiting for a child whose exit status has already been
23734		collected.

23735 The *facility* argument encodes a default facility to be assigned to all messages that do not have an  
 23736 explicit facility already encoded. The initial default facility is LOG\_USER.

23737 The *openlog()* and *syslog()* functions may allocate a file descriptor. It is not necessary to call  
 23738 *openlog()* prior to calling *syslog()*.

23739 The *closelog()* function shall close any open file descriptors allocated by previous calls to  
 23740 *openlog()* or *syslog()*.

23741 The *setlogmask()* function shall set the log priority mask for the current process to *maskpri* and  
 23742 return the previous mask. If the *maskpri* argument is 0, the current log mask is not modified.  
 23743 Calls by the current process to *syslog()* with a priority not set in *maskpri* shall be rejected. The  
 23744 default log mask allows all priorities to be logged. A call to *openlog()* is not required prior to  
 23745 calling *setlogmask()*.

23746 Symbolic constants for use as values of the *logopt*, *facility*, *priority*, and *maskpri* arguments are  
 23747 defined in the **<syslog.h>** header.

## 23748 RETURN VALUE

23749 The *setlogmask()* function shall return the previous log priority mask. The *closelog()*, *openlog()*,  
 23750 and *syslog()* functions shall not return a value.

23751 **ERRORS**

23752       No errors are defined.

23753 **EXAMPLES**23754       **Using openlog()**

23755       The following example causes subsequent calls to *syslog()* to log the process ID with each  
23756       message, and to write messages to the system console if they cannot be sent to the logging  
23757       facility.

```
23758       #include <syslog.h>

23759       char *ident = "Process demo";
23760       int logopt = LOG_PID | LOG_CONS;
23761       int facility = LOG_USER;
23762       ...
23763       openlog(ident, logopt, facility);
```

23764       **Using setlogmask()**

23765       The following example causes subsequent calls to *syslog()* to accept error messages, and to reject  
23766       all other messages.

```
23767       #include <syslog.h>

23768       int result;
23769       int mask = LOG_MASK (LOG_ERR);
23770       ...
23771       result = setlogmask(mask);
```

23772       **Using syslog**

23773       The following example sends the message "This is a message" to the default logging  
23774       facility, marking the message as an error message generated by random processes.

```
23775       #include <syslog.h>

23776       char *message = "This is a message";
23777       int priority = LOG_ERR | LOG_USER;
23778       ...
23779       syslog(priority, message);
```

23780 **APPLICATION USAGE**

23781       None.

23782 **RATIONALE**

23783       None.

23784 **FUTURE DIRECTIONS**

23785       None.

23786 **SEE ALSO**23787       *fprintf()*23788       XBD *<syslog.h>*

**CHANGE HISTORY**

23789 First released in Issue 4, Version 2.

**Issue 5**

23791 Moved from X/OPEN UNIX extension to BASE.

**Issue 6**

23793 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/13 is applied, correcting the EXAMPLES  
23794 section.  
23795

23796 **NAME**

23797 confstr — get configurable variables

23798 **SYNOPSIS**

23799 #include &lt;unistd.h&gt;

23800 size\_t confstr(int name, char \*buf, size\_t len);

23801 **DESCRIPTION**

23802 The *confstr()* function shall return configuration-defined string values. Its use and purpose are  
 23803 similar to *sysconf()*, but it is used where string values rather than numeric values are returned.

23804 The *name* argument represents the system variable to be queried. The implementation shall  
 23805 support the following name values, defined in <unistd.h>. It may support others:

23806 \_CS\_PATH  
 23807 \_CS\_POSIX\_V7\_ILP32\_OFF32\_CFLAGS  
 23808 \_CS\_POSIX\_V7\_ILP32\_OFF32\_LDFLAGS  
 23809 \_CS\_POSIX\_V7\_ILP32\_OFF32\_LIBS  
 23810 \_CS\_POSIX\_V7\_ILP32\_OFFBIG\_CFLAGS  
 23811 \_CS\_POSIX\_V7\_ILP32\_OFFBIG\_LDFLAGS  
 23812 \_CS\_POSIX\_V7\_ILP32\_OFFBIG\_LIBS  
 23813 \_CS\_POSIX\_V7\_LP64\_OFF64\_CFLAGS  
 23814 \_CS\_POSIX\_V7\_LP64\_OFF64\_LDFLAGS  
 23815 \_CS\_POSIX\_V7\_LP64\_OFF64\_LIBS  
 23816 \_CS\_POSIX\_V7\_LPBIG\_OFFBIG\_CFLAGS  
 23817 \_CS\_POSIX\_V7\_LPBIG\_OFFBIG\_LDFLAGS  
 23818 \_CS\_POSIX\_V7\_LPBIG\_OFFBIG\_LIBS  
 23819 \_CS\_POSIX\_V7\_THREADS\_CFLAGS  
 23820 \_CS\_POSIX\_V7\_THREADS\_LDFLAGS  
 23821 \_CS\_POSIX\_V7\_WIDTH\_RESTRICTED\_ENVS  
 23822 \_CS\_V7\_ENV  
 23823 OB \_CS\_POSIX\_V6\_ILP32\_OFF32\_CFLAGS  
 23824 \_CS\_POSIX\_V6\_ILP32\_OFF32\_LDFLAGS  
 23825 \_CS\_POSIX\_V6\_ILP32\_OFF32\_LIBS  
 23826 \_CS\_POSIX\_V6\_ILP32\_OFFBIG\_CFLAGS  
 23827 \_CS\_POSIX\_V6\_ILP32\_OFFBIG\_LDFLAGS  
 23828 \_CS\_POSIX\_V6\_ILP32\_OFFBIG\_LIBS  
 23829 \_CS\_POSIX\_V6\_LP64\_OFF64\_CFLAGS  
 23830 \_CS\_POSIX\_V6\_LP64\_OFF64\_LDFLAGS  
 23831 \_CS\_POSIX\_V6\_LP64\_OFF64\_LIBS  
 23832 \_CS\_POSIX\_V6\_LPBIG\_OFFBIG\_CFLAGS  
 23833 \_CS\_POSIX\_V6\_LPBIG\_OFFBIG\_LDFLAGS  
 23834 \_CS\_POSIX\_V6\_LPBIG\_OFFBIG\_LIBS  
 23835 \_CS\_POSIX\_V6\_WIDTH\_RESTRICTED\_ENVS  
 23836 \_CS\_V6\_ENV

23837 If *len* is not 0, and if *name* has a configuration-defined value, *confstr()* shall copy that value into  
 23838 the *len*-byte buffer pointed to by *buf*. If the string to be returned is longer than *len* bytes,  
 23839 including the terminating null, then *confstr()* shall truncate the string to *len*–1 bytes and null-  
 23840 terminate the result. The application can detect that the string was truncated by comparing the  
 23841 value returned by *confstr()* with *len*.

23842 If *len* is 0 and *buf* is a null pointer, then *confstr()* shall still return the integer value as defined  
 23843 below, but shall not return a string. If *len* is 0 but *buf* is not a null pointer, the result is

23844 unspecified.

23845 After a call to:

23846 `confstr(_CS_V7_ENV, buf, sizeof(buf))`

23847 the string stored in *buf* shall contain a <space>-separated list of the variable=value environment  
23848 variable pairs an implementation requires as part of specifying a conforming environment, as  
23849 described in the implementations' conformance documentation.

23850 If the implementation supports the POSIX shell option, the string stored in *buf* after a call to:

23851 `confstr(_CS_PATH, buf, sizeof(buf))`

23852 can be used as a value of the *PATH* environment variable that accesses all of the standard  
23853 utilities of POSIX.1-2008, that are provided in a manner accessible via the *exec* family of  
23854 functions, if the return value is less than or equal to *sizeof(buf)*.

## 23855 RETURN VALUE

23856 If *name* has a configuration-defined value, *confstr()* shall return the size of buffer that would be  
23857 needed to hold the entire configuration-defined value including the terminating null. If this  
23858 return value is greater than *len*, the string returned in *buf* is truncated.

23859 If *name* is invalid, *confstr()* shall return 0 and set *errno* to indicate the error.

23860 If *name* does not have a configuration-defined value, *confstr()* shall return 0 and leave *errno*  
23861 unchanged.

## 23862 ERRORS

23863 The *confstr()* function shall fail if:

23864 [EINVAL] The value of the *name* argument is invalid.

## 23865 EXAMPLES

23866 None.

## 23867 APPLICATION USAGE

23868 An application can distinguish between an invalid *name* parameter value and one that  
23869 corresponds to a configurable variable that has no configuration-defined value by checking if  
23870 *errno* is modified. This mirrors the behavior of *sysconf()*.

23871 The original need for this function was to provide a way of finding the configuration-defined  
23872 default value for the environment variable *PATH*. Since *PATH* can be modified by the user to  
23873 include directories that could contain utilities replacing the standard utilities in the Shell and  
23874 Utilities volume of POSIX.1-2008, applications need a way to determine the system-supplied  
23875 *PATH* environment variable value that contains the correct search path for the standard utilities.

23876 An application could use:

23877 `confstr(name, (char *)NULL, (size_t)0)`

23878 to find out how big a buffer is needed for the string value; use *malloc()* to allocate a buffer to  
23879 hold the string; and call *confstr()* again to get the string. Alternately, it could allocate a fixed,  
23880 static buffer that is big enough to hold most answers (perhaps 512 or 1024 bytes), but then use  
23881 *malloc()* to allocate a larger buffer if it finds that this is too small.

## 23882 RATIONALE

23883 Application developers can normally determine any configuration variable by means of reading  
23884 from the stream opened by a call to:

23885 `popen("command -p getconf variable", "r");`

The *confstr()* function with a *name* argument of `_CS_PATH` returns a string that can be used as a *PATH* environment variable setting that will reference the standard shell and utilities as described in the Shell and Utilities volume of POSIX.1-2008.

The *confstr()* function copies the returned string into a buffer supplied by the application instead of returning a pointer to a string. This allows a cleaner function in some implementations (such as those with lightweight threads) and resolves questions about when the application must copy the string returned.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*exec*, *fpathconf()*, *sysconf()*

XBD `<unistd.h>`

XCUI *c99*

#### CHANGE HISTORY

First released in Issue 4. Derived from the ISO POSIX-2 standard.

##### Issue 5

A table indicating the permissible values of *name* is added to the DESCRIPTION. All those marked EX are new in this version.

##### Issue 6

The Open Group Corrigendum U033/7 is applied. The return value for the case returning the size of the buffer now explicitly states that this includes the terminating null.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated with new arguments which can be used to determine configuration strings for C compiler flags, linker/loader flags, and libraries for each different supported programming environment. This is a change to support data size neutrality.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The DESCRIPTION is updated to include text describing how `_CS_PATH` can be used to obtain a *PATH* to access the standard utilities.

The macros associated with the *c89* programming models are marked LEGACY and new equivalent macros associated with *c99* are introduced.

##### Issue 7

Austin Group Interpretation 1003.1-2001 #047 is applied, adding the `_CS_V7_ENV` variable.

Austin Group Interpretations 1003.1-2001 #166 is applied to permit an additional compiler flag to enable threads.

The V6 variables for the supported programming environments are marked obsolescent.

The variables for the supported programming environments are updated to be V7.

The LEGACY variables and obsolescent values are removed.

23925 **NAME**

23926 conj, conjf, conjl — complex conjugate functions

23927 **SYNOPSIS**23928 `#include <complex.h>`23929 `double complex conj(double complex z);`23930 `float complex conjf(float complex z);`23931 `long double complex conjl(long double complex z);`23932 **DESCRIPTION**

23933 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 23934 conflict between the requirements described here and the ISO C standard is unintentional. This  
 23935 volume of POSIX.1-2008 defers to the ISO C standard.

23936 These functions shall compute the complex conjugate of *z*, by reversing the sign of its imaginary  
 23937 part.

23938 **RETURN VALUE**

23939 These functions return the complex conjugate value.

23940 **ERRORS**

23941 No errors are defined.

23942 **EXAMPLES**

23943 None.

23944 **APPLICATION USAGE**

23945 None.

23946 **RATIONALE**

23947 None.

23948 **FUTURE DIRECTIONS**

23949 None.

23950 **SEE ALSO**23951 *carg()*, *cimag()*, *cproj()*, *creal()*23952 XBD **<complex.h>**23953 **CHANGE HISTORY**

23954 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

## NAME

connect — connect a socket

## SYNOPSIS

```
#include <sys/socket.h>

int connect(int socket, const struct sockaddr *address,
            socklen_t address_len);
```

## DESCRIPTION

The *connect()* function shall attempt to make a connection on a connection-mode socket or to set or reset the peer address of a connectionless-mode socket. The function takes the following arguments:

<i>socket</i>	Specifies the file descriptor associated with the socket.
<i>address</i>	Points to a <b>sockaddr</b> structure containing the peer address. The length and format of the address depend on the address family of the socket.
<i>address_len</i>	Specifies the length of the <b>sockaddr</b> structure pointed to by the <i>address</i> argument.

If the socket has not already been bound to a local address, *connect()* shall bind it to an address which, unless the socket's address family is AF\_UNIX, is an unused local address.

If the initiating socket is not connection-mode, then *connect()* shall set the socket's peer address, and no connection is made. For SOCK\_DGRAM sockets, the peer address identifies where all datagrams are sent on subsequent *send()* functions, and limits the remote sender for subsequent *recv()* functions. If the *sa\_family* member of *address* is AF\_UNSPEC, the socket's peer address shall be reset. Note that despite no connection being made, the term "connected" is used to describe a connectionless-mode socket for which a peer address has been set.

If the initiating socket is connection-mode, then *connect()* shall attempt to establish a connection to the address specified by the *address* argument. If the connection cannot be established immediately and O\_NONBLOCK is not set for the file descriptor for the socket, *connect()* shall block for up to an unspecified timeout interval until the connection is established. If the timeout interval expires before the connection is established, *connect()* shall fail and the connection attempt shall be aborted. If *connect()* is interrupted by a signal that is caught while blocked waiting to establish a connection, *connect()* shall fail and set *errno* to [EINTR], but the connection request shall not be aborted, and the connection shall be established asynchronously.

If the connection cannot be established immediately and O\_NONBLOCK is set for the file descriptor for the socket, *connect()* shall fail and set *errno* to [EINPROGRESS], but the connection request shall not be aborted, and the connection shall be established asynchronously. Subsequent calls to *connect()* for the same socket, before the connection is established, shall fail and set *errno* to [EALREADY].

When the connection has been established asynchronously, *pselect()*, *select()*, and *poll()* shall indicate that the file descriptor for the socket is ready for writing.

The socket in use may require the process to have appropriate privileges to use the *connect()* function.

## RETURN VALUE

Upon successful completion, *connect()* shall return 0; otherwise, -1 shall be returned and *errno* set to indicate the error.



23998 **ERRORS**23999 The *connect()* function shall fail if:

24000 [EADDRNOTAVAIL]

24001 The specified address is not available from the local machine.

24002 [EAFNOSUPPORT]

24003 The specified address is not a valid address for the address family of the  
24004 specified socket.

24005 [EALREADY] A connection request is already in progress for the specified socket.

24006 [EBADF] The *socket* argument is not a valid file descriptor.

24007 [ECONNREFUSED]

24008 The target address was not listening for connections or refused the connection  
24009 request.24010 [EINPROGRESS] O\_NONBLOCK is set for the file descriptor for the socket and the connection  
24011 cannot be immediately established; the connection shall be established  
24012 asynchronously.24013 [EINTR] The attempt to establish a connection was interrupted by delivery of a signal  
24014 that was caught; the connection shall be established asynchronously.

24015 [EISCONN] The specified socket is connection-mode and is already connected.

24016 [ENETUNREACH]

24017 No route to the network is present.

24018 [ENOTSOCK] The *socket* argument does not refer to a socket.24019 [EPROTOTYPE] The specified address has a different type than the socket bound to the  
24020 specified peer address.

24021 [ETIMEDOUT] The attempt to connect timed out before a connection was made.

24022 If the address family of the socket is AF\_UNIX, then *connect()* shall fail if:

24023 [EIO] An I/O error occurred while reading from or writing to the file system.

24024 [ELOOP] A loop exists in symbolic links encountered during resolution of the pathname  
24025 in *address*.

24026 [ENAMETOOLONG]

24027 The length of a component of a pathname is longer than {NAME\_MAX}.

24028 [ENOENT] A component of the pathname does not name an existing file or the pathname  
24029 is an empty string.24030 [ENOTDIR] A component of the path prefix of the pathname in *address* names an existing  
24031 file that is neither a directory nor a symbolic link to a directory, or the  
24032 pathname in *address* contains at least one non-*<slash>* character and ends with  
24033 one or more trailing *<slash>* characters and the last pathname component  
24034 names an existing file that is neither a directory nor a symbolic link to a  
24035 directory.24036 The *connect()* function may fail if:24037 [EACCES] Search permission is denied for a component of the path prefix; or write access  
24038 to the named socket is denied.

24039	[EADDRINUSE]	Attempt to establish a connection that uses addresses that are already in use.
24040	[ECONNRESET]	Remote host reset the connection request.
24041	[EHOSTUNREACH]	
24042		The destination host cannot be reached (probably because the host is down or
24043		a remote router cannot reach it).
24044	[EINVAL]	The <i>address_len</i> argument is not a valid length for the address family; or
24045		invalid address family in the <b>sockaddr</b> structure.
24046	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during
24047		resolution of the pathname in <i>address</i> .
24048	[ENAMETOOLONG]	
24049		The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a
24050		symbolic link produced an intermediate result with a length that exceeds
24051		{PATH_MAX}.
24052	[ENETDOWN]	The local network interface used to reach the destination is down.
24053	[ENOBUFS]	No buffer space is available.
24054	[EOPNOTSUPP]	The socket is listening and cannot be connected.

#### 24055 EXAMPLES

24056 None.

#### 24057 APPLICATION USAGE

24058 If *connect()* fails, the state of the socket is unspecified. Conforming applications should close the  
24059 file descriptor and create a new socket before attempting to reconnect.

#### 24060 RATIONALE

24061 None.

#### 24062 FUTURE DIRECTIONS

24063 None.

#### 24064 SEE ALSO

24065 *accept()*, *bind()*, *close()*, *getsockname()*, *poll()*, *pselect()*, *send()*, *shutdown()*, *socket()*

24066 XBD <sys/socket.h>

#### 24067 CHANGE HISTORY

24068 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

24069 The wording of the mandatory [ELOOP] error condition is updated, and a second optional  
24070 [ELOOP] error condition is added.

#### 24071 Issue 7

24072 Austin Group Interpretation 1003.1-2001 #035 is applied, clarifying the description of connected  
24073 sockets.

24074 Austin Group Interpretation 1003.1-2001 #143 is applied.

24075 Austin Group Interpretation 1003.1-2001 #188 is applied, changing the method used to reset a  
24076 peer address for a datagram socket.

24077 SD5-XSH-ERN-185 is applied.

24078 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0062 [324] is applied.

24079 **NAME**

24080 copysign, copysignf, copysignl — number manipulation function

24081 **SYNOPSIS**

24082 #include &lt;math.h&gt;

24083 double copysign(double x, double y);

24084 float copysignf(float x, float y);

24085 long double copysignl(long double x, long double y);

24086 **DESCRIPTION**

24087 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 24088 conflict between the requirements described here and the ISO C standard is unintentional. This  
 24089 volume of POSIX.1-2008 defers to the ISO C standard.

24090 These functions shall produce a value with the magnitude of *x* and the sign of *y*. On  
 24091 implementations that represent a signed zero but do not treat negative zero consistently in  
 24092 arithmetic operations, these functions regard the sign of zero as positive.

24093 **RETURN VALUE**

24094 Upon successful completion, these functions shall return a value with the magnitude of *x* and  
 24095 the sign of *y*.

24096 **ERRORS**

24097 No errors are defined.

24098 **EXAMPLES**

24099 None.

24100 **APPLICATION USAGE**

24101 None.

24102 **RATIONALE**

24103 None.

24104 **FUTURE DIRECTIONS**

24105 None.

24106 **SEE ALSO**24107 [signbit\(\)](#)24108 XBD [<math.h>](#)24109 **CHANGE HISTORY**

24110 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

24111 **NAME**

24112 cos, cosf, cosl — cosine function

24113 **SYNOPSIS**

```
24114 #include <math.h>
24115 double cos(double x);
24116 float cosf(float x);
24117 long double cosl(long double x);
```

24118 **DESCRIPTION**

24119 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 24120 conflict between the requirements described here and the ISO C standard is unintentional. This  
 24121 volume of POSIX.1-2008 defers to the ISO C standard.

24122 These functions shall compute the cosine of their argument  $x$ , measured in radians.

24123 An application wishing to check for error situations should set *errno* to zero and call  
 24124 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 24125 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 24126 zero, an error has occurred.

24127 **RETURN VALUE**

24128 Upon successful completion, these functions shall return the cosine of  $x$ .

24129 MX If  $x$  is NaN, a NaN shall be returned.

24130 If  $x$  is  $\pm 0$ , the value 1.0 shall be returned.

24131 If  $x$  is  $\pm \text{Inf}$ , a domain error shall occur, and a NaN shall be returned.

24132 **ERRORS**

24133 These functions shall fail if:

24134 MX **Domain Error** The  $x$  argument is  $\pm \text{Inf}$ .

24135 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 24136 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 24137 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 24138 shall be raised.

24139 **EXAMPLES**24140 **Taking the Cosine of a 45-Degree Angle**

```
24141 #include <math.h>
24142 ...
24143 double radians = 45 * M_PI / 180;
24144 double result;
24145 ...
24146 result = cos(radians);
```

24147 **APPLICATION USAGE**

24148 These functions may lose accuracy when their argument is near an odd multiple of  $\pi/2$  or is far  
 24149 from 0.

24150 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 24151 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

24152 **RATIONALE**

24153 None.

24154 **FUTURE DIRECTIONS**

24155 None.

24156 **SEE ALSO**24157 *acos()*, *feclearexcept()*, *fetestexcept()*, *isnan()*, *sin()*, *tan()*24158 XBD Section 4.20 (on page 117), **<math.h>**24159 **CHANGE HISTORY**

24160 First released in Issue 1. Derived from Issue 1 of the SVID.

24161 **Issue 5**24162 The DESCRIPTION is updated to indicate how an application should check for an error. This  
24163 text was previously published in the APPLICATION USAGE section.24164 **Issue 6**24165 The *cosf()* and *cosl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.24166 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
24167 revised to align with the ISO/IEC 9899:1999 standard.24168 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
24169 marked.24170 **Issue 7**

24171 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0063 [320] is applied.

24172 **NAME**

24173 cosh, coshf, coshl — hyperbolic cosine functions

24174 **SYNOPSIS**

```
24175 #include <math.h>
24176 double cosh(double x);
24177 float coshf(float x);
24178 long double coshl(long double x);
```

24179 **DESCRIPTION**

24180 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 24181 conflict between the requirements described here and the ISO C standard is unintentional. This  
 24182 volume of POSIX.1-2008 defers to the ISO C standard.

24183 These functions shall compute the hyperbolic cosine of their argument  $x$ .

24184 An application wishing to check for error situations should set *errno* to zero and call  
 24185 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 24186 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 24187 zero, an error has occurred.

24188 **RETURN VALUE**

24189 Upon successful completion, these functions shall return the hyperbolic cosine of  $x$ .

24190 If the correct value would cause overflow, a range error shall occur and *cosh()*, *coshf()*, and  
 24191 *coshl()* shall return the value of the macro HUGE\_VAL, HUGE\_VALF, and HUGE\_VALL,  
 24192 respectively.

24193 MX If  $x$  is NaN, a NaN shall be returned.

24194 If  $x$  is  $\pm 0$ , the value 1.0 shall be returned.

24195 If  $x$  is  $\pm \text{Inf}$ ,  $+\text{Inf}$  shall be returned.

24196 **ERRORS**

24197 These functions shall fail if:

24198 Range Error The result would cause an overflow.

24199 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 24200 then *errno* shall be set to [ERANGE]. If the integer expression  
 24201 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the overflow  
 24202 floating-point exception shall be raised.

24203 **EXAMPLES**

24204 None.

24205 **APPLICATION USAGE**

24206 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 24207 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero. -

24208 **RATIONALE**

24209 None.

24210 **FUTURE DIRECTIONS**

24211 None.

24212 **SEE ALSO**24213 *acosh()*, *feclearexcept()*, *fetetestexcept()*, *isnan()*, *sinh()*, *tanh()*24214 XBD Section 4.20 (on page 117), **<math.h>**24215 **CHANGE HISTORY**

24216 First released in Issue 1. Derived from Issue 1 of the SVID.

24217 **Issue 5**24218 The DESCRIPTION is updated to indicate how an application should check for an error. This  
24219 text was previously published in the APPLICATION USAGE section.24220 **Issue 6**24221 The *coshf()* and *coshl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.24222 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
24223 revised to align with the ISO/IEC 9899:1999 standard.24224 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
24225 marked.

**24226 NAME**

24227 cosl — cosine function

**24228 SYNOPSIS**

24229 #include <math.h>

24230 long double cosl(long double x);

**24231 DESCRIPTION**

24232 Refer to *cos()*.



24233 **NAME**

24234 cpow, cpowf, cpowl — complex power functions

24235 **SYNOPSIS**

```
24236 #include <complex.h>
24237 double complex cpow(double complex x, double complex y);
24238 float complex cpowf(float complex x, float complex y);
24239 long double complex cpowl(long double complex x,
24240 long double complex y);
```

24241 **DESCRIPTION**

24242 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 24243 conflict between the requirements described here and the ISO C standard is unintentional. This  
 24244 volume of POSIX.1-2008 defers to the ISO C standard.

24245 These functions shall compute the complex power function  $x^y$ , with a branch cut for the first  
 24246 parameter along the negative real axis.

24247 **RETURN VALUE**

24248 These functions shall return the complex power function value.

24249 **ERRORS**

24250 No errors are defined.

24251 **EXAMPLES**

24252 None.

24253 **APPLICATION USAGE**

24254 None.

24255 **RATIONALE**

24256 None.

24257 **FUTURE DIRECTIONS**

24258 None.

24259 **SEE ALSO**24260 *cabs()*, *csqrt()*24261 XBD **<complex.h>**24262 **CHANGE HISTORY**

24263 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

24264 **NAME**

24265 cproj, cprojf, cprojl — complex projection functions

24266 **SYNOPSIS**

```
24267 #include <complex.h>
24268 double complex cproj(double complex z);
24269 float complex cprojf(float complex z);
24270 long double complex cprojl(long double complex z);
```

24271 **DESCRIPTION**

24272 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
24273 conflict between the requirements described here and the ISO C standard is unintentional. This  
24274 volume of POSIX.1-2008 defers to the ISO C standard.

24275 These functions shall compute a projection of *z* onto the Riemann sphere: *z* projects to *z*, except  
24276 that all complex infinities (even those with one infinite part and one NaN part) project to  
24277 positive infinity on the real axis. If *z* has an infinite part, then *cproj(z)* shall be equivalent to:

```
24278 INFINITY + I * copysign(0.0, cimag(z))
```

24279 **RETURN VALUE**

24280 These functions shall return the value of the projection onto the Riemann sphere.

24281 **ERRORS**

24282 No errors are defined.

24283 **EXAMPLES**

24284 None.

24285 **APPLICATION USAGE**

24286 None.

24287 **RATIONALE**

24288 Two topologies are commonly used in complex mathematics: the complex plane with its  
24289 continuum of infinities, and the Riemann sphere with its single infinity. The complex plane is  
24290 better suited for transcendental functions, the Riemann sphere for algebraic functions. The  
24291 complex types with their multiplicity of infinities provide a useful (though imperfect) model for  
24292 the complex plane. The *cproj()* function helps model the Riemann sphere by mapping all  
24293 infinities to one, and should be used just before any operation, especially comparisons, that  
24294 might give spurious results for any of the other infinities. Note that a complex value with one  
24295 infinite part and one NaN part is regarded as an infinity, not a NaN, because if one part is  
24296 infinite, the complex value is infinite independent of the value of the other part. For the same  
24297 reason, *cabs()* returns an infinity if its argument has an infinite part and a NaN part.

24298 **FUTURE DIRECTIONS**

24299 None.

24300 **SEE ALSO**24301 *carg()*, *cimag()*, *conj()*, *creal()*24302 XBD **<complex.h>**24303 **CHANGE HISTORY**

24304 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

24305 **NAME**

24306 creal, crealf, creall — complex real functions

24307 **SYNOPSIS**

```
24308     #include <complex.h>
24309     double creal(double complex z);
24310     float crealf(float complex z);
24311     long double creall(long double complex z);
```

24312 **DESCRIPTION**

24313 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 24314 conflict between the requirements described here and the ISO C standard is unintentional. This  
 24315 volume of POSIX.1-2008 defers to the ISO C standard.

24316 These functions shall compute the real part of *z*.24317 **RETURN VALUE**

24318 These functions shall return the real part value.

24319 **ERRORS**

24320 No errors are defined.

24321 **EXAMPLES**

24322 None.

24323 **APPLICATION USAGE**24324 For a variable *z* of type **complex**:24325 `z == creal(z) + cimag(z)*I`24326 **RATIONALE**

24327 None.

24328 **FUTURE DIRECTIONS**

24329 None.

24330 **SEE ALSO**24331 *carg()*, *cimag()*, *conj()*, *cproj()*24332 XBD **<complex.h>**24333 **CHANGE HISTORY**

24334 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

24335 **NAME**

24336 creat — create a new file or rewrite an existing one

24337 **SYNOPSIS**24338 OH `#include <sys/stat.h>`24339 `#include <fcntl.h>`24340 `int creat(const char *path, mode_t mode);`24341 **DESCRIPTION**24342 The *creat()* function shall behave as if it is implemented as follows:24343 `int creat(const char *path, mode_t mode)`24344 `{`24345  `return open(path, O_WRONLY|O_CREAT|O_TRUNC, mode);`24346 `}`24347 **RETURN VALUE**24348 Refer to *open()*.24349 **ERRORS**24350 Refer to *open()*.24351 **EXAMPLES**24352 **Creating a File**24353 The following example creates the file */tmp/file* with read and write permissions for the file  
24354 owner and read permission for group and others. The resulting file descriptor is assigned to the  
24355 *fd* variable.24356 `#include <fcntl.h>`24357 `...`24358 `int fd;`24359 `mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;`24360 `char *pathname = "/tmp/file";`24361 `...`24362 `fd = creat(pathname, mode);`24363 `...`24364 **APPLICATION USAGE**

24365 None.

24366 **RATIONALE**24367 The *creat()* function is redundant. Its services are also provided by the *open()* function. It has  
24368 been included primarily for historical purposes since many existing applications depend on it. It  
24369 is best considered a part of the C binding rather than a function that should be provided in other  
24370 languages.24371 **FUTURE DIRECTIONS**

24372 None.

24373 **SEE ALSO**24374 *mknod()*, *open()*24375 XBD *<fcntl.h>*, *<sys/stat.h>*, *<sys/types.h>*

24376 **CHANGE HISTORY**

24377 First released in Issue 1. Derived from Issue 1 of the SVID.

24378 **Issue 6**24379 In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.24380 The following new requirements on POSIX implementations derive from alignment with the  
24381 Single UNIX Specification:

- 24382
- The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was  
24383 required for conforming implementations of previous POSIX specifications, it was not  
24384 required for UNIX applications.

24385 **Issue 7**

24386 SD5-XSH-ERN-186 is applied.

24387 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0064 [291] is applied.

24388 **NAME**24389 crypt — string encoding function (**CRYPT**)24390 **SYNOPSIS**

```
24391 XSI    #include <unistd.h>
24392        char *crypt(const char *key, const char *salt);
```

24393 **DESCRIPTION**24394 The *crypt()* function is a string encoding function. The algorithm is implementation-defined.24395 The *key* argument points to a string to be encoded. The *salt* argument shall be a string of at least  
24396 two bytes in length not including the null character chosen from the set:

```
24397 a b c d e f g h i j k l m n o p q r s t u v w x y z
24398 A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
24399 0 1 2 3 4 5 6 7 8 9 . /
```

24400 The first two bytes of this string may be used to perturb the encoding algorithm.

24401 The return value of *crypt()* points to static data that is overwritten by each call.24402 The *crypt()* function need not be thread-safe.24403 **RETURN VALUE**

24404 Upon successful completion, *crypt()* shall return a pointer to the encoded string. The first two  
24405 bytes of the returned value shall be those of the *salt* argument. Otherwise, it shall return a null  
24406 pointer and set *errno* to indicate the error.

24407 **ERRORS**24408 The *crypt()* function shall fail if:

24409 [ENOSYS] The functionality is not supported on this implementation.

24410 **EXAMPLES**24411 **Encoding Passwords**

24412 The following example finds a user database entry matching a particular user name and changes  
24413 the current password to a new password. The *crypt()* function generates an encoded version of  
24414 each password. The first call to *crypt()* produces an encoded version of the old password; that  
24415 encoded password is then compared to the password stored in the user database. The second  
24416 call to *crypt()* encodes the new password before it is stored.

24417 The *putpwent()* function, used in the following example, is not part of POSIX.1-2008.

```
24418 #include <unistd.h>
24419 #include <pwd.h>
24420 #include <string.h>
24421 #include <stdio.h>
24422 ...
24423 int valid_change;
24424 int pfd; /* Integer for file descriptor returned by open(). */
24425 FILE *fpfd; /* File pointer for use in putpwent(). */
24426 struct passwd *p;
24427 char user[100];
24428 char oldpasswd[100];
24429 char newpasswd[100];
24430 char savepasswd[100];
```

```

24431     ...
24432     valid_change = 0;
24433     while ((p = getpwent()) != NULL) {
24434         /* Change entry if found. */
24435         if (strcmp(p->pw_name, user) == 0) {
24436             if (strcmp(p->pw_passwd, crypt(oldpasswd, p->pw_passwd)) == 0) {
24437                 strcpy(savepasswd, crypt(newpasswd, user));
24438                 p->pw_passwd = savepasswd;
24439                 valid_change = 1;
24440             }
24441             else {
24442                 fprintf(stderr, "Old password is not valid\n");
24443             }
24444         }
24445         /* Put passwd entry into ptmp. */
24446         putpwent(p, fpfd);
24447     }

```

#### 24448 APPLICATION USAGE

24449 The values returned by this function need not be portable among XSI-conformant systems.

24450 Several implementations offer extensions via characters outside of the set specified for the *salt* +  
 24451 argument for specifying alternative algorithms; while not portable, these extensions may offer +  
 24452 better security. The use of *crypt()* for anything other than password hashing is not +  
 24453 recommended.

#### 24454 RATIONALE

24455 None.

#### 24456 FUTURE DIRECTIONS

24457 None.

#### 24458 SEE ALSO

24459 *encrypt()*, *setkey()*

24460 XBD <unistd.h>

#### 24461 CHANGE HISTORY

24462 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 24463 Issue 5

24464 Normative text previously in the APPLICATION USAGE section is moved to the  
 24465 DESCRIPTION.

#### 24466 Issue 7

24467 Austin Group Interpretation 1003.1-2001 #156 is applied.

24468 SD5-XSH-ERN-178 is applied.

24469 **NAME**

24470 csin, csinf, csinl — complex sine functions

24471 **SYNOPSIS**

24472 #include &lt;complex.h&gt;

24473 double complex csin(double complex z);

24474 float complex csinf(float complex z);

24475 long double complex csinl(long double complex z);

24476 **DESCRIPTION**

24477 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
24478 conflict between the requirements described here and the ISO C standard is unintentional. This  
24479 volume of POSIX.1-2008 defers to the ISO C standard.

24480 These functions shall compute the complex sine of z.

24481 **RETURN VALUE**

24482 These functions shall return the complex sine value.

24483 **ERRORS**

24484 No errors are defined.

24485 **EXAMPLES**

24486 None.

24487 **APPLICATION USAGE**

24488 None.

24489 **RATIONALE**

24490 None.

24491 **FUTURE DIRECTIONS**

24492 None.

24493 **SEE ALSO**24494 [casin\(\)](#)24495 XBD [<complex.h>](#)24496 **CHANGE HISTORY**

24497 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.



24498 **NAME**

24499       csinh, csinhf, csinhl — complex hyperbolic sine functions

24500 **SYNOPSIS**

```
24501       #include <complex.h>
24502       double complex csinh(double complex z);
24503       float complex csinhf(float complex z);
24504       long double complex csinhl(long double complex z);
```

24505 **DESCRIPTION**

24506 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
24507       conflict between the requirements described here and the ISO C standard is unintentional. This  
24508       volume of POSIX.1-2008 defers to the ISO C standard.

24509       These functions shall compute the complex hyperbolic sine of *z*.24510 **RETURN VALUE**

24511       These functions shall return the complex hyperbolic sine value.

24512 **ERRORS**

24513       No errors are defined.

24514 **EXAMPLES**

24515       None.

24516 **APPLICATION USAGE**

24517       None.

24518 **RATIONALE**

24519       None.

24520 **FUTURE DIRECTIONS**

24521       None.

24522 **SEE ALSO**24523       *casinh()*24524       XBD *<complex.h>*24525 **CHANGE HISTORY**

24526       First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

24527 **NAME**

24528       csinl — complex sine functions

24529 **SYNOPSIS**

24530       #include &lt;complex.h&gt;

24531       long double complex csinl(long double complex z);

24532 **DESCRIPTION**24533       Refer to *csin()*.

24534 **NAME**

24535 csqrt, csqrtf, csqrtl — complex square root functions

24536 **SYNOPSIS**

24537 #include &lt;complex.h&gt;

24538 double complex csqrt(double complex z);

24539 float complex csqrtf(float complex z);

24540 long double complex csqrtl(long double complex z);

24541 **DESCRIPTION**

24542 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 24543 conflict between the requirements described here and the ISO C standard is unintentional. This  
 24544 volume of POSIX.1-2008 defers to the ISO C standard.

24545 These functions shall compute the complex square root of *z*, with a branch cut along the negative  
 24546 real axis.

24547 **RETURN VALUE**

24548 These functions shall return the complex square root value, in the range of the right half-plane  
 24549 (including the imaginary axis).

24550 **ERRORS**

24551 No errors are defined.

24552 **EXAMPLES**

24553 None.

24554 **APPLICATION USAGE**

24555 None.

24556 **RATIONALE**

24557 None.

24558 **FUTURE DIRECTIONS**

24559 None.

24560 **SEE ALSO**24561 *cabs()*, *cpow()*

24562 XBD &lt;complex.h&gt;

24563 **CHANGE HISTORY**

24564 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

24565 **NAME**

24566       ctan, ctanf, ctanl — complex tangent functions

24567 **SYNOPSIS**

24568       #include &lt;complex.h&gt;

24569       double complex ctan(double complex z);

24570       float complex ctanf(float complex z);

24571       long double complex ctanl(long double complex z);

24572 **DESCRIPTION**

24573 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
24574       conflict between the requirements described here and the ISO C standard is unintentional. This  
24575       volume of POSIX.1-2008 defers to the ISO C standard.

24576       These functions shall compute the complex tangent of z.

24577 **RETURN VALUE**

24578       These functions shall return the complex tangent value.

24579 **ERRORS**

24580       No errors are defined.

24581 **EXAMPLES**

24582       None.

24583 **APPLICATION USAGE**

24584       None.

24585 **RATIONALE**

24586       None.

24587 **FUTURE DIRECTIONS**

24588       None.

24589 **SEE ALSO**

24590       catan()

24591       XBD &lt;complex.h&gt;

24592 **CHANGE HISTORY**

24593       First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

24594 **NAME**

24595 ctanh, ctanhf, ctanhl — complex hyperbolic tangent functions

24596 **SYNOPSIS**

```
24597 #include <complex.h>
24598 double complex ctanh(double complex z);
24599 float complex ctanhf(float complex z);
24600 long double complex ctanhl(long double complex z);
```

24601 **DESCRIPTION**

24602 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
24603 conflict between the requirements described here and the ISO C standard is unintentional. This  
24604 volume of POSIX.1-2008 defers to the ISO C standard.

24605 These functions shall compute the complex hyperbolic tangent of  $z$ .24606 **RETURN VALUE**

24607 These functions shall return the complex hyperbolic tangent value.

24608 **ERRORS**

24609 No errors are defined.

24610 **EXAMPLES**

24611 None.

24612 **APPLICATION USAGE**

24613 None.

24614 **RATIONALE**

24615 None.

24616 **FUTURE DIRECTIONS**

24617 None.

24618 **SEE ALSO**24619 [catanh\(\)](#)24620 XBD [<complex.h>](#)24621 **CHANGE HISTORY**

24622 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

24623 **NAME**

24624       ctanl — complex tangent functions

24625 **SYNOPSIS**

24626       #include &lt;complex.h&gt;

24627       long double complex ctanl(long double complex z);

24628 **DESCRIPTION**24629       Refer to *ctan()*.

24630 **NAME**

24631 ctermid — generate a pathname for the controlling terminal

24632 **SYNOPSIS**

```
24633 CX      #include <stdio.h>
24634      char *ctermid(char *s);
```

24635 **DESCRIPTION**

24636 The *ctermid()* function shall generate a string that, when used as a pathname, refers to the  
 24637 current controlling terminal for the current process. If *ctermid()* returns a pathname, access to the  
 24638 file is not guaranteed.

24639 The *ctermid()* function need not be thread-safe if called with a NULL parameter.

24640 **RETURN VALUE**

24641 If *s* is a null pointer, the string shall be generated in an area that may be static, the address of  
 24642 which shall be returned. The application shall not modify the string returned. The returned  
 24643 pointer might be invalidated or the string content might be overwritten by a subsequent call to  
 24644 *ctermid()*. The returned point might also be invalidated if the calling thread is terminated. If *s* is +  
 24645 not a null pointer, *s* is assumed to point to a character array of at least L\_ctermid bytes; the string  
 24646 is placed in this array and the value of *s* shall be returned. The symbolic constant L\_ctermid is  
 24647 defined in <stdio.h>, and shall have a value greater than 0.

24648 The *ctermid()* function shall return an empty string if the pathname that would refer to the  
 24649 controlling terminal cannot be determined, or if the function is unsuccessful.

24650 **ERRORS**

24651 No errors are defined.

24652 **EXAMPLES**24653 **Determining the Controlling Terminal for the Current Process**

24654 The following example returns a pointer to a string that identifies the controlling terminal for the  
 24655 current process. The pathname for the terminal is stored in the array pointed to by the *ptr*  
 24656 argument, which has a size of L\_ctermid bytes, as indicated by the *term* argument.

```
24657 #include <stdio.h>
24658 ...
24659 char term[L_ctermid];
24660 char *ptr;
24661 ptr = ctermid(term);
```

24662 **APPLICATION USAGE**

24663 The difference between *ctermid()* and *ttyname()* is that *ttyname()* must be handed a file  
 24664 descriptor and return a path of the terminal associated with that file descriptor, while *ctermid()*  
 24665 returns a string (such as "/dev/tty") that refers to the current controlling terminal if used as a  
 24666 pathname.

24667 **RATIONALE**

24668 L\_ctermid must be defined appropriately for a given implementation and must be greater than  
 24669 zero so that array declarations using it are accepted by the compiler. The value includes the  
 24670 terminating null byte.

24671 Conforming applications that use multiple threads cannot call *ctermid()* with NULL as the  
 24672 parameter. If *s* is not NULL, the *ctermid()* function generates a string that, when used as a

24673 pathname, refers to the current controlling terminal for the current process. If *s* is NULL, the  
24674 return value of *ctermid()* is undefined.

24675 There is no additional burden on the programmer—changing to use a hypothetical thread-safe  
24676 version of *ctermid()* along with allocating a buffer is more of a burden than merely allocating a  
24677 buffer. Application code should not assume that the returned string is short, as some  
24678 implementations have more than two pathname components before reaching a logical device  
24679 name.

#### 24680 FUTURE DIRECTIONS

24681 None.

#### 24682 SEE ALSO

24683 *ttynname()*

24684 XBD <stdio.h>

#### 24685 CHANGE HISTORY

24686 First released in Issue 1. Derived from Issue 1 of the SVID.

##### 24687 Issue 5

24688 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

##### 24689 Issue 6

24690 The normative text is updated to avoid use of the term “must” for application requirements.

##### 24691 Issue 7

24692 Austin Group Interpretation 1003.1-2001 #148 is applied, updating the RATIONALE.

24693 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0065 [75,428] is applied.



24694 **NAME**

24695 ctime, ctime\_r — convert a time value to a date and time string

24696 **SYNOPSIS**

```
24697 OB      #include <time.h>
24698          char *ctime(const time_t *clock);
24699 OB CX    char *ctime_r(const time_t *clock, char *buf);
```

24700 **DESCRIPTION**

24701 CX For *ctime()*: The functionality described on this reference page is aligned with the ISO C  
 24702 standard. Any conflict between the requirements described here and the ISO C standard is  
 24703 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

24704 The *ctime()* function shall convert the time pointed to by *clock*, representing time in seconds  
 24705 since the Epoch, to local time in the form of a string. It shall be equivalent to:

```
24706 asctime(localtime(clock))
```

24707 CX The *asctime()*, *ctime()*, *gmtime()*, and *localtime()* functions shall return values in one of two static  
 24708 objects: a broken-down time structure and an array of **char**. Execution of any of the functions  
 24709 may overwrite the information returned in either of these objects by any of the other functions.

24710 The *ctime()* function need not be thread-safe.

24711 The *ctime\_r()* function shall convert the calendar time pointed to by *clock* to local time in exactly  
 24712 the same form as *ctime()* and put the string into the array pointed to by *buf* (which shall be at  
 24713 least 26 bytes in size) and return *buf*.

24714 If *ctime\_r()* sets *tzname*, it shall also set *daylight* and *timezone*. If *ctime\_r()* does not set *tzname*, it  
 24715 shall not set *daylight* and shall not set *timezone*.

24716 **RETURN VALUE**

24717 The *ctime()* function shall return the pointer returned by *asctime()* with that broken-down time  
 24718 as an argument.

24719 CX Upon successful completion, *ctime\_r()* shall return a pointer to the string pointed to by *buf*.  
 24720 When an error is encountered, a null pointer shall be returned.

24721 **ERRORS**

24722 No errors are defined.

24723 **EXAMPLES**

24724 None.

24725 **APPLICATION USAGE**

24726 These functions are included only for compatibility with older implementations. They have  
 24727 undefined behavior if the resulting string would be too long, so the use of these functions  
 24728 should be discouraged. On implementations that do not detect output string length overflow, it  
 24729 is possible to overflow the output buffers in such a way as to cause applications to fail, or  
 24730 possible system security violations. Also, these functions do not support localized date and time  
 24731 formats. To avoid these problems, applications should use *strftime()* to generate strings from  
 24732 broken-down times.

24733 Values for the broken-down time structure can be obtained by calling *gmtime()* or *localtime()*.

24734 The *ctime\_r()* function is thread-safe and shall return values in a user-supplied buffer instead of  
 24735 possibly using a static data area that may be overwritten by each call.

24736 Attempts to use *ctime()* or *ctime\_r()* for times before the Epoch or for times beyond the year 9999

24737 produce undefined results. Refer to *asctime()* (on page 599).

#### 24738 RATIONALE

24739 The standard developers decided to mark the *ctime()* and *ctime\_r()* functions obsolescent even  
24740 though they are in the ISO C standard due to the possibility of buffer overflow. The ISO C  
24741 standard also provides the *strftime()* function which can be used to avoid these problems.

#### 24742 FUTURE DIRECTIONS

24743 These functions may be removed in a future version.

#### 24744 SEE ALSO

24745 *asctime()*, *clock()*, *difftime()*, *gmtime()*, *localtime()*, *mktime()*, *strftime()*, *strptime()*, *time()*, *utime()*

24746 XBD <time.h>

#### 24747 CHANGE HISTORY

24748 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 24749 Issue 5

24750 Normative text previously in the APPLICATION USAGE section is moved to the  
24751 DESCRIPTION.

24752 The *ctime\_r()* function is included for alignment with the POSIX Threads Extension.

24753 A note indicating that the *ctime()* function need not be reentrant is added to the DESCRIPTION.

#### 24754 Issue 6

24755 Extensions beyond the ISO C standard are marked.

24756 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

24757 The APPLICATION USAGE section is updated to include a note on the thread-safe function and  
24758 its avoidance of possibly using a static data area.

#### 24759 Issue 7

24760 Austin Group Interpretation 1003.1-2001 #156 is applied.

24761 SD5-XSH-ERN-25 is applied, updating the APPLICATION USAGE.

24762 Austin Group Interpretation 1003.1-2001 #053 is applied, marking these functions obsolescent.

24763 The *ctime\_r()* function is moved from the Thread-Safe Functions option to the Base.

24764 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0066 [321,428] is applied.

24765 **NAME**

24766 daylight — daylight savings time flag

24767 **SYNOPSIS**

```
24768 XSI      #include <time.h>  
24769      extern int daylight;
```

24770 **DESCRIPTION**24771 Refer to *tzset()*.

## NAME

dbm\_clearerr, dbm\_close, dbm\_delete, dbm\_error, dbm\_fetch, dbm\_firstkey, dbm\_nextkey, dbm\_open, dbm\_store — database functions

## SYNOPSIS

```
XSI
#include <ndbm.h>

int dbm_clearerr(DBM *db);
void dbm_close(DBM *db);
int dbm_delete(DBM *db, datum key);
int dbm_error(DBM *db);
datum dbm_fetch(DBM *db, datum key);
datum dbm_firstkey(DBM *db);
datum dbm_nextkey(DBM *db);
DBM *dbm_open(const char *file, int open_flags, mode_t file_mode);
int dbm_store(DBM *db, datum key, datum content, int store_mode);
```

## DESCRIPTION

These functions create, access, and modify a database.

A **datum** consists of at least two members, *dptr* and *dsize*. The *dptr* member points to an object that is *dsize* bytes in length. Arbitrary binary data, as well as character strings, may be stored in the object pointed to by *dptr*.

A database shall be stored in one or two files. When one file is used, the name of the database file shall be formed by appending the suffix **.db** to the *file* argument given to *dbm\_open()*. When two files are used, the names of the database files shall be formed by appending the suffixes **.dir** and **.pag** respectively to the *file* argument.

The *dbm\_open()* function shall open a database. The *file* argument to the function is the pathname of the database. The *open\_flags* argument has the same meaning as the *flags* argument of *open()* except that a database opened for write-only access opens the files for read and write access and the behavior of the **O\_APPEND** flag is unspecified. The *file\_mode* argument has the same meaning as the third argument of *open()*.

The *dbm\_open()* function need not accept pathnames longer than {PATH\_MAX}–4 bytes (including the terminating null), or pathnames with a last component longer than {NAME\_MAX}–4 bytes (excluding the terminating null).

The *dbm\_close()* function shall close a database. The application shall ensure that argument *db* is a pointer to a **dbm** structure that has been returned from a call to *dbm\_open()*.

These database functions shall support an internal block size large enough to support key/content pairs of at least 1 023 bytes.

The *dbm\_fetch()* function shall read a record from a database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm\_open()*. The argument *key* is a **datum** that has been initialized by the application to the value of the key that matches the key of the record the program is fetching.

The *dbm\_store()* function shall write a record to a database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm\_open()*. The argument *key* is a **datum** that has been initialized by the application to the value of the key that identifies (for subsequent reading, writing, or deleting) the record the application is writing. The argument *content* is a **datum** that has been initialized by the application to the value of the record the program is writing. The argument *store\_mode* controls whether *dbm\_store()* replaces any pre-existing record that has the same key that is specified by the *key* argument. The application shall

set *store\_mode* to either DBM\_INSERT or DBM\_REPLACE. If the database contains a record that matches the *key* argument and *store\_mode* is DBM\_REPLACE, the existing record shall be replaced with the new record. If the database contains a record that matches the *key* argument and *store\_mode* is DBM\_INSERT, the existing record shall be left unchanged and the new record ignored. If the database does not contain a record that matches the *key* argument and *store\_mode* is either DBM\_INSERT or DBM\_REPLACE, the new record shall be inserted in the database.

If the sum of a key/content pair exceeds the internal block size, the result is unspecified. Moreover, the application shall ensure that all key/content pairs that hash together fit on a single block. The *dbm\_store()* function shall return an error in the event that a disk block fills with inseparable data.

The *dbm\_delete()* function shall delete a record and its key from the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm\_open()*. The argument *key* is a **datum** that has been initialized by the application to the value of the key that identifies the record the program is deleting.

The *dbm\_firstkey()* function shall return the first key in the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm\_open()*.

The *dbm\_nextkey()* function shall return the next key in the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm\_open()*. The application shall ensure that the *dbm\_firstkey()* function is called before calling *dbm\_nextkey()*. Subsequent calls to *dbm\_nextkey()* return the next key until all of the keys in the database have been returned.

The *dbm\_error()* function shall return the error condition of the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm\_open()*.

The *dbm\_clearerr()* function shall clear the error condition of the database. The argument *db* is a pointer to a database structure that has been returned from a call to *dbm\_open()*.

The *dptr* pointers returned by these functions may point into static storage that may be changed by subsequent calls.

These functions need not be thread-safe.

## RETURN VALUE

The *dbm\_store()* and *dbm\_delete()* functions shall return 0 when they succeed and a negative value when they fail.

The *dbm\_store()* function shall return 1 if it is called with a *flags* value of DBM\_INSERT and the function finds an existing record with the same key.

The *dbm\_error()* function shall return 0 if the error condition is not set and return a non-zero value if the error condition is set.

The return value of *dbm\_clearerr()* is unspecified.

The *dbm\_firstkey()* and *dbm\_nextkey()* functions shall return a key **datum**. When the end of the database is reached, the *dptr* member of the key is a null pointer. If an error is detected, the *dptr* member of the key shall be a null pointer and the error condition of the database shall be set.

The *dbm\_fetch()* function shall return a content **datum**. If no record in the database matches the key or if an error condition has been detected in the database, the *dptr* member of the content shall be a null pointer.

The *dbm\_open()* function shall return a pointer to a database structure. If an error is detected during the operation, *dbm\_open()* shall return a (DBM \*)0.

## ERRORS

No errors are defined.

## EXAMPLES

None.

## APPLICATION USAGE

The following code can be used to traverse the database:

```
for(key = dbm_firstkey(db); key.dptr != NULL; key = dbm_nextkey(db))
```

The *dbm\_\** functions provided in this library should not be confused in any way with those of a general-purpose database management system. These functions do not provide for multiple search keys per entry, they do not protect against multi-user access (in other words they do not lock records or files), and they do not provide the many other useful database functions that are found in more robust database management systems. Creating and updating databases by use of these functions is relatively slow because of data copies that occur upon hash collisions. These functions are useful for applications requiring fast lookup of relatively static information that is to be indexed by a single key.

Note that a strictly conforming application is extremely limited by these functions: since there is no way to determine that the keys in use do not all hash to the same value (although that would be rare), a strictly conforming application cannot be guaranteed that it can store more than one block's worth of data in the database. As long as a key collision does not occur, additional data may be stored, but because there is no way to determine whether an error is due to a key collision or some other error condition (*dbm\_error()* being effectively a Boolean), once an error is detected, the application is effectively limited to guessing what the error might be if it wishes to continue using these functions.

The *dbm\_delete()* function need not physically reclaim file space, although it does make it available for reuse by the database.

After calling *dbm\_store()* or *dbm\_delete()* during a pass through the keys by *dbm\_firstkey()* and *dbm\_nextkey()*, the application should reset the database by calling *dbm\_firstkey()* before again calling *dbm\_nextkey()*. The contents of these files are unspecified and may not be portable.

Applications should take care that database pathname arguments specified to *dbm\_open()* are not prefixes of unrelated files. This might be done, for example, by placing databases in a separate directory.

Since some implementations use three characters for a suffix and others use four characters for a suffix, applications should ensure that the maximum portable pathname length passed to *dbm\_open()* is no greater than {PATH\_MAX}-4 bytes, with the last component of the pathname no greater than {NAME\_MAX}-4 bytes.

## RATIONALE

Previously the standard required the database to be stored in two files, one file being a directory containing a bitmap of keys and having **.dir** as its suffix. The second file containing all data and having **.pag** as its suffix. This has been changed not to specify the use of the files and to allow newer implementations of the Berkeley DB interface using a single file that have evolved while remaining compatible with the application programming interface. The standard developers considered removing the specific suffixes altogether but decided to retain them so as not to pollute the application file name space more than necessary and to allow for portable backups of the database.

**24906 FUTURE DIRECTIONS**

24907 None.

**24908 SEE ALSO**

24909 *open()*

24910 XBD <ndbm.h>

**24911 CHANGE HISTORY**

24912 First released in Issue 4, Version 2.

**24913 Issue 5**

24914 Moved from X/OPEN UNIX extension to BASE.

24915 Normative text previously in the APPLICATION USAGE section is moved to the  
24916 DESCRIPTION.

24917 A note indicating that these functions need not be reentrant is added to the DESCRIPTION.

**24918 Issue 6**

24919 The normative text is updated to avoid use of the term “must” for application requirements.

**24920 Issue 7**

24921 Austin Group Interpretation 1003.1-2001 #042 is applied so that the DESCRIPTION permits  
24922 newer implementations of the Berkeley DB interface.

24923 Austin Group Interpretation 1003.1-2001 #156 is applied.

24924 **NAME**

24925 difftime — compute the difference between two calendar time values

24926 **SYNOPSIS**

24927 #include &lt;time.h&gt;

24928 double difftime(time\_t time1, time\_t time0);

24929 **DESCRIPTION**

24930 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
24931 conflict between the requirements described here and the ISO C standard is unintentional. This  
24932 volume of POSIX.1-2008 defers to the ISO C standard.

24933 The *difftime()* function shall compute the difference between two calendar times (as returned by  
24934 *time()*): *time1* − *time0*.

24935 **RETURN VALUE**24936 The *difftime()* function shall return the difference expressed in seconds as a type **double**.24937 **ERRORS**

24938 No errors are defined.

24939 **EXAMPLES**

24940 None.

24941 **APPLICATION USAGE**

24942 None.

24943 **RATIONALE**

24944 None.

24945 **FUTURE DIRECTIONS**

24946 None.

24947 **SEE ALSO**24948 *asctime()*, *clock()*, *ctime()*, *gmtime()*, *localtime()*, *mktime()*, *strftime()*, *strptime()*, *time()*, *utime()*

24949 XBD &lt;time.h&gt;

24950 **CHANGE HISTORY**

24951 First released in Issue 4. Derived from the ISO C standard.



24952 **NAME**

24953 dirfd — extract the file descriptor used by a DIR stream

24954 **SYNOPSIS**

```
24955 #include <dirent.h>
24956 int dirfd(DIR *dirp);
```

24957 **DESCRIPTION**

24958 The *dirfd()* function shall return a file descriptor referring to the same directory as the *dirp*  
 24959 argument. This file descriptor shall be closed by a call to *closedir()*. If any attempt is made to  
 24960 close the file descriptor, or to modify the state of the associated description, other than by means  
 24961 XSI of *closedir()*, *readdir()*, *readdir\_r()*, *rewinddir()*, or *seekdir()*, the behavior is undefined.

24962 **RETURN VALUE**

24963 Upon successful completion, the *dirfd()* function shall return an integer which contains a file  
 24964 descriptor for the stream pointed to by *dirp*. Otherwise, it shall return *-1* and shall set *errno* to  
 24965 indicate the error.

24966 **ERRORS**24967 The *dirfd()* function may fail if:

24968 [EINVAL] The *dirp* argument does not refer to a valid directory stream. -

24969 **EXAMPLES**

24970 None.

24971 **APPLICATION USAGE**

24972 The *dirfd()* function is intended to be a mechanism by which an application may obtain a file  
 24973 descriptor to use for the *fchdir()* function.

24974 **RATIONALE**

24975 This interface was introduced because the Base Definitions volume of POSIX.1-2008 does not  
 24976 make public the **DIR** data structure. Applications tend to use the *fchdir()* function on the file  
 24977 descriptor returned by this interface, and this has proven useful for security reasons; in  
 24978 particular, it is a better technique than others where directory names might change.

24979 The description uses the term “a file descriptor” rather than “the file descriptor”. The  
 24980 implication intended is that an implementation that does not use an *fd* for *opendir()* could still  
 24981 *open()* the directory to implement the *dirfd()* function. Such a descriptor must be closed later  
 24982 during a call to *closedir()*.

24983 If it is necessary to allocate an *fd* to be returned by *dirfd()*, it should be done at the time of a call -  
 24984 to *opendir()*.

24985 **FUTURE DIRECTIONS**

24986 None.

24987 **SEE ALSO**24988 *closedir()*, *fchdir()*, *fdopendir()*, *fileno()*, *open()*, *readdir()*

24989 XBD &lt;dirent.h&gt;

24990 **CHANGE HISTORY**

24991 First released in Issue 7.

24992 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0067 [422] is applied.

24993 **NAME**

24994       dirname — report the parent directory name of a file pathname

24995 **SYNOPSIS**

```
24996 XSI      #include <libgen.h>
24997      char *dirname(char *path);
```

24998 **DESCRIPTION**

24999       The *dirname()* function shall take a pointer to a character string that contains a pathname, and  
 25000       return a pointer to a string that is a pathname of the parent directory of that file. The *dirname()* |  
 25001       function shall not perform pathname resolution; the result shall not be affected by whether or |  
 25002       not *path* exists or by its file type. Trailing ' / ' characters in the path that are not also leading ' / ' |  
 25003       characters shall not be counted as part of the path.

25004       If *path* does not contain a ' / ', then *dirname()* shall return a pointer to the string " . ". If *path* is a  
 25005       null pointer or points to an empty string, *dirname()* shall return a pointer to the string " . ".

25006       The *dirname()* function may modify the string pointed to by *path*, and may return a pointer to +  
 25007       static storage that may then be overwritten by a subsequent call to *dirname()*. +

25008       The *dirname()* function need not be thread-safe.

25009 **RETURN VALUE**

25010       The *dirname()* function shall return a pointer to a string as described above. |

25011       The *dirname()* function may modify the string pointed to by *path*, and may return a pointer to |  
 25012       internal storage. The returned pointer might be invalidated or the storage might be overwritten |  
 25013       by a subsequent call to *dirname()*. The returned pointer might also be invalidated if the calling +  
 25014       thread is terminated.

25015 **ERRORS**

25016       No errors are defined.

25017 **EXAMPLES**

25018       The following code fragment reads a pathname, changes the current working directory to the  
 25019       parent directory, and opens the file.

```
25020      char *path = NULL, *pathcopy;
25021      size_t buflen = 0;
25022      ssize_t linelen = 0;
25023      int fd;
25024
25025      linelen = getline(&path, &buflen, stdin);
25026
25027      path[linelen-1] = 0;
25028      pathcopy = strdup(path);
25029      if (chdir(dirname(pathcopy)) < 0) {
25030          ...
25031      }
25032      if ((fd = open(basename(path), O_RDONLY)) >= 0) {
25033          ...
25034          close (fd);
25035      }
25036      free (pathcopy);
25037      free (path);
```

25037 The EXAMPLES section of the *basename()* function (see *basename()*) includes a table showing |  
 25038 examples of the results of processing several sample pathnames by the *basename()* and *dirname()* |  
 25039 functions and by the *basename* and *dirname* utilities.

#### 25040 APPLICATION USAGE

25041 The *dirname()* and *basename()* functions together yield a complete pathname. The expression  
 25042 *dirname(path)* obtains the pathname of the directory where *basename(path)* is found.

25043 Since the meaning of the leading "/" is implementation-defined, *dirname("//foo")* may return  
 25044 either "/" or "/" (but nothing else).

#### 25045 RATIONALE

25046 None.

#### 25047 FUTURE DIRECTIONS

25048 None.

#### 25049 SEE ALSO

25050 *basename()*

25051 XBD <libgen.h>

25052 XCU *basename*, *dirname*

+

#### 25053 CHANGE HISTORY

25054 First released in Issue 4, Version 2.

#### 25055 Issue 5

25056 Moved from X/OPEN UNIX extension to BASE.

25057 Normative text previously in the APPLICATION USAGE section is moved to the  
 25058 DESCRIPTION.

25059 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

#### 25060 Issue 7

25061 Austin Group Interpretation 1003.1-2001 #156 is applied.

25062 The EXAMPLES section is revised.

25063 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0068 [75] is applied.

25064 **NAME**

25065 div — compute the quotient and remainder of an integer division

25066 **SYNOPSIS**

25067 #include &lt;stdlib.h&gt;

25068 div\_t div(int *numer*, int *denom*);25069 **DESCRIPTION**

25070 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
25071 conflict between the requirements described here and the ISO C standard is unintentional. This  
25072 volume of POSIX.1-2008 defers to the ISO C standard.

25073 The *div()* function shall compute the quotient and remainder of the division of the numerator  
25074 *numer* by the denominator *denom*. If the division is inexact, the resulting quotient is the integer  
25075 of lesser magnitude that is the nearest to the algebraic quotient. If the result cannot be  
25076 represented, the behavior is undefined; otherwise, *quot\*denom+rem* shall equal *numer*.

25077 **RETURN VALUE**

25078 The *div()* function shall return a structure of type **div\_t**, comprising both the quotient and the  
25079 remainder. The structure includes the following members, in any order:

```
25080 int quot; /* quotient */  
25081 int rem; /* remainder */
```

25082 **ERRORS**

25083 No errors are defined.

25084 **EXAMPLES**

25085 None.

25086 **APPLICATION USAGE**

25087 None.

25088 **RATIONALE**

25089 None.

25090 **FUTURE DIRECTIONS**

25091 None.

25092 **SEE ALSO**25093 *ldiv()*

25094 XBD &lt;stdlib.h&gt;

25095 **CHANGE HISTORY**

25096 First released in Issue 4. Derived from the ISO C standard.

25097 **NAME**

25098       dlclose — close a symbol table handle

25099 **SYNOPSIS**

25100       #include &lt;dlfcn.h&gt;

25101       int dlclose(void \*handle);

25102 **DESCRIPTION**

25103       The *dlclose()* function shall inform the system that the symbol table handle specified by *handle* is  
 25104       no longer needed by the application.

25105       An application writer may use *dlclose()* to make a statement of intent on the part of the process,  
 25106       but this statement does not create any requirement upon the implementation. When the symbol  
 25107       table handle is closed, the implementation may unload the executable object files that were  
 25108       loaded by *dlopen()* when the symbol table handle was opened and those that were loaded by  
 25109       *dlsym()* when using the symbol table handle identified by *handle*.

25110       Once a symbol table handle has been closed, an application should assume that any symbols  
 25111       (function identifiers and data object identifiers) made visible using *handle*, are no longer  
 25112       available to the process.

25113       Although a *dlclose()* operation is not required to remove any functions or data objects from the  
 25114       address space, neither is an implementation prohibited from doing so. The only restriction on  
 25115       such a removal is that no function nor data object shall be removed to which references have  
 25116       been relocated, until or unless all such references are removed. For instance, an executable object  
 25117       file that had been loaded with a *dlopen()* operation specifying the RTLD\_GLOBAL flag might  
 25118       provide a target for dynamic relocations performed in the processing of other relocatable  
 25119       objects—in such environments, an application may assume that no relocation, once made, shall  
 25120       be undone or remade unless the executable object file containing the relocated object has itself  
 25121       been removed.

25122 **RETURN VALUE**

25123       If the referenced symbol table handle was successfully closed, *dlclose()* shall return 0. If *handle*  
 25124       does not refer to an open symbol table handle or if the symbol table handle could not be closed,  
 25125       *dlclose()* shall return a non-zero value. More detailed diagnostic information shall be available  
 25126       through *dLError()*.

25127 **ERRORS**

25128       No errors are defined.

25129 **EXAMPLES**25130       The following example illustrates use of *dlopen()* and *dlclose()*:

```

25131       #include <dlfcn.h>
25132       int eret;
25133       void *mylib;
25134       ...
25135       /* Open a dynamic library and then close it ... */
25136       mylib = dlopen("mylib.so", RTLD_LOCAL | RTLD_LAZY);
25137       ...
25138       eret = dlclose(mylib);
25139       ...
```

**APPLICATION USAGE**

A conforming application should employ a symbol table handle returned from a *dlopen()* invocation only within a given scope bracketed by a *dlopen()* operation and the corresponding *dlclose()* operation. Implementations are free to use reference counting or other techniques such that multiple calls to *dlopen()* referencing the same executable object file may return a pointer to the same data object as the symbol table handle.

Implementations are also free to re-use a handle. For these reasons, the value of a handle must be treated as an opaque data type by the application, used only in calls to *dlsym()* and *dlclose()*.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*dlerror()*, *dlopen()*, *dlsym()*

XBD <dlfcn.h>

**CHANGE HISTORY**

First released in Issue 5.

**Issue 6**

The DESCRIPTION is updated to say that the referenced object is closed “if this is the last reference to it”.

**Issue 7**

The *dlopen()* function is moved from the XSI option to Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0069 [74] is applied.

**25163 NAME**

25164 **dlderror** — get diagnostic information

**25165 SYNOPSIS**

```
25166 #include <dldfcn.h>
25167 char *dlderror(void);
```

**25168 DESCRIPTION**

25169 The *dlderror()* function shall return a null-terminated character string (with no trailing  
25170 <newline>) that describes the last error that occurred during dynamic linking processing. If no  
25171 dynamic linking errors have occurred since the last invocation of *dlderror()*, *dlderror()* shall return  
25172 NULL. Thus, invoking *dlderror()* a second time, immediately following a prior invocation, shall  
25173 result in NULL being returned.

25174 It is implementation-defined whether or not the *dlderror()* function is thread-safe. A thread-safe  
25175 implementation shall return only errors that occur on the current thread.

**25176 RETURN VALUE**

25177 If successful, *dlderror()* shall return a null-terminated character string; otherwise, NULL shall be  
25178 returned.

25179 The application shall not modify the string returned. The returned pointer might be invalidated  
25180 or the string content might be overwritten by a subsequent call to *dlderror()* in the same thread (if  
25181 *dlderror()* is thread-safe) or in any thread (if *dlderror()* is not thread-safe). The returned pointer  
25182 might also be invalidated if the calling thread is terminated.

**25183 ERRORS**

25184 No errors are defined.

**25185 EXAMPLES**

25186 The following example prints out the last dynamic linking error:

```
25187 ...
25188 #include <dldfcn.h>
25189 char *errstr;
25190 errstr = dlderror();
25191 if (errstr != NULL)
25192     printf ("A dynamic linking error occurred: (%s)\n", errstr);
25193 ...
```

**25194 APPLICATION USAGE**

25195 Depending on the application environment with respect to asynchronous execution events, such  
25196 as signals or other asynchronous computation sharing the address space, conforming  
25197 applications should use a critical section to retrieve the error pointer and buffer.

**25198 RATIONALE**

25199 None.

**25200 FUTURE DIRECTIONS**

25201 None.

**25202 SEE ALSO**

25203 *dldclose()*, *dldopen()*, *dldsym()*  
25204 XBD <dldfcn.h>

**CHANGE HISTORY**

25205  
25206 First released in Issue 5.

**Issue 6**

25207  
25208 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

**Issue 7**

25209  
25210 Austin Group Interpretation 1003.1-2001 #156 is applied.

25211 The *dlerror()* function is moved from the XSI option to the Base.

25212  
25213 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0070 [75], XSH/TC1-2008/0071 [97],  
and XSH/TC1-2008/0072 [133] are applied.



**NAME**

dlopen — open a symbol table handle

**SYNOPSIS**

```
#include <dlfcn.h>
```

```
void *dlopen(const char *file, int mode);
```

**DESCRIPTION**

The *dlopen()* function shall make the symbols (function identifiers and data object identifiers) in the executable object file specified by *file* available to the calling program.

The class of executable object files eligible for this operation and the manner of their construction are implementation-defined, though typically such files are shared libraries or programs.

Implementations may permit the construction of embedded dependencies in executable object files. In such cases, a *dlopen()* operation shall load those dependencies in addition to the executable object file specified by *file*. Implementations may also impose specific constraints on the construction of programs that can employ *dlopen()* and its related services.

A successful *dlopen()* shall return a symbol table handle which the caller may use on subsequent calls to *dlsym()* and *dlclose()*.

The value of this symbol table handle should not be interpreted in any way by the caller.

The *file* argument is used to construct a pathname to the executable object file. If *file* contains a <slash> character, the *file* argument is used as the pathname for the file. Otherwise, *file* is used in an implementation-defined manner to yield a pathname.

If *file* is a null pointer, *dlopen()* shall return a global symbol table handle for the currently running process image. This symbol table handle shall provide access to the symbols from an ordered set of executable object files consisting of the original program image file, any executable object files loaded at program start-up as specified by that process file (for example, shared libraries), and the set of executable object files loaded using *dlopen()* operations with the RTLD\_GLOBAL flag. As the latter set of executable object files can change during execution, the set of symbols made available by this symbol table handle can also change dynamically.

Only a single copy of an executable object file shall be brought into the address space, even if *dlopen()* is invoked multiple times in reference to the executable object file, and even if different pathnames are used to reference the executable object file.

The *mode* parameter describes how *dlopen()* shall operate upon *file* with respect to the processing of relocations and the scope of visibility of the symbols provided within *file*. When an executable object file is brought into the address space of a process, it may contain references to symbols whose addresses are not known until the executable object file is loaded.

These references shall be relocated before the symbols can be accessed. The *mode* parameter governs when these relocations take place and may have the following values:

RTLD_LAZY	Relocations shall be performed at an implementation-defined time, ranging from the time of the <i>dlopen()</i> call until the first reference to a given symbol occurs. Specifying RTLD_LAZY should improve performance on implementations supporting dynamic symbol binding since a process might not reference all of the symbols in an executable object file. And, for systems supporting dynamic symbol resolution for normal process execution, this behavior mimics the normal handling of process execution.
-----------	--

RTLD\_NOW All necessary relocations shall be performed when the executable object file is first loaded. This may waste some processing if relocations are performed for symbols that are never referenced. This behavior may be useful for applications that need to know that all symbols referenced during execution will be available before *dlopen()* returns.

Any executable object file loaded by *dlopen()* that requires relocations against global symbols can reference the symbols in the original process image file, any executable object files loaded at program start-up, from the initial process image itself, from any other executable object file included in the same *dlopen()* invocation, and any executable object files that were loaded in any *dlopen()* invocation and which specified the RTLD\_GLOBAL flag. To determine the scope of visibility for the symbols loaded with a *dlopen()* invocation, the *mode* parameter should be a bitwise-inclusive OR with one of the following values:

RTLD\_GLOBAL The executable object file's symbols shall be made available for relocation processing of any other executable object file. In addition, symbol lookup using *dlopen(NULL,mode)* and an associated *dlsym()* allows executable object files loaded with this mode to be searched.

RTLD\_LOCAL The executable object file's symbols shall not be made available for relocation processing of any other executable object file.

If neither RTLD\_GLOBAL nor RTLD\_LOCAL is specified, the default behavior is unspecified.

If an executable object file is specified in multiple *dlopen()* invocations, *mode* is interpreted at each invocation.

If RTLD\_NOW has been specified, all relocations shall have been completed rendering further RTLD\_NOW operations redundant and any further RTLD\_LAZY operations irrelevant.

If RTLD\_GLOBAL has been specified, the executable object file shall maintain the RTLD\_GLOBAL status regardless of any previous or future specification of RTLD\_LOCAL, as long as the executable object file remains in the address space (see *dlclose()*).

Symbols introduced into the process image through calls to *dlopen()* may be used in relocation activities. Symbols so introduced may duplicate symbols already defined by the program or previous *dlopen()* operations. To resolve the ambiguities such a situation might present, the resolution of a symbol reference to symbol definition is based on a symbol resolution order. Two such resolution orders are defined: load order and dependency order. Load order establishes an ordering among symbol definitions, such that the first definition loaded (including definitions from the process image file and any dependent executable object files loaded with it) has priority over executable object files added later (by *dlopen()*). Load ordering is used in relocation processing. Dependency ordering uses a breadth-first order starting with a given executable object file, then all of its dependencies, then any dependents of those, iterating until all dependencies are satisfied. With the exception of the global symbol table handle obtained via a *dlopen()* operation with a null pointer as the *file* argument, dependency ordering is used by the *dlsym()* function. Load ordering is used in *dlsym()* operations upon the global symbol table handle.

When an executable object file is first made accessible via *dlopen()*, it and its dependent executable object files are added in dependency order. Once all the executable object files are added, relocations are performed using load order. Note that if an executable object file or its dependencies had been previously loaded, the load and dependency orders may yield different resolutions.

The symbols introduced by *dlopen()* operations and available through *dlsym()* are at a minimum those which are exported as identifiers of global scope by the executable object file. Typically,

25305 such identifiers shall be those that were specified in (for example) C source code as having  
 25306 **extern** linkage. The precise manner in which an implementation constructs the set of exported  
 25307 symbols for an executable object file is implementation-defined.

#### 25308 **RETURN VALUE**

25309 Upon successful completion, *dlopen()* shall return a symbol table handle. If *file* cannot be found,  
 25310 cannot be opened for reading, is not of an appropriate executable object file format for  
 25311 processing by *dlopen()*, or if an error occurs during the process of loading *file* or relocating its  
 25312 symbolic references, *dlopen()* shall return a null pointer. More detailed diagnostic information  
 25313 shall be available through *dlderror()*.

#### 25314 **ERRORS**

25315 No errors are defined.

#### 25316 **EXAMPLES**

25317 Refer to *dlsym()*.

#### 25318 **APPLICATION USAGE**

25319 None.

#### 25320 **RATIONALE**

25321 None.

#### 25322 **FUTURE DIRECTIONS**

25323 None.

#### 25324 **SEE ALSO**

25325 *dlclose()*, *dlderror()*, *dlsym()*

25326 XBD <dlfcn.h>

#### 25327 **CHANGE HISTORY**

25328 First released in Issue 5.

#### 25329 **Issue 6**

25330 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/21 is applied, changing the default  
 25331 behavior in the DESCRIPTION when neither RTLD\_GLOBAL nor RTLD\_LOCAL are specified  
 25332 from implementation-defined to unspecified.

#### 25333 **Issue 7**

25334 The *dlopen()* function is moved from the XSI option to the Base.

25335 The EXAMPLES section is updated to refer to *dlsym()*.

25336 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0073 [74] is applied.

**NAME**

dlsym — get the address of a symbol from a symbol table handle

**SYNOPSIS**

```
#include <dlfcn.h>
```

```
void *dlsym(void *restrict handle, const char *restrict name);
```

**DESCRIPTION**

The *dlsym()* function shall obtain the address of a symbol (a function identifier or a data object identifier) defined in the symbol table identified by the *handle* argument. The *handle* argument is a symbol table handle returned from a call to *dlopen()* (and which has not since been released by a call to *dlclose()*), and *name* is the symbol's name as a character string. The return value from *dlsym()*, cast to a pointer to the type of the named symbol, can be used to call (in the case of a function) or access the contents of (in the case of a data object) the named symbol.

The *dlsym()* function shall search for the named symbol in the symbol table referenced by *handle*. If the symbol table was created with lazy loading (see *RTLD\_LAZY* in *dlopen()*), load ordering shall be used in *dlsym()* operations to relocate executable object files needed to resolve the symbol. The symbol resolution algorithm used shall be dependency order as described in *dlopen()*.

The *RTLD\_DEFAULT* and *RTLD\_NEXT* symbolic constants (which may be defined in *<dlfcn.h>*) are reserved for future use as special values that applications may be allowed to use for *handle*.

**RETURN VALUE**

Upon successful completion, if *name* names a function identifier, *dlsym()* shall return the address of the function converted from type pointer to function to type pointer to **void**; otherwise, *dlsym()* shall return the address of the data object associated with the data object identifier named by *name* converted from a pointer to the type of the data object to a pointer to **void**. If *handle* does not refer to a valid symbol table handle or if the symbol named by *name* cannot be found in the symbol table associated with *handle*, *dlsym()* shall return a null pointer.

More detailed diagnostic information shall be available through *dLError()*.

**ERRORS**

No errors are defined.

**EXAMPLES**

The following example shows how *dlopen()* and *dlsym()* can be used to access either a function or a data object. For simplicity, error checking has been omitted.

```
void *handle;
int (*fptr)(int), *iptr, result;
/* open the needed symbol table */
handle = dlopen("/usr/home/me/libfoo.so", RTLD_LOCAL | RTLD_LAZY);
/* find the address of the function my_function */
fptr = (int (*)(int))dlsym(handle, "my_function");
/* find the address of the data object my_object */
iptr = (int *)dlsym(handle, "my_OBJ");
/* invoke my_function, passing the value of my_OBJ as the parameter */
result = (*fptr)(*iptr);
```

**APPLICATION USAGE**

The following special purpose values for *handle* are reserved for future use and have the indicated meanings:

**RTLD\_DEFAULT** The identifier lookup happens in the normal global scope; that is, a search for an identifier using *handle* would find the same definition as a direct use of this identifier in the program code.

**RTLD\_NEXT** Specifies the next executable object file after this one that defines *name*. This one refers to the executable object file containing the invocation of *dlsym()*. The next executable object file is the one found upon the application of a load order symbol resolution algorithm (see *dlopen()*). The next symbol is either one of global scope (because it was introduced as part of the original process image or because it was added with a *dlopen()* operation including the **RTLD\_GLOBAL** flag), or is in an executable object file that was included in the same *dlopen()* operation that loaded this one.

The **RTLD\_NEXT** flag is useful to navigate an intentionally created hierarchy of multiply-defined symbols created through interposition. For example, if a program wished to create an implementation of *malloc()* that embedded some statistics gathering about memory allocations, such an implementation could use the real *malloc()* definition to perform the memory allocation — and itself only embed the necessary logic to implement the statistics gathering function.

Note that conversion from a **void \*** pointer to a function pointer as in:

```
fptr = (int (*)(int))dlsym(handle, "my_function");
```

is not defined by the ISO C standard. This standard requires this conversion to work correctly on conforming implementations.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*dlclose()*, *dlerror()*, *dlopen()*

XBD [\*\*<dlfcn.h>\*\*](#)

**CHANGE HISTORY**

First released in Issue 5.

**Issue 6**

The **restrict** keyword is added to the *dlsym()* prototype for alignment with the ISO/IEC 9899:1999 standard.

The **RTLD\_DEFAULT** and **RTLD\_NEXT** flags are reserved for future use.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/14 is applied, correcting an example, and adding text to the **RATIONALE** describing issues related to conversion of pointers to functions and back again.

**Issue 7**

The *dlsym()* function is moved from the XSI option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0074 [74] is applied.

## 25422 NAME

25423 dprintf — print formatted output

## 25424 SYNOPSIS

```
25425 CX #include <stdio.h>
25426 int dprintf(int fil-des, const char *restrict format, ...);
```

## 25427 DESCRIPTION

25428 Refer to *fprintf()*.

**NAME**

drand48, erand48, jrand48, lcong48, lrand48, mrand48, nrand48, seed48, srand48 — generate uniformly distributed pseudo-random numbers

**SYNOPSIS**

```
XSI #include <stdlib.h>

double drand48(void);
double erand48(unsigned short xsubi[3]);
long jrand48(unsigned short xsubi[3]);
void lcong48(unsigned short param[7]);
long lrand48(void);
long mrand48(void);
long nrand48(unsigned short xsubi[3]);
unsigned short *seed48(unsigned short seed16v[3]);
void srand48(long seedval);
```

**DESCRIPTION**

This family of functions shall generate pseudo-random numbers using a linear congruential algorithm and 48-bit integer arithmetic.

The *drand48()* and *erand48()* functions shall return non-negative, double-precision, floating-point values, uniformly distributed over the interval [0.0,1.0).

The *lrnd48()* and *nrnd48()* functions shall return non-negative, long integers, uniformly distributed over the interval  $[0, 2^{31})$ .

The *mrnd48()* and *jrnd48()* functions shall return signed long integers uniformly distributed over the interval  $[-2^{31}, 2^{31})$ .

The *srand48()*, *seed48()*, and *lcong48()* functions are initialization entry points, one of which should be invoked before either *drand48()*, *lrnd48()*, or *mrnd48()* is called. (Although it is not recommended practice, constant default initializer values shall be supplied automatically if *drand48()*, *lrnd48()*, or *mrnd48()* is called without a prior call to an initialization entry point.) The *erand48()*, *nrnd48()*, and *jrnd48()* functions do not require an initialization entry point to be called first.

All the routines work by generating a sequence of 48-bit integer values,  $X_i$ , according to the linear congruential formula:

$$X_{n+1} = (aX_n + c)_{\text{mod } m} \quad n \geq 0$$

The parameter  $m = 2^{48}$ ; hence 48-bit integer arithmetic is performed. Unless *lcong48()* is invoked, the multiplier value  $a$  and the addend value  $c$  are given by:

$$a = 5\text{DEECE66D}_{16} = 273673163155_8$$

$$c = \text{B}_{16} = 13_8$$

The value returned by any of the *drand48()*, *erand48()*, *jrnd48()*, *lrnd48()*, *mrnd48()*, or *nrnd48()* functions is computed by first generating the next 48-bit  $X_i$  in the sequence. Then the appropriate number of bits, according to the type of data item to be returned, are copied from the high-order (leftmost) bits of  $X_i$  and transformed into the returned value.

The *drand48()*, *lrnd48()*, and *mrnd48()* functions store the last 48-bit  $X_i$  generated in an internal buffer; that is why the application shall ensure that these are initialized prior to being invoked. The *erand48()*, *nrnd48()*, and *jrnd48()* functions require the calling program to



provide storage for the successive  $X_i$  values in the array specified as an argument when the functions are invoked. That is why these routines do not have to be initialized; the calling program merely has to place the desired initial value of  $X_i$  into the array and pass it as an argument. By using different arguments, *erand48()*, *rand48()*, and *jrand48()* allow separate modules of a large program to generate several *independent* streams of pseudo-random numbers; that is, the sequence of numbers in each stream shall *not* depend upon how many times the routines are called to generate numbers for the other streams.

The initializer function *srand48()* sets the high-order 32 bits of  $X_i$  to the low-order 32 bits contained in its argument. The low-order 16 bits of  $X_i$  are set to the arbitrary value  $330E_{16}$ .

The initializer function *seed48()* sets the value of  $X_i$  to the 48-bit value specified in the argument array. The low-order 16 bits of  $X_i$  are set to the low-order 16 bits of *seed16v*[0]. The mid-order 16 bits of  $X_i$  are set to the low-order 16 bits of *seed16v*[1]. The high-order 16 bits of  $X_i$  are set to the low-order 16 bits of *seed16v*[2]. In addition, the previous value of  $X_i$  is copied into a 48-bit internal buffer, used only by *seed48()*, and a pointer to this buffer is the value returned by *seed48()*. This returned pointer, which can just be ignored if not needed, is useful if a program is to be restarted from a given point at some future time—use the pointer to get at and store the last  $X_i$  value, and then use this value to reinitialize via *seed48()* when the program is restarted.

The initializer function *lcong48()* allows the user to specify the initial  $X_i$ , the multiplier value  $a$ , and the addend value  $c$ . Argument array elements *param*[0-2] specify  $X_i$ , *param*[3-5] specify the multiplier  $a$ , and *param*[6] specifies the 16-bit addend  $c$ . After *lcong48()* is called, a subsequent call to either *srand48()* or *seed48()* shall restore the standard multiplier and addend values,  $a$  and  $c$ , specified above.

The *drand48()*, *lrand48()*, and *mrand48()* functions need not be thread-safe.

#### RETURN VALUE

As described in the DESCRIPTION above.

#### ERRORS

No errors are defined.

#### EXAMPLES

None.

#### APPLICATION USAGE

These functions should be avoided whenever non-trivial requirements (including safety) have to be fulfilled.

#### RATIONALE

None.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*initstate()*, *rand()*

XBD <stdlib.h>

#### CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.



25513 **Issue 5**

25514 A note indicating that the *drand48()*, *lrand48()*, and *mrnd48()* functions need not be reentrant is  
25515 added to the DESCRIPTION.

25516 **Issue 6**

25517 The normative text is updated to avoid use of the term “must” for application requirements.

25518 **Issue 7**

25519 Austin Group Interpretation 1003.1-2001 #156 is applied.

25520 **NAME**

25521 dup, dup2 — duplicate an open file descriptor

25522 **SYNOPSIS**

```
25523 #include <unistd.h>
25524 int dup(int fildes);
25525 int dup2(int fildes, int fildes2);
```

25526 **DESCRIPTION**

25527 The *dup()* function provides an alternative interface to the service provided by *fcntl()* using the  
 25528 F\_DUPFD command. The call *dup(filde)s* shall be equivalent to:

```
25529 fcntl(fildes, F_DUPFD, 0);
```

25530 The *dup2()* function shall cause the file descriptor *filde*s2 to refer to the same open file  
 25531 description as the file descriptor *filde*s and to share any locks, and shall return *filde*s2. If *filde*s2 is  
 25532 already a valid open file descriptor, it shall be closed first, unless *filde*s is equal to *filde*s2 in which  
 25533 case *dup2()* shall return *filde*s2 without closing it. If the close operation fails to close *filde*s2,  
 25534 *dup2()* shall return -1 without changing the open file description to which *filde*s2 refers. If *filde*s  
 25535 is not a valid file descriptor, *dup2()* shall return -1 and shall not close *filde*s2. If *filde*s2 is less than  
 25536 0 or greater than or equal to {OPEN\_MAX}, *dup2()* shall return -1 with *errno* set to [EBADF].

25537 Upon successful completion, if *filde*s is not equal to *filde*s2, the FD\_CLOEXEC flag associated  
 25538 with *filde*s2 shall be cleared. If *filde*s is equal to *filde*s2, the FD\_CLOEXEC flag associated with  
 25539 *filde*s2 shall not be changed.

25540 TYM If *filde*s refers to a typed memory object, the result of the *dup2()* function is unspecified.

25541 **RETURN VALUE**

25542 Upon successful completion a non-negative integer, namely the file descriptor, shall be returned;  
 25543 otherwise, -1 shall be returned and *errno* set to indicate the error.

25544 **ERRORS**

25545 The *dup()* function shall fail if:

25546 [EBADF] The *filde*s argument is not a valid open file descriptor.

25547 [EMFILE] All file descriptors available to the process are currently open.

25548 The *dup2()* function shall fail if:

25549 [EBADF] The *filde*s argument is not a valid open file descriptor or the argument *filde*s2 is  
 25550 negative or greater than or equal to {OPEN\_MAX}.

25551 [EINTR] The *dup2()* function was interrupted by a signal.

25552 The *dup2()* function may fail if:

25553 [EIO] An I/O error occurred while attempting to close *filde*s2.

25554 **EXAMPLES**25555 **Redirecting Standard Output to a File**

25556 The following example closes standard output for the current processes, re-assigns standard  
 25557 output to go to the file referenced by *pdf*, and closes the original file descriptor to clean up.

```
25558 #include <unistd.h>
25559 ...
25560 int pdf;
25561 ...
```

```

25562     close(1);
25563     dup(pfd);
25564     close(pfd);
25565     ...

```

### 25566 Redirecting Error Messages

25567 The following example redirects messages from *stderr* to *stdout*.

```

25568     #include <unistd.h>
25569     ...
25570     dup2(1, 2);
25571     ...

```

### 25572 APPLICATION USAGE

25573 Implementations may use file descriptors that must be inherited into child processes for the  
 25574 child process to remain conforming, such as for message catalog or tracing purposes. Therefore,  
 25575 an application that calls *dup2()* with an arbitrary integer for *filides2* risks non-conforming  
 25576 behavior, and *dup2()* can only portably be used to overwrite file descriptor values that the  
 25577 application has obtained through explicit actions, or for the three file descriptors corresponding  
 25578 to the standard file streams. In order to avoid a race condition of leaking an unintended file  
 25579 descriptor into a child process, an application should consider opening all file descriptors with  
 25580 the FD\_CLOEXEC bit set unless the file descriptor is intended to be inherited across *exec*.

### 25581 RATIONALE

25582 The *dup()* function is redundant. Its services are also provided by the *fcntl()* function. It has been  
 25583 included in this volume of POSIX.1-2008 primarily for historical reasons, since many existing  
 25584 applications use it. On the other hand, the *dup2()* function provides unique services, as no other  
 25585 interface is able to atomically replace an existing file descriptor.

25586 The *dup2()* function is not marked obsolescent because it presents a type-safe version of  
 25587 functionality provided in a type-unsafe version by *fcntl()*. It is used in the POSIX Ada binding.

25588 The *dup2()* function is not intended for use in critical regions as a synchronization mechanism.

25589 In the description of [EBADF], the case of *filides* being out of range is covered by the given case of  
 25590 *filides* not being valid. The descriptions for *filides* and *filides2* are different because the only kind of  
 25591 invalidity that is relevant for *filides2* is whether it is out of range; that is, it does not matter  
 25592 whether *filides2* refers to an open file when the *dup2()* call is made.

### 25593 FUTURE DIRECTIONS

25594 None.

### 25595 SEE ALSO

25596 *close()*, *fcntl()*, *open()*

25597 XBD <unistd.h>

### 25598 CHANGE HISTORY

25599 First released in Issue 1. Derived from Issue 1 of the SVID.

### 25600 Issue 7

25601 SD5-XSH-ERN-187 is applied.

25602 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0075 [149,428] and  
 25603 XSH/TC1-2008/0076 [149] are applied.

25604 **NAME**

25605 duplocale — duplicate a locale object

25606 **SYNOPSIS**

```
25607 CX    #include <locale.h>
25608        locale_t duplocale(locale_t locobj);
```

25609 **DESCRIPTION**

25610 The *duplocale()* function shall create a duplicate copy of the locale object referenced by the *locobj*  
25611 argument.

25612 If the *locobj* argument is *LC\_GLOBAL\_LOCALE*, *duplocale()* shall create a new locale object  
25613 containing a copy of the global locale determined by the *setlocale()* function.

25614 The behavior is undefined if the *locobj* argument is not a valid locale object handle.

25615 **RETURN VALUE**

25616 Upon successful completion, the *duplocale()* function shall return a handle for a new locale  
25617 object. Otherwise, *duplocale()* shall return **(locale\_t)0** and set *errno* to indicate the error.

25618 **ERRORS**

25619 The *duplocale()* function shall fail if:

25620 [ENOMEM] There is not enough memory available to create the locale object or load the  
25621 locale data.

25622 **EXAMPLES**25623 **Constructing an Altered Version of an Existing Locale Object**

25624 The following example shows a code fragment to create a slightly altered version of an existing  
25625 locale object. The function takes a locale object and a locale name and it replaces the *LC\_TIME*  
25626 category data in the locale object with that from the named locale.

```
25627 #include <locale.h>
25628 ...
25629 locale_t
25630 with_changed_lc_time (locale_t obj, const char *name)
25631 {
25632     locale_t retval = duplocale (obj);
25633     if (retval != (locale_t) 0)
25634     {
25635         locale_t changed = newlocale (LC_TIME_MASK, name, retval);
25636         if (changed == (locale_t) 0)
25637             /* An error occurred. Free all allocated resources. */
25638             freelocale (retval);
25639         retval = changed;
25640     }
25641     return retval;
25642 }
```

**APPLICATION USAGE**

The use of the *duplocale()* function is recommended for situations where a locale object is being used in multiple places, and it is possible that the lifetime of the locale object might end before all uses are finished. Another reason to duplicate a locale object is if a slightly modified form is needed. This can be achieved by a call to *newlocale()* following the *duplocale()* call.

As with the *newlocale()* function, handles for locale objects created by the *duplocale()* function should be released by a corresponding call to *freelocale()*.

The *duplocale()* function can also be used in conjunction with *uselocale((locale\_t)0)*. This returns the locale in effect for the calling thread, but can have the value LC\_GLOBAL\_LOCALE. Passing LC\_GLOBAL\_LOCALE to functions such as *isalnum\_l()* results in undefined behavior, but applications can convert it into a usable locale object by using *duplocale()*.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*freelocale()*, *newlocale()*, *uselocale()*

XBD <locale.h>

**CHANGE HISTORY**

First released in Issue 7.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0077 [283,301], XSH/TC1-2008/0078 [283], and XSH/TC1-2008/0079 [301] are applied.

25665 **NAME**25666       encrypt — encoding function (**CRYPT**)25667 **SYNOPSIS**

```
25668 XSI      #include <unistd.h>
25669          void encrypt(char block[64], int edflag);
```

25670 **DESCRIPTION**

25671       The *encrypt()* function shall provide access to an implementation-defined encoding algorithm.  
25672       The key generated by *setkey()* is used to encrypt the string *block* with *encrypt()*.

25673       The *block* argument to *encrypt()* shall be an array of length 64 bytes containing only the bytes  
25674       with values of 0 and 1. The array is modified in place to a similar array using the key set by  
25675       *setkey()*. If *edflag* is 0, the argument is encoded. If *edflag* is 1, the argument may be decoded (see  
25676       the APPLICATION USAGE section); if the argument is not decoded, *errno* shall be set to  
25677       [ENOSYS].

25678       The *encrypt()* function shall not change the setting of *errno* if successful. An application wishing  
25679       to check for error situations should set *errno* to 0 before calling *encrypt()*. If *errno* is non-zero on  
25680       return, an error has occurred.

25681       The *encrypt()* function need not be thread-safe.

25682 **RETURN VALUE**

25683       The *encrypt()* function shall not return a value.

25684 **ERRORS**

25685       The *encrypt()* function shall fail if:

25686       [ENOSYS]       The functionality is not supported on this implementation.

25687 **EXAMPLES**

25688       None.

25689 **APPLICATION USAGE**

25690       Historical implementations of the *encrypt()* function used a rather primitive encoding algorithm.

25691       In some environments, decoding might not be implemented. This is related to some Government  
25692       restrictions on encryption and decryption routines. Historical practice has been to ship a  
25693       different version of the encryption library without the decryption feature in the routines  
25694       supplied. Thus the exported version of *encrypt()* does encoding but not decoding.

25695 **RATIONALE**

25696       None.

25697 **FUTURE DIRECTIONS**

25698       A future version of the standard may mark this interface as obsolete or remove it altogether. |

25699 **SEE ALSO**

25700       *crypt()*, *setkey()*

25701       XBD <unistd.h>

25702 **CHANGE HISTORY**

25703       First released in Issue 1. Derived from Issue 1 of the SVID.

25704 **Issue 5**

25705 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

25706 **Issue 6**

25707 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

25708 **Issue 7**

25709 Austin Group Interpretation 1003.1-2001 #156 is applied.

25710 **NAME**

25711 endgrent, getgrent, setgrent — group database entry functions

25712 **SYNOPSIS**

```

25713 XSI      #include <grp.h>
25714          void endgrent(void);
25715          struct group *getgrent(void);
25716          void setgrent(void);

```

25717 **DESCRIPTION**

25718 The *getgrent()* function shall return a pointer to a structure containing the broken-out fields of an  
 25719 entry in the group database. If the group database is not already open, *getgrent()* shall open it  
 25720 and return a pointer to a **group** structure containing the first entry in the database. Thereafter, it  
 25721 shall return a pointer to a **group** structure containing the next group structure in the group  
 25722 database, so successive calls may be used to search the entire database.

25723 An implementation that provides extended security controls may impose further  
 25724 implementation-defined restrictions on accessing the group database. In particular, the system  
 25725 may deny the existence of some or all of the group database entries associated with groups other  
 25726 than those groups associated with the caller and may omit users other than the caller from the  
 25727 list of members of groups in database entries that are returned.

25728 The *setgrent()* function shall rewind the group database so that the next *getgrent()* call returns  
 25729 the first entry, allowing repeated searches.

25730 The *endgrent()* function shall close the group database.

25731 The *setgrent()* and *endgrent()* functions shall not change the setting of *errno* if successful.

25732 On error, the *setgrent()* and *endgrent()* functions shall set *errno* to indicate the error.

25733 Since no value is returned by the *setgrent()* and *endgrent()* functions, an application wishing to  
 25734 check for error situations should set *errno* to 0, then call the function, then check *errno*.

25735 These functions need not be thread-safe.

25736 **RETURN VALUE**

25737 On successful completion, *getgrent()* shall return a pointer to a group structure. On end-of-file,  
 25738 *getgrent()* shall return a null pointer and shall not change the setting of *errno*. On error,  
 25739 *getgrent()* shall return a null pointer and *errno* shall be set to indicate the error.

25740 The application shall not modify the structure to which the return value points, nor any storage  
 25741 areas pointed to by pointers within the structure. The returned pointer, and pointers within the  
 25742 structure, might be invalidated or the structure or the storage areas might be overwritten by a  
 25743 subsequent call to *getgrgid()*, *getgrnam()*, or *getgrent()*. The returned pointer, and pointers  
 25744 within the structure, might also be invalidated if the calling thread is terminated.

25745 **ERRORS**

25746 These functions may fail if:

25747 [EINTR] A signal was caught during the operation.

25748 [EIO] An I/O error has occurred.

25749 In addition, the *getgrent()* and *setgrent()* functions may fail if:

25750 [EMFILE] All file descriptors available to the process are currently open.



25751 [ENFILE] The maximum allowable number of files is currently open in the system.

25752 **EXAMPLES**

25753 None.

25754 **APPLICATION USAGE**

25755 These functions are provided due to their historical usage. Applications should avoid

25756 dependencies on fields in the group database, whether the database is a single file, or where in

25757 the file system name space the database resides. Applications should use *getgrnam()* and

25758 *getgrgid()* whenever possible because it avoids these dependencies.

25759 **RATIONALE**

25760 None.

25761 **FUTURE DIRECTIONS**

25762 None.

25763 **SEE ALSO**

25764 *endpwent()*, *getgrgid()*, *getgrnam()*, *getlogin()*

25765 XBD <grp.h>

25766 **CHANGE HISTORY**

25767 First released in Issue 4, Version 2.

25768 **Issue 5**

25769 Moved from X/OPEN UNIX extension to BASE.

25770 Normative text previously in the APPLICATION USAGE section is moved to the RETURN

25771 VALUE section.

25772 A note indicating that these functions need not be reentrant is added to the DESCRIPTION.

25773 **Issue 6**

25774 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

25775 **Issue 7**

25776 Austin Group Interpretation 1003.1-2001 #156 is applied.

25777 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

25778 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0080 [75] is applied.

25779 **NAME**

25780 endhostent, gethostent, sethostent — network host database functions

25781 **SYNOPSIS**

```
25782 #include <netdb.h>
25783 void endhostent(void);
25784 struct hostent *gethostent(void);
25785 void sethostent(int stayopen);
```

25786 **DESCRIPTION**

25787 These functions shall retrieve information about hosts. This information is considered to be  
25788 stored in a database that can be accessed sequentially or randomly. The implementation of this  
25789 database is unspecified.

25790 **Note:** In many cases this database is implemented by the Domain Name System, as documented in  
25791 RFC 1034, RFC 1035, and RFC 1886.

25792 The *sethostent()* function shall open a connection to the database and set the next entry for  
25793 retrieval to the first entry in the database. If the *stayopen* argument is non-zero, the connection  
25794 shall not be closed by a call to *gethostent()*, and the implementation may maintain an open file  
25795 descriptor.

25796 The *gethostent()* function shall read the next entry in the database, opening and closing a  
25797 connection to the database as necessary.

25798 Entries shall be returned in **hostent** structures.

25799 The *endhostent()* function shall close the connection to the database, releasing any open file  
25800 descriptor.

25801 These functions need not be thread-safe.

25802 **RETURN VALUE**

25803 Upon successful completion, the *gethostent()* function shall return a pointer to a **hostent**  
25804 structure if the requested entry was found, and a null pointer if the end of the database was  
25805 reached or the requested entry was not found.

25806 The application shall not modify the structure to which the return value points, nor any storage  
25807 areas pointed to by pointers within the structure. The returned pointer, and pointers within the  
25808 structure, might be invalidated or the structure or the storage areas might be overwritten by a  
25809 subsequent call to *gethostent()*. The returned pointer, and pointers within the structure, might  
25810 also be invalidated if the calling thread is terminated. +

25811 **ERRORS**

25812 No errors are defined for *endhostent()*, *gethostent()*, and *sethostent()*.

25813 **EXAMPLES**

25814 None.

25815 **APPLICATION USAGE**

25816 None.

25817 **RATIONALE**

25818 None.

25819 **FUTURE DIRECTIONS**

25820 None.

25821 **SEE ALSO**25822 *endservent()*

25823 XBD &lt;netdb.h&gt;

25824 **CHANGE HISTORY**

25825 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

25826 **Issue 7**

25827 Austin Group Interpretation 1003.1-2001 #156 is applied.

25828 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0081 [75,428] and  
25829 XSH/TC1-2008/0082 [75] are applied.

**NAME**

endnetent, getnetbyaddr, getnetbyname, getnetent, setnetent — network database functions

**SYNOPSIS**

```
#include <netdb.h>

void endnetent(void);
struct netent *getnetbyaddr(uint32_t net, int type);
struct netent *getnetbyname(const char *name);
struct netent *getnetent(void);
void setnetent(int stayopen);
```

**DESCRIPTION**

These functions shall retrieve information about networks. This information is considered to be stored in a database that can be accessed sequentially or randomly. The implementation of this database is unspecified.

The *setnetent()* function shall open and rewind the database. If the *stayopen* argument is non-zero, the connection to the *net* database shall not be closed after each call to *getnetent()* (either directly, or indirectly through one of the other *getnet\*()* functions), and the implementation may maintain an open file descriptor to the database.

The *getnetent()* function shall read the next entry of the database, opening and closing a connection to the database as necessary.

The *getnetbyaddr()* function shall search the database from the beginning, and find the first entry for which the address family specified by *type* matches the *n\_addrtype* member and the network number *net* matches the *n\_net* member, opening and closing a connection to the database as necessary. The *net* argument shall be the network number in host byte order.

The *getnetbyname()* function shall search the database from the beginning and find the first entry for which the network name specified by *name* matches the *n\_name* member, opening and closing a connection to the database as necessary.

The *getnetbyaddr()*, *getnetbyname()*, and *getnetent()* functions shall each return a pointer to a **netent** structure, the members of which shall contain the fields of an entry in the network database.

The *endnetent()* function shall close the database, releasing any open file descriptor.

These functions need not be thread-safe.

**RETURN VALUE**

Upon successful completion, *getnetbyaddr()*, *getnetbyname()*, and *getnetent()* shall return a pointer to a **netent** structure if the requested entry was found, and a null pointer if the end of the database was reached or the requested entry was not found. Otherwise, a null pointer shall be returned.

The application shall not modify the structure to which the return value points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *getnetbyaddr()*, *getnetbyname()*, or *getnetent()*. The returned pointer, and pointers within the structure, might also be invalidated if the calling thread is terminated.

**ERRORS**

No errors are defined.

25873 **EXAMPLES**

25874 None.

25875 **APPLICATION USAGE**

25876 None.

25877 **RATIONALE**

25878 None.

25879 **FUTURE DIRECTIONS**

25880 None.

25881 **SEE ALSO**25882 XBD [<netdb.h>](#)25883 **CHANGE HISTORY**

25884 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

25885 **Issue 7**

25886 Austin Group Interpretation 1003.1-2001 #156 is applied.

25887 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0083 [75] and XSH/TC1-2008/0084  
25888 [75] are applied.

## NAME

endprotoent, getprotobyname, getprotobynumber, getprotoent, setprotoent — network protocol database functions

## SYNOPSIS

```
#include <netdb.h>

void endprotoent(void);
struct protoent *getprotobyname(const char *name);
struct protoent *getprotobynumber(int proto);
struct protoent *getprotoent(void);
void setprotoent(int stayopen);
```

## DESCRIPTION

These functions shall retrieve information about protocols. This information is considered to be stored in a database that can be accessed sequentially or randomly. The implementation of this database is unspecified.

The *setprotoent()* function shall open a connection to the database, and set the next entry to the first entry. If the *stayopen* argument is non-zero, the connection to the network protocol database shall not be closed after each call to *getprotoent()* (either directly, or indirectly through one of the other *getproto\**(*)* functions), and the implementation may maintain an open file descriptor for the database.

The *getprotobyname()* function shall search the database from the beginning and find the first entry for which the protocol name specified by *name* matches the *p\_name* member, opening and closing a connection to the database as necessary.

The *getprotobynumber()* function shall search the database from the beginning and find the first entry for which the protocol number specified by *proto* matches the *p\_proto* member, opening and closing a connection to the database as necessary.

The *getprotoent()* function shall read the next entry of the database, opening and closing a connection to the database as necessary.

The *getprotobyname()*, *getprotobynumber()*, and *getprotoent()* functions shall each return a pointer to a **protoent** structure, the members of which shall contain the fields of an entry in the network protocol database.

The *endprotoent()* function shall close the connection to the database, releasing any open file descriptor.

These functions need not be thread-safe.

## RETURN VALUE

Upon successful completion, *getprotobyname()*, *getprotobynumber()*, and *getprotoent()* return a pointer to a **protoent** structure if the requested entry was found, and a null pointer if the end of the database was reached or the requested entry was not found. Otherwise, a null pointer is returned.

The application shall not modify the structure to which the return value points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *getprotobyname()*, *getprotobynumber()*, or *getprotoent()*. The returned pointer, and pointers within the structure, might also be invalidated if the calling thread is terminated.

**25932 ERRORS**

25933 No errors are defined.

**25934 EXAMPLES**

25935 None.

**25936 APPLICATION USAGE**

25937 None.

**25938 RATIONALE**

25939 None.

**25940 FUTURE DIRECTIONS**

25941 None.

**25942 SEE ALSO**

25943 XBD <netdb.h>

**25944 CHANGE HISTORY**

25945 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

**25946 Issue 7**

25947 Austin Group Interpretation 1003.1-2001 #156 is applied.

25948 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0085 [75] and XSH/TC1-2008/0086  
25949 [75] are applied.

**NAME**

endpwent, getpwent, setpwent — user database functions

**SYNOPSIS**

```
XSI    #include <pwd.h>
      void endpwent(void);
      struct passwd *getpwent(void);
      void setpwent(void);
```

**DESCRIPTION**

These functions shall retrieve information about users.

The *getpwent()* function shall return a pointer to a structure containing the broken-out fields of an entry in the user database. Each entry in the user database contains a **passwd** structure. If the user database is not already open, *getpwent()* shall open it and return a pointer to a **passwd** structure containing the first entry in the database. Thereafter, it shall return a pointer to a **passwd** structure containing the next entry in the user database. Successive calls can be used to search the entire user database.

If an end-of-file or an error is encountered on reading, *getpwent()* shall return a null pointer.

An implementation that provides extended security controls may impose further implementation-defined restrictions on accessing the user database. In particular, the system may deny the existence of some or all of the user database entries associated with users other than the caller.

The *setpwent()* function shall rewind the user database so that the next *getpwent()* call returns the first entry, allowing repeated searches.

The *endpwent()* function shall close the user database.

The *setpwent()* and *endpwent()* functions shall not change the setting of *errno* if successful.

On error, the *setpwent()* and *endpwent()* functions shall set *errno* to indicate the error.

Since no value is returned by the *setpwent()* and *endpwent()* functions, an application wishing to check for error situations should set *errno* to 0, then call the function, then check *errno*.

These functions need not be thread-safe.

**RETURN VALUE**

On successful completion, *getpwent()* shall return a pointer to a **passwd** structure. On end-of-file, *getpwent()* shall return a null pointer and shall not change the setting of *errno*. On error, *getpwent()* shall return a null pointer and *errno* shall be set to indicate the error.

The application shall not modify the structure to which the return value points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *getpwuid()*, *getpwnam()*, or *getpwent()*. The returned pointer, and pointers within the structure, might also be invalidated if the calling thread is terminated.

**ERRORS**

These functions may fail if:

[EINTR] A signal was caught during the operation.

[EIO] An I/O error has occurred.

In addition, *getpwent()* and *setpwent()* may fail if:



- 25992 [EMFILE] All file descriptors available to the process are currently open.
- 25993 [ENFILE] The maximum allowable number of files is currently open in the system.

## 25994 EXAMPLES

### 25995 Searching the User Database

25996 The following example uses the *getpwent()* function to get successive entries in the user  
 25997 database, returning a pointer to a **passwd** structure that contains information about each user.  
 25998 The call to *endpwent()* closes the user database and cleans up.

```
25999 #include <pwd.h>
26000 #include <stdio.h>

26001 void printname(uid_t uid)
26002 {
26003     struct passwd *pwd;
26004     setpwent();
26005     while((pwd = getpwent()) != NULL) {
26006         if (pwd->pw_uid == uid) {
26007             printf("name=%s\n", pwd->pw_name);
26008             break;
26009         }
26010     }
26011     endpwent();
26012 }
```

## 26013 APPLICATION USAGE

26014 These functions are provided due to their historical usage. Applications should avoid  
 26015 dependencies on fields in the password database, whether the database is a single file, or where  
 26016 in the file system name space the database resides. Applications should use *getpwuid()*  
 26017 whenever possible because it avoids these dependencies.

## 26018 RATIONALE

26019 None.

## 26020 FUTURE DIRECTIONS

26021 None.

## 26022 SEE ALSO

26023 *endgrent()*, *getlogin()*, *getpwnam()*, *getpwuid()*

26024 XBD **<pwd.h>**

## 26025 CHANGE HISTORY

26026 First released in Issue 4, Version 2.

### 26027 Issue 5

26028 Moved from X/OPEN UNIX extension to BASE.

26029 Normative text previously in the APPLICATION USAGE section is moved to the RETURN  
 26030 VALUE section.

26031 A note indicating that these functions need not be reentrant is added to the DESCRIPTION.

26032 **Issue 6**

26033 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

26034 **Issue 7**

26035 Austin Group Interpretation 1003.1-2001 #156 is applied.

26036 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

26037 The EXAMPLES section is revised.

26038 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0087 [75] is applied.

**NAME**

endservent, getservbyname, getservbyport, getservent, setservent — network services database functions

**SYNOPSIS**

```
#include <netdb.h>

void endservent(void);
struct servent *getservbyname(const char *name, const char *proto);
struct servent *getservbyport(int port, const char *proto);
struct servent *getservent(void);
void setservent(int stayopen);
```

**DESCRIPTION**

These functions shall retrieve information about network services. This information is considered to be stored in a database that can be accessed sequentially or randomly. The implementation of this database is unspecified.

The *setservent()* function shall open a connection to the database, and set the next entry to the first entry. If the *stayopen* argument is non-zero, the *net* database shall not be closed after each call to the *getservent()* function (either directly, or indirectly through one of the other *getserv\*()* functions), and the implementation may maintain an open file descriptor for the database.

The *getservent()* function shall read the next entry of the database, opening and closing a connection to the database as necessary.

The *getservbyname()* function shall search the database from the beginning and find the first entry for which the service name specified by *name* matches the *s\_name* member and the protocol name specified by *proto* matches the *s\_proto* member, opening and closing a connection to the database as necessary. If *proto* is a null pointer, any value of the *s\_proto* member shall be matched.

The *getservbyport()* function shall search the database from the beginning and find the first entry for which the port specified by *port* matches the *s\_port* member and the protocol name specified by *proto* matches the *s\_proto* member, opening and closing a connection to the database as necessary. If *proto* is a null pointer, any value of the *s\_proto* member shall be matched. The *port* argument shall be a value obtained by converting a **uint16\_t** in network byte order to **int**.

The *getservbyname()*, *getservbyport()*, and *getservent()* functions shall each return a pointer to a **servent** structure, the members of which shall contain the fields of an entry in the network services database.

The *endservent()* function shall close the database, releasing any open file descriptor.

These functions need not be thread-safe.

**RETURN VALUE**

Upon successful completion, *getservbyname()*, *getservbyport()*, and *getservent()* return a pointer to a **servent** structure if the requested entry was found, and a null pointer if the end of the database was reached or the requested entry was not found. Otherwise, a null pointer is returned.

The application shall not modify the structure to which the return value points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *getservbyname()*, *getservbyport()*, or *getservent()*. The returned pointer, and pointers within the structure, might also be invalidated if the calling thread is terminated.

26083 **ERRORS**

26084 No errors are defined.

26085 **EXAMPLES**

26086 None.

26087 **APPLICATION USAGE**26088 The *port* argument of *getservbyport()* need not be compatible with the port values of all address  
26089 families.26090 **RATIONALE**

26091 None.

26092 **FUTURE DIRECTIONS**

26093 None.

26094 **SEE ALSO**26095 *endhostent()*, *endprotoent()*, *htonl()*, *inet\_addr()*

26096 XBD &lt;netdb.h&gt;

26097 **CHANGE HISTORY**

26098 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

26099 **Issue 7**

26100 Austin Group Interpretation 1003.1-2001 #156 is applied.

26101 SD5-XBD-ERN-14 is applied.

26102 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0088 [75] and XSH/TC1-2008/0089  
26103 [75] are applied.

26104 **NAME**

26105 endutxent, getutxent, getutxid, getutxline, pututxline, setutxent — user accounting database  
 26106 functions

26107 **SYNOPSIS**

```
26108 XSI      #include <utmpx.h>
26109
26109 void endutxent(void);
26110 struct utmpx *getutxent(void);
26111 struct utmpx *getutxid(const struct utmpx *id);
26112 struct utmpx *getutxline(const struct utmpx *line);
26113 struct utmpx *pututxline(const struct utmpx *utmpx);
26114 void setutxent(void);
```

26115 **DESCRIPTION**

26116 These functions shall provide access to the user accounting database.

26117 The *getutxent()* function shall read the next entry from the user accounting database. If the  
 26118 database is not already open, it shall open it. If it reaches the end of the database, it shall fail.

26119 The *getutxid()* function shall search forward from the current point in the database. If the  
 26120 *ut\_type* value of the **utmpx** structure pointed to by *id* is *BOOT\_TIME*, *OLD\_TIME*, or  
 26121 *NEW\_TIME*, then it shall stop when it finds an entry with a matching *ut\_type* value. If the  
 26122 *ut\_type* value is *INIT\_PROCESS*, *LOGIN\_PROCESS*, *USER\_PROCESS*, or *DEAD\_PROCESS*,  
 26123 then it shall stop when it finds an entry whose type is one of these four and whose *ut\_id* member  
 26124 matches the *ut\_id* member of the **utmpx** structure pointed to by *id*. If the end of the database is  
 26125 reached without a match, *getutxid()* shall fail.

26126 The *getutxline()* function shall search forward from the current point in the database until it  
 26127 finds an entry of the type *LOGIN\_PROCESS* or *USER\_PROCESS* which also has a *ut\_line* value  
 26128 matching that in the **utmpx** structure pointed to by *line*. If the end of the database is reached  
 26129 without a match, *getutxline()* shall fail.

26130 The *getutxid()* or *getutxline()* function may cache data. For this reason, to use *getutxline()* to  
 26131 search for multiple occurrences, the application shall zero out the static data after each success,  
 26132 or *getutxline()* may return a pointer to the same **utmpx** structure.

26133 There is one exception to the rule about clearing the structure before further reads are done. The  
 26134 implicit read done by *pututxline()* (if it finds that it is not already at the correct place in the user  
 26135 accounting database) shall not modify the static structure returned by *getutxent()*, *getutxid()*, or  
 26136 *getutxline()*, if the application has modified this structure and passed the pointer back to  
 26137 *pututxline()*.

26138 For all entries that match a request, the *ut\_type* member indicates the type of the entry. Other  
 26139 members of the entry shall contain meaningful data based on the value of the *ut\_type* member as  
 26140 follows:

ut_type Member	Other Members with Meaningful Data
EMPTY	No others
BOOT_TIME	<i>ut_tv</i>
OLD_TIME	<i>ut_tv</i>
NEW_TIME	<i>ut_tv</i>
USER_PROCESS	<i>ut_id</i> , <i>ut_user</i> (login name of the user), <i>ut_line</i> , <i>ut_pid</i> , <i>ut_tv</i>
INIT_PROCESS	<i>ut_id</i> , <i>ut_pid</i> , <i>ut_tv</i>
LOGIN_PROCESS	<i>ut_id</i> , <i>ut_user</i> (implementation-defined name of the login process), <i>ut_line</i> , <i>ut_pid</i> , <i>ut_tv</i>
DEAD_PROCESS	<i>ut_id</i> , <i>ut_pid</i> , <i>ut_tv</i>

An implementation that provides extended security controls may impose implementation-defined restrictions on accessing the user accounting database. In particular, the system may deny the existence of some or all of the user accounting database entries associated with users other than the caller.

If the process has appropriate privileges, the *pututxline()* function shall write out the structure into the user accounting database. It shall search for a record as if by *getutxid()* that satisfies the request. If this search succeeds, then the entry shall be replaced. Otherwise, a new entry shall be made at the end of the user accounting database.

The *endutxent()* function shall close the user accounting database.

The *setutxent()* function shall reset the input to the beginning of the database. This should be done before each search for a new entry if it is desired that the entire database be examined.

These functions need not be thread-safe.

## RETURN VALUE

Upon successful completion, *getutxent()*, *getutxid()*, and *getutxline()* shall return a pointer to a **utmpx** structure containing a copy of the requested entry in the user accounting database. Otherwise, a null pointer shall be returned.

The return value may point to a static area which is overwritten by a subsequent call to *getutxid()* or *getutxline()*.

Upon successful completion, *pututxline()* shall return a pointer to a **utmpx** structure containing a copy of the entry added to the user accounting database. Otherwise, a null pointer shall be returned.

The *endutxent()* and *setutxent()* functions shall not return a value.

## ERRORS

No errors are defined for the *endutxent()*, *getutxent()*, *getutxid()*, *getutxline()*, and *setutxent()* functions.

The *pututxline()* function may fail if:

[EPERM]           The process does not have appropriate privileges.

**26178 EXAMPLES**

26179 None.

**26180 APPLICATION USAGE**

26181 The sizes of the arrays in the structure can be found using the *sizeof* operator.

**26182 RATIONALE**

26183 None.

**26184 FUTURE DIRECTIONS**

26185 None.

**26186 SEE ALSO**

26187 XBD [<utmpx.h>](#)

**26188 CHANGE HISTORY**

26189 First released in Issue 4, Version 2.

**26190 Issue 5**

26191 Moved from X/OPEN UNIX extension to BASE.

26192 Normative text previously in the APPLICATION USAGE section is moved to the  
26193 DESCRIPTION.

26194 A note indicating that these functions need not be reentrant is added to the DESCRIPTION.

**26195 Issue 6**

26196 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

**26197 Issue 7**

26198 Austin Group Interpretation 1003.1-2001 #156 is applied.

26199 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0090 [213,428] and  
26200 XSH/TC1-2008/0091 [213] are applied.

26201 **NAME**

26202        environ — array of character pointers to the environment strings

26203 **SYNOPSIS**

26204        extern char \*\*environ;

26205 **DESCRIPTION**26206        Refer to *exec* and XBD [Chapter 8](#) (on page 173).



26207 **NAME**

26208       erand48 — generate uniformly distributed pseudo-random numbers

26209 **SYNOPSIS**

26210 XSI       #include &lt;stdlib.h&gt;

26211       double erand48(unsigned short xsubi[3]);

26212 **DESCRIPTION**26213       Refer to *drand48()*.

26214 **NAME**

26215 erf, erff, erfl — error functions

26216 **SYNOPSIS**

```
26217 #include <math.h>
26218 double erf(double x);
26219 float erff(float x);
26220 long double erfl(long double x);
```

26221 **DESCRIPTION**

26222 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 26223 conflict between the requirements described here and the ISO C standard is unintentional. This  
 26224 volume of POSIX.1-2008 defers to the ISO C standard.

26225 These functions shall compute the error function of their argument  $x$ , defined as:

$$26226 \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

26227 An application wishing to check for error situations should set *errno* to zero and call  
 26228 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 26229 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 26230 zero, an error has occurred.

26231 **RETURN VALUE**

26232 Upon successful completion, these functions shall return the value of the error function.

26233 MX If  $x$  is NaN, a NaN shall be returned.

26234 If  $x$  is  $\pm 0$ ,  $\pm 0$  shall be returned.

26235 If  $x$  is  $\pm \text{Inf}$ ,  $\pm 1$  shall be returned.

26236 If the correct value would cause underflow, a range error may occur, and *erf()*, *erff()*, and *erfl()*  
 26237 shall return an implementation-defined value no greater in magnitude than DBL\_MIN,  
 26238 FLT\_MIN, and LDBL\_MIN, respectively.

26239 MXX If the IEC 60559 Floating-Point option is supported,  $2 * x / \sqrt{\pi}$  should be returned.

26240 **ERRORS**

26241 These functions may fail if:

26242 MX **Range Error** The result underflows.

26243 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 26244 then *errno* shall be set to [ERANGE]. If the integer expression  
 26245 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 26246 floating-point exception shall be raised.

26247 **EXAMPLES**26248 **Computing the Probability for a Normal Variate**

26249 This example shows how to use *erf()* to compute the probability that a normal variate assumes a  
 26250 value in the range  $[x_1, x_2]$  with  $x_1 \leq x_2$ .

26251 This example uses the constant `M_SQRT1_2` which is part of the XSI option.

```
26252 #include <math.h>
26253
26254 double
26255 Phi(const double x1, const double x2)
26256 {
26257     return ( erf(x2*M_SQRT1_2) - erf(x1*M_SQRT1_2) ) / 2;
```

26258 **APPLICATION USAGE**

26259 Underflow occurs when  $|x| < \text{DBL\_MIN} * (\sqrt{\pi}/2)$ .

26260 On error, the expressions *(math\_errhandling & MATH\_ERRNO)* and *(math\_errhandling &*  
 26261 *MATH\_ERREXCEPT)* are independent of each other, but at least one of them must be non-zero.

26262 **RATIONALE**

26263 None.

26264 **FUTURE DIRECTIONS**

26265 None.

26266 **SEE ALSO**

26267 *erfc()*, *feclearexcept()*, *fetestexcept()*, *isnan()*

26268 XBD [Section 4.20](#) (on page 117), *<math.h>*

26269 **CHANGE HISTORY**

26270 First released in Issue 1. Derived from Issue 1 of the SVID.

26271 **Issue 5**

26272 The DESCRIPTION is updated to indicate how an application should check for an error. This  
 26273 text was previously published in the APPLICATION USAGE section.

26274 **Issue 6**

26275 The *erf()* function is no longer marked as an extension.

26276 The *erfc()* function is split out onto its own reference page.

26277 The *erff()* and *erfl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

26278 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
 26279 revised to align with the ISO/IEC 9899:1999 standard.

26280 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
 26281 marked.

26282 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/22 is applied, adding the example to the  
 26283 EXAMPLES section.

26284 **Issue 7**

26285 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0092 [68] is applied.

26286 **NAME**

26287       erfc, erfcf, erfcl — complementary error functions

26288 **SYNOPSIS**

26289       #include &lt;math.h&gt;

26290       double erfc(double x);

26291       float erfcf(float x);

26292       long double erfcl(long double x);

26293 **DESCRIPTION**

26294 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 26295 conflict between the requirements described here and the ISO C standard is unintentional. This  
 26296 volume of POSIX.1-2008 defers to the ISO C standard.

26297       These functions shall compute the complementary error function  $1.0 - \operatorname{erf}(x)$ .

26298       An application wishing to check for error situations should set *errno* to zero and call  
 26299 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 26300 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 26301 zero, an error has occurred.

26302 **RETURN VALUE**

26303       Upon successful completion, these functions shall return the value of the complementary error  
 26304 function.

26305 MXX       If the correct value would cause underflow, and is not representable, a range error may occur,  
 26306 MXX       and *erfc*(), *erfcf*(), and *erfcl*() shall return 0.0, or (if the IEC 60559 Floating-Point option is not  
 26307 supported) an implementation-defined value no greater in magnitude than DBL\_MIN,  
 26308 FLT\_MIN, and LDBL\_MIN, respectively.

26309 MX       If *x* is NaN, a NaN shall be returned.26310       If *x* is  $\pm 0$ , +1 shall be returned.26311       If *x* is  $-\infty$ , +2 shall be returned.26312       If *x* is  $+\infty$ , +0 shall be returned.

26313 MXX       If the correct value would cause underflow and is representable, a range error may occur and  
 26314 the correct value shall be returned.

26315 **ERRORS**

26316       These functions may fail if:

26317       Range Error       The result underflows.

26318       If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 26319 then *errno* shall be set to [ERANGE]. If the integer expression  
 26320 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 26321 floating-point exception shall be raised.

26322 **EXAMPLES**

26323 None.

26324 **APPLICATION USAGE**

26325 The *erfc()* function is provided because of the extreme loss of relative accuracy if *erf(x)* is called  
 26326 for large *x* and the result subtracted from 1.0.

26327 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* & -  
 26328 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

26329 **RATIONALE**

26330 None.

26331 **FUTURE DIRECTIONS**

26332 None.

26333 **SEE ALSO**26334 *erf()*, *feclearexcept()*, *fetestexcept()*, *isnan()*26335 XBD Section 4.20 (on page 117), **<math.h>**26336 **CHANGE HISTORY**

26337 First released in Issue 1. Derived from Issue 1 of the SVID.

26338 **Issue 5**

26339 The DESCRIPTION is updated to indicate how an application should check for an error. This  
 26340 text was previously published in the APPLICATION USAGE section.

26341 **Issue 6**26342 The *erfc()* function is no longer marked as an extension.26343 These functions are split out from the *erf()* reference page.

26344 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
 26345 revised to align with the ISO/IEC 9899:1999 standard.

26346 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
 26347 marked.

26348 **Issue 7**

26349 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0093 [68] and XSH/TC1-2008/0094  
 26350 [68] are applied.

26351 **NAME**26352 `erff, erfl` — error functions26353 **SYNOPSIS**26354 `#include <math.h>`26355 `float erff(float x);`26356 `long double erfl(long double x);`26357 **DESCRIPTION**26358 Refer to *erf()*.

26359 **NAME**26360 `errno` — error return value26361 **SYNOPSIS**26362 `#include <errno.h>`26363 **DESCRIPTION**26364 The lvalue *errno* is used by many functions to return error values.

26365 Many functions provide an error number in *errno*, which has type **int** and is defined in  
 26366 **<errno.h>**. The value of *errno* shall be defined only after a call to a function for which it is  
 26367 explicitly stated to be set and until it is changed by the next function call or if the application  
 26368 assigns it a value. The value of *errno* should only be examined when it is indicated to be valid by  
 26369 a function's return value. Applications shall obtain the definition of *errno* by the inclusion of  
 26370 **<errno.h>**. No function in this volume of POSIX.1-2008 shall set *errno* to 0. The setting of *errno*  
 26371 after a successful call to a function is unspecified unless the description of that function specifies  
 26372 that *errno* shall not be modified.

26373 It is unspecified whether *errno* is a macro or an identifier declared with external linkage. If a  
 26374 macro definition is suppressed in order to access an actual object, or a program defines an  
 26375 identifier with the name *errno*, the behavior is undefined.

26376 The symbolic values stored in *errno* are documented in the ERRORS sections on all relevant  
 26377 pages.

26378 **RETURN VALUE**

26379 None.

26380 **ERRORS**

26381 None.

26382 **EXAMPLES**

26383 None.

26384 **APPLICATION USAGE**

26385 Previously both POSIX and X/Open documents were more restrictive than the ISO C standard  
 26386 in that they required *errno* to be defined as an external variable, whereas the ISO C standard  
 26387 required only that *errno* be defined as a modifiable lvalue with type **int**.

26388 An application that needs to examine the value of *errno* to determine the error should set it to 0  
 26389 before a function call, then inspect it before a subsequent function call.

26390 **RATIONALE**

26391 None.

26392 **FUTURE DIRECTIONS**

26393 None.

26394 **SEE ALSO**26395 [Section 2.3](#)26396 XBD **<errno.h>**26397 **CHANGE HISTORY**

26398 First released in Issue 1. Derived from Issue 1 of the SVID.

26399 **Issue 5**

26400 The following sentence is deleted from the DESCRIPTION: “The value of *errno* is 0 at program  
 26401 start-up, but is never set to 0 by any XSI function”.

26402 The DESCRIPTION also no longer states that conforming implementations may support the  
26403 declaration:

26404 `extern int errno;`

26405 **Issue 6**

26406 Obsolescent text regarding defining *errno* as:

26407 `extern int errno`

26408 is removed.

26409 Text regarding no function setting *errno* to zero to indicate an error is changed to no function  
26410 shall set *errno* to zero. This is for alignment with the ISO/IEC 9899:1999 standard.

26411 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/23 is applied, adding text to the  
26412 DESCRIPTION stating that the setting of *errno* after a successful call to a function is unspecified  
26413 unless the description of the function requires that it will not be modified.



**NAME**

environ, execl, execlx, execlp, execv, execve, execvp, fexecve — execute a file

**SYNOPSIS**

```
#include <unistd.h>

extern char **environ;

int execl(const char *path, const char *arg0, ... /*, (char *)0 */);
int execlx(const char *path, const char *arg0, ... /*,
    (char *)0, char *const envp[] */);
int execlp(const char *file, const char *arg0, ... /*, (char *)0 */);
int execv(const char *path, char *const argv[]);
int execve(const char *path, char *const argv[], char *const envp[]);
int execvp(const char *file, char *const argv[]);
int fexecve(int fd, char *const argv[], char *const envp[]);
```

**DESCRIPTION**

The *exec* family of functions shall replace the current process image with a new process image. The new image shall be constructed from a regular, executable file called the *new process image file*. There shall be no return from a successful *exec*, because the calling process image is overlaid by the new process image.

The *fexecve()* function shall be equivalent to the *execve()* function except that the file to be executed is determined by the file descriptor *fd* instead of a pathname. The file offset of *fd* is ignored.

When a C-language program is executed as a result of a call to one of the *exec* family of functions, it shall be entered as a C-language function call as follows:

```
int main (int argc, char *argv[]);
```

where *argc* is the argument count and *argv* is an array of character pointers to the arguments themselves. In addition, the following variable, which must be declared by the user if it is to be used directly:

```
extern char **environ;
```

is initialized as a pointer to an array of character pointers to the environment strings. The *argv* and *environ* arrays are each terminated by a null pointer. The null pointer terminating the *argv* array is not counted in *argc*.

Applications can change the entire environment in a single operation by assigning the *environ* variable to point to an array of character pointers to the new environment strings. After assigning a new value to *environ*, applications should not rely on the new environment strings remaining part of the environment, as a call to *getenv()*, *putenv()*, *setenv()*, *unsetenv()*, or any function that is dependent on an environment variable may, on noticing that *environ* has changed, copy the environment strings to a new array and assign *environ* to point to it.

XSI

Any application that directly modifies the pointers to which the *environ* variable points has undefined behavior.

Conforming multi-threaded applications shall not use the *environ* variable to access or modify any environment variable while any other thread is concurrently modifying any environment variable. A call to any function dependent on any environment variable shall be considered a use of the *environ* variable to access that environment variable.

The arguments specified by a program with one of the *exec* functions shall be passed on to the new process image in the corresponding *main()* arguments.

26459 The argument *path* points to a pathname that identifies the new process image file.

26460 The argument *file* is used to construct a pathname that identifies the new process image file. If  
26461 the *file* argument contains a <slash> character, the *file* argument shall be used as the pathname  
26462 for this file. Otherwise, the path prefix for this file is obtained by a search of the directories  
26463 passed as the environment variable *PATH* (see XBD Chapter 8, on page 173). If this environment  
26464 variable is not present, the results of the search are implementation-defined.

26465 There are two distinct ways in which the contents of the process image file may cause the  
26466 execution to fail, distinguished by the setting of *errno* to either [ENOEXEC] or [EINVAL] (see the  
26467 ERRORS section). In the cases where the other members of the *exec* family of functions would  
26468 fail and set *errno* to [ENOEXEC], the *execlp()* and *execvp()* functions shall execute a command  
26469 interpreter and the environment of the executed command shall be as if the process invoked the  
26470 *sh* utility using *execl()* as follows:

26471 `execl(<shell path>, arg0, file, arg1, ..., (char *)0);`

26472 where <shell path> is an unspecified pathname for the *sh* utility, *file* is the process image file, and  
26473 for *execvp()*, where *arg0*, *arg1*, and so on correspond to the values passed to *execvp()* in *argv*[0],  
26474 *argv*[1], and so on.

26475 The arguments represented by *arg0*,... are pointers to null-terminated character strings. These  
26476 strings shall constitute the argument list available to the new process image. The list is  
26477 terminated by a null pointer. The argument *arg0* should point to a filename string that is  
26478 associated with the process being started by one of the *exec* functions.

26479 The argument *argv* is an array of character pointers to null-terminated strings. The application  
26480 shall ensure that the last member of this array is a null pointer. These strings shall constitute the  
26481 argument list available to the new process image. The value in *argv*[0] should point to a filename  
26482 string that is associated with the process being started by one of the *exec* functions.

26483 The argument *envp* is an array of character pointers to null-terminated strings. These strings  
26484 shall constitute the environment for the new process image. The *envp* array is terminated by a  
26485 null pointer.

26486 For those forms not containing an *envp* pointer (*execl()*, *execv()*, *execlp()*, and *execvp()*), the  
26487 environment for the new process image shall be taken from the external variable *environ* in the  
26488 calling process.

26489 The number of bytes available for the new process' combined argument and environment lists is  
26490 {ARG\_MAX}. It is implementation-defined whether null terminators, pointers, and/or any  
26491 alignment bytes are included in this total.

26492 File descriptors open in the calling process image shall remain open in the new process image,  
26493 except for those whose close-on-exec flag FD\_CLOEXEC is set. For those file descriptors that  
26494 remain open, all attributes of the open file description remain unchanged. For any file descriptor  
26495 that is closed for this reason, file locks are removed as a result of the close as described in *close()*.  
26496 Locks that are not removed by closing of file descriptors remain unchanged.

26497 If file descriptor 0, 1, or 2 would otherwise be closed after a successful call to one of the *exec*  
26498 family of functions, implementations may open an unspecified file for the file descriptor in the  
26499 new process image. If a standard utility or a conforming application is executed with file  
26500 descriptor 0 not open for reading or with file descriptor 1 or 2 not open for writing, the  
26501 environment in which the utility or application is executed shall be deemed non-conforming,  
26502 and consequently the utility or application might not behave as described in this standard.

26503 Directory streams open in the calling process image shall be closed in the new process image.

26504		The state of the floating-point environment in the initial thread of the new process image shall
26505		be set to the default.
26506		The state of conversion descriptors and message catalog descriptors in the new process image is
26507		undefined.
26508		For the new process image, the equivalent of:
26509		<code>setlocale(LC_ALL, "C")</code>
26510		shall be executed at start-up.
26511		Signals set to the default action (SIG_DFL) in the calling process image shall be set to the default
26512		action in the new process image. Except for SIGCHLD, signals set to be ignored (SIG_IGN) by
26513		the calling process image shall be set to be ignored by the new process image. Signals set to be
26514		caught by the calling process image shall be set to the default action in the new process image
26515		(see <signal.h>).
26516		If the SIGCHLD signal is set to be ignored by the calling process image, it is unspecified whether
26517		the SIGCHLD signal is set to be ignored or to the default action in the new process image.
26518	XSI	After a successful call to any of the <i>exec</i> functions, alternate signal stacks are not preserved and
26519		the SA_ONSTACK flag shall be cleared for all signals.
26520		After a successful call to any of the <i>exec</i> functions, any functions previously registered by the
26521		<i>atexit()</i> or <i>pthread_atfork()</i> functions are no longer registered.
26522	XSI	If the ST_NOSUID bit is set for the file system containing the new process image file, then the
26523		effective user ID, effective group ID, saved set-user-ID, and saved set-group-ID are unchanged
26524		in the new process image. Otherwise, if the set-user-ID mode bit of the new process image file is
26525		set, the effective user ID of the new process image shall be set to the user ID of the new process
26526		image file. Similarly, if the set-group-ID mode bit of the new process image file is set, the
26527		effective group ID of the new process image shall be set to the group ID of the new process
26528		image file. The real user ID, real group ID, and supplementary group IDs of the new process
26529		image shall remain the same as those of the calling process image. The effective user ID and
26530		effective group ID of the new process image shall be saved (as the saved set-user-ID and the
26531		saved set-group-ID) for use by <i>setuid()</i> .
26532	XSI	Any shared memory segments attached to the calling process image shall not be attached to the
26533		new process image.
26534		Any named semaphores open in the calling process shall be closed as if by appropriate calls to
26535		<i>sem_close()</i> .
26536	TYM	Any blocks of typed memory that were mapped in the calling process are unmapped, as if
26537		<i>munmap()</i> was implicitly called to unmap them.
26538	ML	Memory locks established by the calling process via calls to <i>mlockall()</i> or <i>mlock()</i> shall be
26539		removed. If locked pages in the address space of the calling process are also mapped into the
26540		address spaces of other processes and are locked by those processes, the locks established by the
26541		other processes shall be unaffected by the call by this process to the <i>exec</i> function. If the <i>exec</i>
26542		function fails, the effect on memory locks is unspecified.
26543		Memory mappings created in the process are unmapped before the address space is rebuilt for
26544		the new process image.
26545	SS	When the calling process image does not use the SCHED_FIFO, SCHED_RR, or
26546		SCHED_SPORADIC scheduling policies, the scheduling policy and parameters of the new
26547		process image and the initial thread in that new process image are implementation-defined.

26548	PS	When the calling process image uses the SCHED_FIFO, SCHED_RR, or SCHED_SPORADIC scheduling policies, the process policy and scheduling parameter settings shall not be changed
26549		by a call to an <i>exec</i> function. The initial thread in the new process image shall inherit the process
26550	TPS	scheduling policy and parameters. It shall have the default system contention scope, but shall
26551		inherit its allocation domain from the calling process image.
26552		
26553		Per-process timers created by the calling process shall be deleted before replacing the current
26554		process image with the new process image.
26555	MSG	All open message queue descriptors in the calling process shall be closed, as described in
26556		<i>mq_close()</i> .
26557		Any outstanding asynchronous I/O operations may be canceled. Those asynchronous I/O
26558		operations that are not canceled shall complete as if the <i>exec</i> function had not yet occurred, but
26559		any associated signal notifications shall be suppressed. It is unspecified whether the <i>exec</i>
26560		function itself blocks awaiting such I/O completion. In no event, however, shall the new process
26561		image created by the <i>exec</i> function be affected by the presence of outstanding asynchronous I/O
26562		operations at the time the <i>exec</i> function is called. Whether any I/O is canceled, and which I/O
26563		may be canceled upon <i>exec</i> , is implementation-defined.
26564	CPT	The new process image shall inherit the CPU-time clock of the calling process image. This
26565		inheritance means that the process CPU-time clock of the process being <i>exec</i> -ed shall not be
26566		reinitialized or altered as a result of the <i>exec</i> function other than to reflect the time spent by the
26567		process executing the <i>exec</i> function itself.
26568	TCT	The initial value of the CPU-time clock of the initial thread of the new process image shall be set
26569		to zero.
26570	OB TRC	If the calling process is being traced, the new process image shall continue to be traced into the
26571		same trace stream as the original process image, but the new process image shall not inherit the
26572		mapping of trace event names to trace event type identifiers that was defined by calls to the
26573		<i>posix_trace_eventid_open()</i> or the <i>posix_trace_trid_eventid_open()</i> functions in the calling process
26574		image.
26575		If the calling process is a trace controller process, any trace streams that were created by the
26576		calling process shall be shut down as described in the <i>posix_trace_shutdown()</i> function.
26577		The thread ID of the initial thread in the new process image is unspecified.
26578		The size and location of the stack on which the initial thread in the new process image runs is
26579		unspecified.
26580		The initial thread in the new process image shall have its cancellation type set to
26581		PTHREAD_CANCEL_DEFERRED and its cancellation state set to
26582		PTHREAD_CANCEL_ENABLED.
26583		The initial thread in the new process image shall have all thread-specific data values set to
26584		NULL and all thread-specific data keys shall be removed by the call to <i>exec</i> without running
26585		destructors.
26586		The initial thread in the new process image shall be joinable, as if created with the <i>detachstate</i>
26587		attribute set to PTHREAD_CREATE_JOINABLE.
26588		The new process shall inherit at least the following attributes from the calling process image:
26589	XSI	<ul style="list-style-type: none"> <li>Nice value (see <i>nice()</i>)</li> </ul>

26590	XSI	<ul style="list-style-type: none"> <li>• <i>semadj</i> values (see <i>semop()</i>)</li> </ul>
26591		<ul style="list-style-type: none"> <li>• Process ID</li> </ul>
26592		<ul style="list-style-type: none"> <li>• Parent process ID</li> </ul>
26593		<ul style="list-style-type: none"> <li>• Process group ID</li> </ul>
26594		<ul style="list-style-type: none"> <li>• Session membership</li> </ul>
26595		<ul style="list-style-type: none"> <li>• Real user ID</li> </ul>
26596		<ul style="list-style-type: none"> <li>• Real group ID</li> </ul>
26597		<ul style="list-style-type: none"> <li>• Supplementary group IDs</li> </ul>
26598		<ul style="list-style-type: none"> <li>• Time left until an alarm clock signal (see <i>alarm()</i>)</li> </ul>
26599		<ul style="list-style-type: none"> <li>• Current working directory</li> </ul>
26600		<ul style="list-style-type: none"> <li>• Root directory</li> </ul>
26601		<ul style="list-style-type: none"> <li>• File mode creation mask (see <i>umask()</i>)</li> </ul>
26602	XSI	<ul style="list-style-type: none"> <li>• File size limit (see <i>getrlimit()</i> and <i>setrlimit()</i>)</li> </ul>
26603		<ul style="list-style-type: none"> <li>• Process signal mask (see <i>pthread_sigmask()</i>)</li> </ul>
26604		<ul style="list-style-type: none"> <li>• Pending signal (see <i>sigpending()</i>)</li> </ul>
26605		<ul style="list-style-type: none"> <li>• <i>tms_utime</i>, <i>tms_stime</i>, <i>tms_cutime</i>, and <i>tms_cstime</i> (see <i>times()</i>)</li> </ul>
26606	XSI	<ul style="list-style-type: none"> <li>• Resource limits</li> </ul>
26607		<ul style="list-style-type: none"> <li>• Controlling terminal</li> </ul>
26608	XSI	<ul style="list-style-type: none"> <li>• Interval timers</li> </ul>
26609		The initial thread of the new process shall inherit at least the following attributes from the
26610		calling thread:
26611		<ul style="list-style-type: none"> <li>• Signal mask (see <i>sigprocmask()</i> and <i>pthread_sigmask()</i>)</li> </ul>
26612		<ul style="list-style-type: none"> <li>• Pending signals (see <i>sigpending()</i>)</li> </ul>
26613		All other process attributes defined in this volume of POSIX.1-2008 shall be inherited in the new
26614		process image from the old process image. All other thread attributes defined in this volume of
26615		POSIX.1-2008 shall be inherited in the initial thread in the new process image from the calling
26616		thread in the old process image. The inheritance of process or thread attributes not defined by
26617		this volume of POSIX.1-2008 is implementation-defined.
26618		A call to any <i>exec</i> function from a process with more than one thread shall result in all threads
26619		being terminated and the new executable image being loaded and executed. No destructor
26620		functions or cleanup handlers shall be called.
26621		Upon successful completion, the <i>exec</i> functions shall mark for update the last data access
26622		timestamp of the file. If an <i>exec</i> function failed but was able to locate the process image file,
26623		whether the last data access timestamp is marked for update is unspecified. Should the <i>exec</i>
26624		function succeed, the process image file shall be considered to have been opened with <i>open()</i> .
26625		The corresponding <i>close()</i> shall be considered to occur at a time after this open, but before
26626		process termination or successful completion of a subsequent call to one of the <i>exec</i> functions,
26627		<i>posix_spawn()</i> , or <i>posix_spawnnp()</i> . The <i>argv[]</i> and <i>envp[]</i> arrays of pointers and the strings to
26628		which those arrays point shall not be modified by a call to one of the <i>exec</i> functions, except as a
26629		consequence of replacing the process image.

26630 XSI The saved resource limits in the new process image are set to be a copy of the process'  
26631 corresponding hard and soft limits.

## 26632 RETURN VALUE

26633 If one of the *exec* functions returns to the calling process image, an error has occurred; the return  
26634 value shall be -1, and *errno* shall be set to indicate the error.

## 26635 ERRORS

26636 The *exec* functions shall fail if:

26637 [E2BIG] The number of bytes used by the new process image's argument list and  
26638 environment list is greater than the system-imposed limit of {ARG\_MAX}  
26639 bytes.

26640 [EACCES] The new process image file is not a regular file and the implementation does  
26641 not support execution of files of its type.

26642 [EINVAL] The new process image file has appropriate privileges and has a recognized  
26643 executable binary format, but the system does not support execution of a file  
26644 with this format.

26645 The *exec* functions, except for *fexecve()*, shall fail if:

26646 [EACCES] Search permission is denied for a directory listed in the new process image  
26647 file's path prefix, or the new process image file denies execution permission.

26648 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path* or *file*  
26649 argument.

26650 [ENAMETOOLONG]

26651 The length of a component of a pathname is longer than {NAME\_MAX}.

26652 [ENOENT] A component of *path* or *file* does not name an existing file or *path* or *file* is an  
26653 empty string.

26654 [ENOTDIR] A component of the new process image file's path prefix names an existing file  
26655 that is neither a directory nor a symbolic link to a directory, or the new process  
26656 image file's pathname contains at least one non-*<slash>* character and ends  
26657 with one or more trailing *<slash>* characters and the last pathname  
26658 component names an existing file that is neither a directory nor a symbolic  
26659 link to a directory.

26660 The *exec* functions, except for *execlp()* and *execvp()*, shall fail if:

26661 [ENOEXEC] The new process image file has the appropriate access permission but has an  
26662 unrecognized format.

26663 The *fexecve()* function shall fail if:

26664 [EBADF] The *fd* argument is not a valid file descriptor open for executing.

26665 The *exec* functions may fail if:

26666 [ENOMEM] The new process image requires more memory than is allowed by the  
26667 hardware or system-imposed memory management constraints.

26668 The *exec* functions, except for *fexecve()*, may fail if:

26669 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
26670 resolution of the *path* or *file* argument.



26671 [ENAMETOOLONG]  
 26672 The length of the *path* argument or the length of the pathname constructed  
 26673 from the *file* argument exceeds {PATH\_MAX}, or pathname resolution of a  
 26674 symbolic link produced an intermediate result with a length that exceeds  
 26675 {PATH\_MAX}.

26676 [ETXTBSY] The new process image file is a pure procedure (shared text) file that is  
 26677 currently open for writing by some process.

## 26678 EXAMPLES

### 26679 Using `execl()`

26680 The following example executes the *ls* command, specifying the pathname of the executable  
 26681 (*/bin/ls*) and using arguments supplied directly to the command to produce single-column  
 26682 output.

```
26683 #include <unistd.h>
26684 int ret;
26685 ...
26686 ret = execl ("/bin/ls", "ls", "-l", (char *)0);
```

### 26687 Using `execle()`

26688 The following example is similar to [Using `execl\(\)`](#). In addition, it specifies the environment for  
 26689 the new process image using the *env* argument.

```
26690 #include <unistd.h>
26691 int ret;
26692 char *env[] = { "HOME=/usr/home", "LOGNAME=home", (char *)0 };
26693 ...
26694 ret = execle ("/bin/ls", "ls", "-l", (char *)0, env);
```

### 26695 Using `execlp()`

26696 The following example searches for the location of the *ls* command among the directories  
 26697 specified by the *PATH* environment variable.

```
26698 #include <unistd.h>
26699 int ret;
26700 ...
26701 ret = execlp ("ls", "ls", "-l", (char *)0);
```

### 26702 Using `execv()`

26703 The following example passes arguments to the *ls* command in the *cmd* array.

```
26704 #include <unistd.h>
26705 int ret;
26706 char *cmd[] = { "ls", "-l", (char *)0 };
26707 ...
26708 ret = execv ("/bin/ls", cmd);
```

### Using `execve()`

The following example passes arguments to the `ls` command in the `cmd` array, and specifies the environment for the new process image using the `env` argument.

```
#include <unistd.h>

int ret;
char *cmd[] = { "ls", "-l", (char *)0 };
char *env[] = { "HOME=/usr/home", "LOGNAME=home", (char *)0 };
...
ret = execve ("/bin/ls", cmd, env);
```

### Using `execvp()`

The following example searches for the location of the `ls` command among the directories specified by the `PATH` environment variable, and passes arguments to the `ls` command in the `cmd` array.

```
#include <unistd.h>

int ret;
char *cmd[] = { "ls", "-l", (char *)0 };
...
ret = execvp ("ls", cmd);
```

## APPLICATION USAGE

As the state of conversion descriptors and message catalog descriptors in the new process image is undefined, conforming applications should not rely on their use and should close them prior to calling one of the `exec` functions.

Applications that require other than the default POSIX locale as the global locale in the new process image should call `setlocale()` with the appropriate parameters.

When assigning a new value to the `environ` variable, applications should ensure that the environment to which it will point contains at least the following:

1. Any implementation-defined variables required by the implementation to provide a conforming environment. See the `_CS_V7_ENV` entry in `<unistd.h>` and `confstr()` for details.
2. A value for `PATH` which finds conforming versions of all standard utilities before any other versions.

The same constraint applies to the `envp` array passed to `execle()` or `execve()`, in order to ensure that the new process image is invoked in a conforming environment.

Applications should not execute programs with file descriptor 0 not open for reading or with file descriptor 1 or 2 not open for writing, as this might cause the executed program to misbehave. In order not to pass on these file descriptors to an executed program, applications should not just close them but should reopen them on, for example, `/dev/null`. Some implementations may reopen them automatically, but applications should not rely on this being done.

If an application wants to perform a checksum test of the file being executed before executing it, the file will need to be opened with read permission to perform the checksum test.

Since execute permission is checked by `fexecve()`, the file description `fd` need not have been opened with the `O_EXEC` flag. However, if the file to be executed denies read and write permission for the process preparing to do the `exec`, the only way to provide the `fd` to `fexecve()`



will be to use the `O_EXEC` flag when opening *fd*. In this case, the application will not be able to perform a checksum test since it will not be able to read the contents of the file.

Note that when a file descriptor is opened with `O_RDONLY`, `O_RDWR`, or `O_WRONLY` mode, the file descriptor can be used to read, read and write, or write the file, respectively, even if the mode of the file changes after the file was opened. Using the `O_EXEC` open mode is different; *fexecve()* will ignore the mode that was used when the file descriptor was opened and the *exec* will fail if the mode of the file associated with *fd* does not grant execute permission to the calling process at the time *fexecve()* is called.

## RATIONALE

Early proposals required that the value of *argc* passed to *main()* be “one or greater”. This was driven by the same requirement in drafts of the ISO C standard. In fact, historical implementations have passed a value of zero when no arguments are supplied to the caller of the *exec* functions. This requirement was removed from the ISO C standard and subsequently removed from this volume of POSIX.1-2008 as well. The wording, in particular the use of the word *should*, requires a Strictly Conforming POSIX Application to pass at least one argument to the *exec* function, thus guaranteeing that *argc* be one or greater when invoked by such an application. In fact, this is good practice, since many existing applications reference *argv[0]* without first checking the value of *argc*.

The requirement on a Strictly Conforming POSIX Application also states that the value passed as the first argument be a filename string associated with the process being started. Although some existing applications pass a pathname rather than a filename string in some circumstances, a filename string is more generally useful, since the common usage of *argv[0]* is in printing diagnostics. In some cases the filename passed is not the actual filename of the file; for example, many implementations of the *login* utility use a convention of prefixing a <hyphen-minus> (‘-’) to the actual filename, which indicates to the command interpreter being invoked that it is a “login shell”.

Also, note that the *test* and *[* utilities require specific strings for the *argv[0]* argument to have deterministic behavior across all implementations. +

Historically, there have been two ways that implementations can *exec* shell scripts. +

One common historical implementation is that the *execl()*, *execv()*, *execle()*, and *execve()* functions return an `[ENOEXEC]` error for any file not recognizable as executable, including a shell script. When the *execlp()* and *execvp()* functions encounter such a file, they assume the file to be a shell script and invoke a known command interpreter to interpret such files. This is now required by POSIX.1-2008. These implementations of *execvp()* and *execlp()* only give the `[ENOEXEC]` error in the rare case of a problem with the command interpreter’s executable file. Because of these implementations, the `[ENOEXEC]` error is not mentioned for *execlp()* or *execvp()*, although implementations can still give it.

Another way that some historical implementations handle shell scripts is by recognizing the first two bytes of the file as the character string “#!” and using the remainder of the first line of the file as the name of the command interpreter to execute.

One potential source of confusion noted by the standard developers is over how the contents of a process image file affect the behavior of the *exec* family of functions. The following is a description of the actions taken:

1. If the process image file is a valid executable (in a format that is executable and valid and having appropriate privileges) for this system, then the system executes the file.

2. If the process image file has appropriate privileges and is in a format that is executable but not valid for this system (such as a recognized binary for another architecture), then this is an error and *errno* is set to [EINVAL] (see later RATIONALE on [EINVAL]).
3. If the process image file has appropriate privileges but is not otherwise recognized:
  - a. If this is a call to *execlp()* or *execvp()*, then they invoke a command interpreter assuming that the process image file is a shell script.
  - b. If this is not a call to *execlp()* or *execvp()*, then an error occurs and *errno* is set to [ENOEXEC].

Applications that do not require to access their arguments may use the form:

```
main(void)
```

as specified in the ISO C standard. However, the implementation will always provide the two arguments *argc* and *argv*, even if they are not used.

Some implementations provide a third argument to *main()* called *envp*. This is defined as a pointer to the environment. The ISO C standard specifies invoking *main()* with two arguments, so implementations must support applications written this way. Since this volume of POSIX.1-2008 defines the global variable *environ*, which is also provided by historical implementations and can be used anywhere that *envp* could be used, there is no functional need for the *envp* argument. Applications should use the *getenv()* function rather than accessing the environment directly via either *envp* or *environ*. Implementations are required to support the two-argument calling sequence, but this does not prohibit an implementation from supporting *envp* as an optional third argument.

This volume of POSIX.1-2008 specifies that signals set to SIG\_IGN remain set to SIG\_IGN, and that the new process image inherits the signal mask of the thread that called *exec* in the old process image. This is consistent with historical implementations, and it permits some useful functionality, such as the *nohup* command. However, it should be noted that many existing applications wrongly assume that they start with certain signals set to the default action and/or unblocked. In particular, applications written with a simpler signal model that does not include blocking of signals, such as the one in the ISO C standard, may not behave properly if invoked with some signals blocked. Therefore, it is best not to block or ignore signals across *execs* without explicit reason to do so, and especially not to block signals across *execs* of arbitrary (not closely cooperating) programs.

The *exec* functions always save the value of the effective user ID and effective group ID of the process at the completion of the *exec*, whether or not the set-user-ID or the set-group-ID bit of the process image file is set.

The statement about *argv[]* and *envp[]* being constants is included to make explicit to future writers of language bindings that these objects are completely constant. Due to a limitation of the ISO C standard, it is not possible to state that idea in standard C. Specifying two levels of *const-qualification* for the *argv[]* and *envp[]* parameters for the *exec* functions may seem to be the natural choice, given that these functions do not modify either the array of pointers or the characters to which the function points, but this would disallow existing correct code. Instead, only the array of pointers is noted as constant. The table of assignment compatibility for *dst=src* derived from the ISO C standard summarizes the compatibility:

<i>dst:</i>	<b>char *[]</b>	<b>const char *[]</b>	<b>char *const[]</b>	<b>const char *const[]</b>
<i>src:</i>				
<b>char *[]</b>	VALID	—	VALID	—
<b>const char *[]</b>	—	VALID	—	VALID
<b>char * const []</b>	—	—	VALID	—
<b>const char *const[]</b>	—	—	—	VALID

Since all existing code has a source type matching the first row, the column that gives the most valid combinations is the third column. The only other possibility is the fourth column, but using it would require a cast on the *argv* or *envp* arguments. It is unfortunate that the fourth column cannot be used, because the declaration a non-expert would naturally use would be that in the second row.

The ISO C standard and this volume of POSIX.1-2008 do not conflict on the use of *environ*, but some historical implementations of *environ* may cause a conflict. As long as *environ* is treated in the same way as an entry point (for example, *fork()*), it conforms to both standards. A library can contain *fork()*, but if there is a user-provided *fork()*, that *fork()* is given precedence and no problem ensues. The situation is similar for *environ*: the definition in this volume of POSIX.1-2008 is to be used if there is no user-provided *environ* to take precedence. At least three implementations are known to exist that solve this problem.

[E2BIG] The limit {ARG\_MAX} applies not just to the size of the argument list, but to the sum of that and the size of the environment list.

[EFAULT] Some historical systems return [EFAULT] rather than [ENOEXEC] when the new process image file is corrupted. They are non-conforming.

[EINVAL] This error condition was added to POSIX.1-2008 to allow an implementation to detect executable files generated for different architectures, and indicate this situation to the application. Historical implementations of shells, *execvp()*, and *execlp()* that encounter an [ENOEXEC] error will execute a shell on the assumption that the file is a shell script. This will not produce the desired effect when the file is a valid executable for a different architecture. An implementation may now choose to avoid this problem by returning [EINVAL] when a valid executable for a different architecture is encountered. Some historical implementations return [EINVAL] to indicate that the *path* argument contains a character with the high order bit set. The standard developers chose to deviate from historical practice for the following reasons:

1. The new utilization of [EINVAL] will provide some measure of utility to the user community.
2. Historical use of [EINVAL] is not acceptable in an internationalized operating environment.

[ENAMETOOLONG]

Since the file pathname may be constructed by taking elements in the *PATH* variable and putting them together with the filename, the [ENAMETOOLONG] error condition could also be reached this way.

[ETXTBSY] System V returns this error when the executable file is currently open for writing by some process. This volume of POSIX.1-2008 neither requires nor prohibits this behavior.

Other systems (such as System V) may return [EINTR] from *exec*. This is not addressed by this volume of POSIX.1-2008, but implementations may have a window between the call to *exec* and

the time that a signal could cause one of the *exec* calls to return with [EINTR].

An explicit statement regarding the floating-point environment (as defined in the `<fenv.h>` header) was added to make it clear that the floating-point environment is set to its default when a call to one of the *exec* functions succeeds. The requirements for inheritance or setting to the default for other process and thread start-up functions is covered by more generic statements in their descriptions and can be summarized as follows:

*posix\_spawn()* Set to default.

*fork()* Inherit.

*pthread\_create()* Inherit.

The purpose of the *fexecve()* function is to enable executing a file which has been verified to be the intended file. It is possible to actively check the file by reading from the file descriptor and be sure that the file is not exchanged for another between the reading and the execution. Alternatively, a function like *openat()* can be used to open a file which has been found by reading the content of a directory using *readdir()*.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*alarm()*, *atexit()*, *chmod()*, *close()*, *confstr()*, *exit()*, *fcntl()*, *fork()*, *fstatvfs()*, *getenv()*, *getitimer()*, *getrlimit()*, *mknod()*, *mmap()*, *nice()*, *open()*, *posix\_spawn()*, *posix\_trace\_create()*, *posix\_trace\_event()*, *posix\_trace\_eventid\_equal()*, *pthread\_atfork()*, *pthread\_sigmask()*, *putenv()*, *readdir()*, *semop()*, *setlocale()*, *shmat()*, *sigaction()*, *sigaltstack()*, *sigpending()*, *system()*, *times()*, *ulimit()*, *umask()*

XBD Chapter 8 (on page 173), `<unistd.h>`

XCU *test*

## CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

### Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

Large File Summit extensions are added.

### Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, behavior is defined for when the process image file is not a valid executable.
- In this version, `_POSIX_SAVED_IDS` is mandated, thus the effective user ID and effective group ID of the new process image shall be saved (as the saved set-user-ID and the saved set-group-ID) for use by the *setuid()* function.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.
- The [ETXTBSY] optional error condition is added.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [EINVAL] mandatory error condition is added.
- The [ELOOP] optional error condition is added.

The description of CPU-time clock semantics is added for alignment with IEEE Std 1003.1d-1999.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding semantics for typed memory.

The normative text is updated to avoid use of the term “must” for application requirements.

The description of tracing semantics is added for alignment with IEEE Std 1003.1q-2000.

IEEE PASC Interpretation 1003.1 #132 is applied.

The DESCRIPTION is updated to make it explicit that the floating-point environment in the new process image is set to the default.

The DESCRIPTION and RATIONALE are updated to include clarifications of how the contents of a process image file affect the behavior of the *exec* functions.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/15 is applied, adding a new paragraph to the DESCRIPTION and text to the end of the APPLICATION USAGE section. This change addresses a security concern, where implementations may want to reopen file descriptors 0, 1, and 2 for programs with the set-user-id or set-group-id file mode bits calling the *exec* family of functions.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/24 is applied, applying changes to the DESCRIPTION, addressing which attributes are inherited by threads, and behavioral requirements for threads attributes.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/25 is applied, updating text in the RATIONALE from “the process signal mask be unchanged across an *exec*” to “the new process image inherits the signal mask of the thread that called *exec* in the old process image”.

## Issue 7

Austin Group Interpretation 1003.1-2001 #047 is applied, adding the description of `_CS_V7_ENV` to the APPLICATION USAGE.

Austin Group Interpretation 1003.1-2001 #143 is applied.

The *fexecve()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Functionality relating to the Asynchronous Input and Output, Memory Mapped Files, Threads, and Timers options is moved to the Base.

Changes are made related to support for finegrained timestamps.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0095 [386], XSH/TC1-2008/0096 [167], XSH/TC1-2008/0097 [291], XSH/TC1-2008/0098 [173], XSH/TC1-2008/0099 [296], XSH/TC1-2008/00100 [324], XSH/TC1-2008/00101 [296], XSH/TC1-2008/00102 [302], XSH/TC1-2008/00103 [167], XSH/TC1-2008/00104 [173], and XSH/TC1-2008/00105 [291,429] are applied.

26964 **NAME**

26965 exit — terminate a process

26966 **SYNOPSIS**

26967 #include &lt;stdlib.h&gt;

26968 void exit(int status);

26969 **DESCRIPTION**

26970 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 26971 conflict between the requirements described here and the ISO C standard is unintentional. This  
 26972 volume of POSIX.1-2008 defers to the ISO C standard.

26973 CX The value of *status* may be 0, EXIT\_SUCCESS, EXIT\_FAILURE, or any other value, though only  
 26974 the least significant 8 bits (that is, *status* & 0377) shall be available from *wait()* and *waitpid()*; the  
 26975 full value shall be available from *waitid()* and in the **siginfo\_t** passed to a signal handler for  
 26976 SIGCHLD.

26977 The *exit()* function shall first call all functions registered by *atexit()*, in the reverse order of their  
 26978 registration, except that a function is called after any previously registered functions that had  
 26979 already been called at the time it was registered. Each function is called as many times as it was  
 26980 registered. If, during the call to any such function, a call to the *longjmp()* function is made that  
 26981 would terminate the call to the registered function, the behavior is undefined.

26982 If a function registered by a call to *atexit()* fails to return, the remaining registered functions shall  
 26983 not be called and the rest of the *exit()* processing shall not be completed. If *exit()* is called more  
 26984 than once, the behavior is undefined.

26985 The *exit()* function shall then flush all open streams with unwritten buffered data and close all  
 26986 CX open streams. Finally, the process shall be terminated with the same consequences as described  
 26987 in [Consequences of Process Termination](#) (on page 553).

26988 **RETURN VALUE**26989 The *exit()* function does not return.26990 **ERRORS**

26991 No errors are defined.

26992 **EXAMPLES**

26993 None.

26994 **APPLICATION USAGE**

26995 None.

26996 **RATIONALE**26997 See *\_Exit()*.26998 **FUTURE DIRECTIONS**

26999 None.

27000 **SEE ALSO**27001 [\\_Exit\(\)](#), [atexit\(\)](#), [exec](#), [longjmp\(\)](#), [tmpfile\(\)](#), [wait\(\)](#), [waitid\(\)](#)27002 XBD [<stdlib.h>](#)27003 **CHANGE HISTORY**



27004 **Issue 7**

27005 Austin Group Interpretation 1003.1-2001 #031 is applied, separating the `_Exit()` and `_exit()`  
27006 functions from the `exit()` function.

27007 Austin Group Interpretation 1003.1-2001 #085 is applied.

27008 **NAME**27009 `exp`, `expf`, `expl` — exponential function27010 **SYNOPSIS**

```
27011 #include <math.h>
27012 double exp(double x);
27013 float expf(float x);
27014 long double expl(long double x);
```

27015 **DESCRIPTION**

27016 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 27017 conflict between the requirements described here and the ISO C standard is unintentional. This  
 27018 volume of POSIX.1-2008 defers to the ISO C standard.

27019 These functions shall compute the base-*e* exponential of *x*.

27020 An application wishing to check for error situations should set *errno* to zero and call  
 27021 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 27022 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 27023 zero, an error has occurred.

27024 **RETURN VALUE**

27025 Upon successful completion, these functions shall return the exponential value of *x*.

27026 If the correct value would cause overflow, a range error shall occur and *exp()*, *expf()*, and *expl()*  
 27027 shall return the value of the macro HUGE\_VAL, HUGE\_VALF, and HUGE\_VALL, respectively.

27028 MXX If the correct value would cause underflow, and is not representable, a range error may occur,  
 27029 MXX and *exp()*, *expf()*, and *expl()* shall return 0.0, or (if the IEC 60559 Floating-Point option is not  
 27030 supported) an implementation-defined value no greater in magnitude than DBL\_MIN,  
 27031 FLT\_MIN, and LDBL\_MIN, respectively.

27032 MX If *x* is NaN, a NaN shall be returned.

27033 If *x* is  $\pm 0$ , 1 shall be returned.

27034 If *x* is  $-\text{Inf}$ , +0 shall be returned.

27035 If *x* is  $+\text{Inf}$ , *x* shall be returned.

27036 MXX If the correct value would cause underflow, and is representable, a range error may occur and  
 27037 the correct value shall be returned.

27038 **ERRORS**

27039 These functions shall fail if:

27040 Range Error The result overflows.

27041 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 27042 then *errno* shall be set to [ERANGE]. If the integer expression  
 27043 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the overflow  
 27044 floating-point exception shall be raised.

27045 These functions may fail if:

27046 Range Error The result underflows.

27047 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 27048 then *errno* shall be set to [ERANGE]. If the integer expression  
 27049 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 27050 floating-point exception shall be raised.



27051 **EXAMPLES**27052 **Computing the Density of the Standard Normal Distribution**

27053 This function shows an implementation for the density of the standard normal distribution  
 27054 using `exp()`. This example uses the constant `M_PI` which is part of the XSI option.

```
27055 #include <math.h>
27056 double
27057 normal_density (double x)
27058 {
27059     return exp(-x*x/2) / sqrt (2*M_PI);
27060 }
```

27061 **APPLICATION USAGE**

27062 On error, the expressions `(math_errhandling & MATH_ERRNO)` and `(math_errhandling & -`  
 27063 `MATH_ERREXCEPT)` are independent of each other, but at least one of them must be non-zero.

27064 **RATIONALE**

27065 None.

27066 **FUTURE DIRECTIONS**

27067 None.

27068 **SEE ALSO**

27069 [\*feclearexcept\(\)\*](#), [\*fetestexcept\(\)\*](#), [\*isnan\(\)\*](#), [\*log\(\)\*](#)

27070 XBD [Section 4.20](#) (on page 117), [\*\*<math.h>\*\*](#)

27071 **CHANGE HISTORY**

27072 First released in Issue 1. Derived from Issue 1 of the SVID.

27073 **Issue 5**

27074 The DESCRIPTION is updated to indicate how an application should check for an error. This  
 27075 text was previously published in the APPLICATION USAGE section.

27076 **Issue 6**

27077 The `expf()` and `expl()` functions are added for alignment with the ISO/IEC 9899:1999 standard.

27078 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
 27079 revised to align with the ISO/IEC 9899:1999 standard.

27080 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
 27081 marked.

27082 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/26 is applied, adding the example to the  
 27083 EXAMPLES section.

27084 **Issue 7**

27085 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0106 [68] and XSH/TC1-2008/0107  
 27086 [68] are applied.

27087 **NAME**

27088 exp2, exp2f, exp2l — exponential base 2 functions

27089 **SYNOPSIS**

```
27090 #include <math.h>
27091 double exp2(double x);
27092 float exp2f(float x);
27093 long double exp2l(long double x);
```

27094 **DESCRIPTION**

27095 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 27096 conflict between the requirements described here and the ISO C standard is unintentional. This  
 27097 volume of POSIX.1-2008 defers to the ISO C standard.

27098 These functions shall compute the base-2 exponential of  $x$ .

27099 An application wishing to check for error situations should set *errno* to zero and call  
 27100 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 27101 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 27102 zero, an error has occurred.

27103 **RETURN VALUE**

27104 Upon successful completion, these functions shall return  $2^x$ .

27105 If the correct value would cause overflow, a range error shall occur and *exp2()*, *exp2f()*, and  
 27106 *exp2l()* shall return the value of the macro HUGE\_VAL, HUGE\_VALF, and HUGE\_VALL,  
 27107 respectively.

27108 MXX If the correct value would cause underflow, and is not representable, a range error may occur,  
 27109 MXX and *exp2()*, *exp2f()*, and *exp2l()* shall return 0.0, or (if the IEC 60559 Floating-Point option is not  
 27110 supported) an implementation-defined value no greater in magnitude than DBL\_MIN,  
 27111 FLT\_MIN, and LDBL\_MIN, respectively.

27112 MX If  $x$  is NaN, a NaN shall be returned.

27113 If  $x$  is  $\pm 0$ , 1 shall be returned.

27114 If  $x$  is  $-\text{Inf}$ , +0 shall be returned.

27115 If  $x$  is  $+\text{Inf}$ ,  $x$  shall be returned.

27116 MXX If the correct value would cause underflow, and is representable, a range error may occur and  
 27117 the correct value shall be returned.

27118 **ERRORS**

27119 These functions shall fail if:

27120 Range Error The result overflows.

27121 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 27122 then *errno* shall be set to [ERANGE]. If the integer expression  
 27123 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the overflow  
 27124 floating-point exception shall be raised.

27125 These functions may fail if:

27126 Range Error The result underflows.

27127 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 27128 then *errno* shall be set to [ERANGE]. If the integer expression  
 27129 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow

27130 floating-point exception shall be raised.

27131 **EXAMPLES**

27132 None.

27133 **APPLICATION USAGE**

27134 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* & -  
27135 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

27136 **RATIONALE**

27137 None.

27138 **FUTURE DIRECTIONS**

27139 None.

27140 **SEE ALSO**

27141 *exp()*, *feclearexcept()*, *fetestexcept()*, *isnan()*, *log()*

27142 XBD Section 4.20 (on page 117), **<math.h>**

27143 **CHANGE HISTORY**

27144 First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

27145 **Issue 7**

27146 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0108 [68] and XSH/TC1-2008/0109  
27147 [68] are applied.

27148 **NAME**

27149 expm1, expm1f, expm1l — compute exponential functions

27150 **SYNOPSIS**

```
27151 #include <math.h>
27152 double expm1(double x);
27153 float expm1f(float x);
27154 long double expm1l(long double x);
```

27155 **DESCRIPTION**

27156 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 27157 conflict between the requirements described here and the ISO C standard is unintentional. This  
 27158 volume of POSIX.1-2008 defers to the ISO C standard.

27159 These functions shall compute  $e^x - 1.0$ .

27160 An application wishing to check for error situations should set *errno* to zero and call  
 27161 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 27162 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 27163 zero, an error has occurred.

27164 **RETURN VALUE**27165 Upon successful completion, these functions return  $e^x - 1.0$ .

27166 If the correct value would cause overflow, a range error shall occur and *expm1()*, *expm1f()*, and  
 27167 *expm1l()* shall return the value of the macro HUGE\_VAL, HUGE\_VALF, and HUGE\_VALL,  
 27168 respectively.

27169 MX If *x* is NaN, a NaN shall be returned.27170 If *x* is  $\pm 0$ ,  $\pm 0$  shall be returned.27171 If *x* is  $-\text{Inf}$ ,  $-1$  shall be returned.27172 If *x* is  $+\text{Inf}$ , *x* shall be returned.

27173 If *x* is subnormal, a range error may occur  
 27174 MXX and *x* should be returned.

27175 MX If *x* is not returned, *expm1()*, *expm1f()*, and *expm1l()* shall return an implementation-defined  
 27176 value no greater in magnitude than DBL\_MIN, FLT\_MIN, and LDBL\_MIN, respectively.

27177 **ERRORS**

27178 These functions shall fail if:

27179 Range Error The result overflows.

27180 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 27181 then *errno* shall be set to [ERANGE]. If the integer expression  
 27182 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the overflow  
 27183 floating-point exception shall be raised.

27184 These functions may fail if:

27185 MX Range Error The value of *x* is subnormal.

27186 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 27187 then *errno* shall be set to [ERANGE]. If the integer expression  
 27188 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 27189 floating-point exception shall be raised.

**EXAMPLES**

None.

**APPLICATION USAGE**

The value of  $\text{expm1}(x)$  may be more accurate than  $\exp(x)-1.0$  for small values of  $x$ .

The  $\text{expm1}()$  and  $\log1p()$  functions are useful for financial calculations of  $((1+x)^n-1)/x$ , namely:

$\text{expm1}(n * \log1p(x)) / x$

when  $x$  is very small (for example, when calculating small daily interest rates). These functions also simplify writing accurate inverse hyperbolic functions.

On error, the expressions  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$  and  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$  are independent of each other, but at least one of them must be non-zero.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[\*exp\(\)\*](#), [\*feclearexcept\(\)\*](#), [\*fetestexcept\(\)\*](#), [\*ilogb\(\)\*](#), [\*log1p\(\)\*](#)

XBD [Section 4.20](#) (on page 117), [\*\*<math.h>\*\*](#)

**CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

**Issue 6**

The  $\text{expm1f}()$  and  $\text{expm1l}()$  functions are added for alignment with the ISO/IEC 9899:1999 standard.

The  $\text{expm1}()$  function is no longer marked as an extension.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

**Issue 7**

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0110 [68] is applied.

27221 **NAME**

27222 fabs, fabsf, fabsl — absolute value function

27223 **SYNOPSIS**

```
27224 #include <math.h>
27225 double fabs(double x);
27226 float fabsf(float x);
27227 long double fabsl(long double x);
```

27228 **DESCRIPTION**

27229 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 27230 conflict between the requirements described here and the ISO C standard is unintentional. This  
 27231 volume of POSIX.1-2008 defers to the ISO C standard.

27232 These functions shall compute the absolute value of their argument  $x$ ,  $|x|$ .

27233 **RETURN VALUE**

27234 Upon successful completion, these functions shall return the absolute value of  $x$ .

27235 MX If  $x$  is NaN, a NaN shall be returned.

27236 If  $x$  is  $\pm 0$ ,  $+0$  shall be returned.

27237 If  $x$  is  $\pm \text{Inf}$ ,  $+\text{Inf}$  shall be returned.

27238 **ERRORS**

27239 No errors are defined.

27240 **EXAMPLES**27241 **Computing the 1-Norm of a Floating-Point Vector**

27242 This example shows the use of *fabs()* to compute the 1-norm of a vector defined as follows:

```
27243 norm1(v) = |v[0]| + |v[1]| + ... + |v[n-1]|
```

27244 where  $|x|$  denotes the absolute value of  $x$ ,  $n$  denotes the vector's dimension  $v[i]$  denotes the  $i$ -th  
 27245 component of  $v$  ( $0 \leq i < n$ ).

```
27246 #include <math.h>
27247 double
27248 norm1(const double v[], const int n)
27249 {
27250     int    i;
27251     double nl_v; /* 1-norm of v */
27252     nl_v = 0;
27253     for (i=0; i<n; i++) {
27254         nl_v += fabs (v[i]);
27255     }
27256     return nl_v;
27257 }
```

27258 **APPLICATION USAGE**

27259 None.

27260 **RATIONALE**

27261 None.

27262 **FUTURE DIRECTIONS**

27263 None.

27264 **SEE ALSO**27265 *isnan()*27266 XBD **<math.h>**27267 **CHANGE HISTORY**

27268 First released in Issue 1. Derived from Issue 1 of the SVID.

27269 **Issue 5**

27270 The DESCRIPTION is updated to indicate how an application should check for an error. This  
 27271 text was previously published in the APPLICATION USAGE section.

27272 **Issue 6**

27273 The *fabsf()* and *fabsl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

27274 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
 27275 revised to align with the ISO/IEC 9899:1999 standard.

27276 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
 27277 marked.

27278 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/27 is applied, adding the example to the  
 27279 EXAMPLES section.

27280 **NAME**

27281        faccessat — determine accessibility of a file relative to directory file descriptor

27282 **SYNOPSIS**27283        `#include <unistd.h>`27284        `int faccessat(int fd, const char *path, int amode, int flag);`27285 **DESCRIPTION**27286        Refer to *access()*.



**NAME**

fattach — attach a STREAMS-based file descriptor to a file in the file system name space (STREAMS)

**SYNOPSIS**

```
#include <stropts.h>

int fattach(int fildes, const char *path);
```

**DESCRIPTION**

The *fattach()* function shall attach a STREAMS-based file descriptor to a file, effectively associating a pathname with *fildes*. The application shall ensure that the *fildes* argument is a valid open file descriptor associated with a STREAMS file. The *path* argument points to a pathname of an existing file. The application shall have appropriate privileges or be the owner of the file named by *path* and have write permission. A successful call to *fattach()* shall cause all pathnames that name the file named by *path* to name the STREAMS file associated with *fildes*, until the STREAMS file is detached from the file. A STREAMS file can be attached to more than one file and can have several pathnames associated with it.

The attributes of the named STREAMS file shall be initialized as follows: the permissions, user ID, group ID, and times are set to those of the file named by *path*, the number of links is set to 1, and the size and device identifier are set to those of the STREAMS file associated with *fildes*. If any attributes of the named STREAMS file are subsequently changed (for example, by *chmod()*), neither the attributes of the underlying file nor the attributes of the STREAMS file to which *fildes* refers shall be affected.

File descriptors referring to the underlying file, opened prior to an *fattach()* call, shall continue to refer to the underlying file.

**RETURN VALUE**

Upon successful completion, *fattach()* shall return 0. Otherwise,  $-1$  shall be returned and *errno* set to indicate the error.

**ERRORS**

The *fattach()* function shall fail if:

- |                |   |
|----------------|---|
| [EACCES]       | Search permission is denied for a component of the path prefix, or the process is the owner of <i>path</i> but does not have write permissions on the file named by <i>path</i> .   |
| [EBADF]        | The <i>fildes</i> argument is not a valid open file descriptor.   |
| [EBUSY]        | The file named by <i>path</i> is currently a mount point or has a STREAMS file attached to it.  |
| [ELOOP]        | A loop exists in symbolic links encountered during resolution of the <i>path</i> argument.  |
| [ENAMETOOLONG] | The length of a component of a pathname is longer than {NAME_MAX}.  |
| [ENOENT]       | A component of <i>path</i> does not name an existing file or <i>path</i> is an empty string.  |
| [ENOTDIR]      | A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>path</i> argument contains at least one non- <code>&lt;slash&gt;</code> character and ends with one or more trailing <code>&lt;slash&gt;</code> characters. |

27330 [EPERM] The effective user ID of the process is not the owner of the file named by *path*  
 27331 and the process does not have appropriate privileges.

27332 The *fattach()* function may fail if:

27333 [EINVAL] The *fildev* argument does not refer to a STREAMS file.

27334 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
 27335 resolution of the *path* argument.

27336 [ENAMETOOLONG]

27337 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
 27338 symbolic link produced an intermediate result with a length that exceeds  
 27339 {PATH\_MAX}.

27340 [EXDEV] A link to a file on another file system was attempted.

## 27341 EXAMPLES

### 27342 Attaching a File Descriptor to a File

27343 In the following example, *fd* refers to an open STREAMS file. The call to *fattach()* associates this  
 27344 STREAM with the file **/tmp/named-STREAM**, such that any future calls to open **/tmp/named-**  
 27345 **STREAM**, prior to breaking the attachment via a call to *fdetach()*, will instead create a new file  
 27346 handle referring to the STREAMS file associated with *fd*.

```
27347 #include <stropts.h>
27348 ...
27349     int fd;
27350     char *pathname = "/tmp/named-STREAM";
27351     int ret;
27352
27353     ret = fattach(fd, pathname);
```

## 27353 APPLICATION USAGE

27354 The *fattach()* function behaves similarly to the traditional *mount()* function in the way a file is  
 27355 temporarily replaced by the root directory of the mounted file system. In the case of *fattach()*, the  
 27356 replaced file need not be a directory and the replacing file is a STREAMS file.

## 27357 RATIONALE

27358 The file attributes of a file which has been the subject of an *fattach()* call are specifically set  
 27359 because of an artifact of the original implementation. The internal mechanism was the same as  
 27360 for the *mount()* function. Since *mount()* is typically only applied to directories, the effects when  
 27361 applied to a regular file are a little surprising, especially as regards the link count which rigidly  
 27362 remains one, even if there were several links originally and despite the fact that all original links  
 27363 refer to the STREAM as long as the *fattach()* remains in effect.

## 27364 FUTURE DIRECTIONS

27365 The *fattach()* function may be removed in a future version.

## 27366 SEE ALSO

27367 *fdetach()*, *isastream()*

27368 XBD <stropts.h>

## 27369 CHANGE HISTORY

27370 First released in Issue 4, Version 2.

27371 **Issue 5**

27372 Moved from X/OPEN UNIX extension to BASE.

27373 The [EXDEV] error is added to the list of optional errors in the ERRORS section.

27374 **Issue 6**

27375 This function is marked as part of the XSI STREAMS Option Group.

27376 The normative text is updated to avoid use of the term “must” for application requirements.

27377 The wording of the mandatory [ELOOP] error condition is updated, and a second optional  
27378 [ELOOP] error condition is added.27379 **Issue 7**

27380 Austin Group Interpretation 1003.1-2001 #143 is applied.

27381 The *fattach()* function is marked obsolescent.27382 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a  
27383 pathname exists but is not a directory or a symbolic link to a directory.27384 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0111 [146,324] and  
27385 XSH/TC1-2008/0112 [291] are applied.

27386 **NAME**

27387 fchdir — change working directory

27388 **SYNOPSIS**

27389 #include &lt;unistd.h&gt;

27390 int fchdir(int *fildev*);27391 **DESCRIPTION**27392 The *fchdir()* function shall be equivalent to *chdir()* except that the directory that is to be the new  
27393 current working directory is specified by the file descriptor *fildev*.27394 A conforming application can obtain a file descriptor for a file of type directory using *open()*,  
27395 provided that the file status flags and access modes do not contain O\_WRONLY or O\_RDWR.27396 **RETURN VALUE**27397 Upon successful completion, *fchdir()* shall return 0. Otherwise, it shall return -1 and set *errno* to  
27398 indicate the error. On failure the current working directory shall remain unchanged.27399 **ERRORS**27400 The *fchdir()* function shall fail if:27401 [EACCES] Search permission is denied for the directory referenced by *fildev*.27402 [EBADF] The *fildev* argument is not an open file descriptor.27403 [ENOTDIR] The open file descriptor *fildev* does not refer to a directory.27404 The *fchdir()* may fail if:27405 [EINTR] A signal was caught during the execution of *fchdir()*.

27406 [EIO] An I/O error occurred while reading from or writing to the file system.

27407 **EXAMPLES**

27408 None.

27409 **APPLICATION USAGE**

27410 None.

27411 **RATIONALE**

27412 None.

27413 **FUTURE DIRECTIONS**

27414 None.

27415 **SEE ALSO**27416 *chdir()*, *dirfd()*

27417 XBD &lt;unistd.h&gt;

27418 **CHANGE HISTORY**

27419 First released in Issue 4, Version 2.

27420 **Issue 5**

27421 Moved from X/OPEN UNIX extension to BASE.

27422 **Issue 7**27423 The *fchdir()* function is moved from the XSI option to the Base.

27424 **NAME**

27425 fchmod — change mode of a file

27426 **SYNOPSIS**

27427 #include &lt;sys/stat.h&gt;

27428 int fchmod(int *fildes*, mode\_t *mode*);27429 **DESCRIPTION**27430 The *fchmod()* function shall be equivalent to *chmod()* except that the file whose permissions are  
27431 changed is specified by the file descriptor *fildes*.27432 SHM If *fildes* references a shared memory object, the *fchmod()* function need only affect the S\_IRUSR,  
27433 S\_IWUSR, S\_IRGRP, S\_IWGRP, S\_IROTH, and S\_IWOTH file permission bits.27434 TYM If *fildes* references a typed memory object, the behavior of *fchmod()* is unspecified.27435 If *fildes* refers to a socket, the behavior of *fchmod()* is unspecified.27436 OB XSR If *fildes* refers to a STREAM (which is *fattach()*-ed into the file system name space) the call  
27437 returns successfully, doing nothing.27438 **RETURN VALUE**27439 Upon successful completion, *fchmod()* shall return 0. Otherwise, it shall return -1 and set *errno* to  
27440 indicate the error.27441 **ERRORS**27442 The *fchmod()* function shall fail if:27443 [EBADF] The *fildes* argument is not an open file descriptor.27444 [EPERM] The effective user ID does not match the owner of the file and the process does  
27445 not have appropriate privileges.27446 [EROFS] The file referred to by *fildes* resides on a read-only file system.27447 The *fchmod()* function may fail if:27448 XSI [EINTR] The *fchmod()* function was interrupted by a signal.27449 XSI [EINVAL] The value of the *mode* argument is invalid.27450 [EINVAL] The *fildes* argument refers to a pipe and the implementation disallows  
27451 execution of *fchmod()* on a pipe.27452 **EXAMPLES**27453 **Changing the Current Permissions for a File**27454 The following example shows how to change the permissions for a file named */home/cnd/mod1*  
27455 so that the owner and group have read/write/execute permissions, but the world only has  
27456 read/write permissions.

27457 #include &lt;sys/stat.h&gt;

27458 #include &lt;fcntl.h&gt;

27459 mode\_t mode;

27460 int fildes;

27461 ...

27462 fildes = open("/home/cnd/mod1", O\_RDWR);

27463 fchmod(fildes, S\_IRWXU | S\_IRWXG | S\_IROTH | S\_IWOTH);

27464 **APPLICATION USAGE**

27465 None.

27466 **RATIONALE**

27467 None.

27468 **FUTURE DIRECTIONS**

27469 None.

27470 **SEE ALSO**27471 *chmod()*, *chown()*, *creat()*, *fcntl()*, *fstatat()*, *fstatvfs()*, *mknod()*, *open()*, *read()*, *write()*

27472 XBD &lt;sys/stat.h&gt;

27473 **CHANGE HISTORY**

27474 First released in Issue 4, Version 2.

27475 **Issue 5**

27476 Moved from X/OPEN UNIX extension to BASE and aligned with *fchmod()* in the POSIX  
27477 Realtime Extension. Specifically, the second paragraph of the DESCRIPTION is added and a  
27478 second instance of [EINVAL] is defined in the list of optional errors.

27479 **Issue 6**

27480 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by stating that *fchmod()*  
27481 behavior is unspecified for typed memory objects.

27482 **NAME**

27483 fchmodat — change mode of a file relative to directory file descriptor

27484 **SYNOPSIS**

27485 #include &lt;sys/stat.h&gt;

27486 int fchmodat(int *fd*, const char \**path*, mode\_t *mode*, int *flag*);27487 **DESCRIPTION**27488 Refer to *chmod()*.

27489 **NAME**

27490 fchown — change owner and group of a file

27491 **SYNOPSIS**

27492 #include &lt;unistd.h&gt;

27493 int fchown(int *fildev*, uid\_t *owner*, gid\_t *group*);27494 **DESCRIPTION**27495 The *fchown()* function shall be equivalent to *chown()* except that the file whose owner and group  
27496 are changed is specified by the file descriptor *fildev*.27497 **RETURN VALUE**27498 Upon successful completion, *fchown()* shall return 0. Otherwise, it shall return -1 and set *errno* to  
27499 indicate the error.27500 **ERRORS**27501 The *fchown()* function shall fail if:27502 [EBADF] The *fildev* argument is not an open file descriptor.27503 [EPERM] The effective user ID does not match the owner of the file or the process does  
27504 not have appropriate privileges and \_POSIX\_CHOWN\_RESTRICTED  
27505 indicates that such privilege is required.27506 [EROFS] The file referred to by *fildev* resides on a read-only file system.27507 The *fchown()* function may fail if:27508 [EINVAL] The owner or group ID is not a value supported by the implementation. The  
27509 OB XSR *fildev* argument refers to a pipe or socket or an *fattach()*-ed STREAM and the  
27510 implementation disallows execution of *fchown()* on a pipe.

27511 [EIO] A physical I/O error has occurred.

27512 [EINTR] The *fchown()* function was interrupted by a signal which was caught.27513 **EXAMPLES**27514 **Changing the Current Owner of a File**27515 The following example shows how to change the owner of a file named */home/cnd/mod1* to  
27516 "jones" and the group to "cnd".27517 The numeric value for the user ID is obtained by extracting the user ID from the user database  
27518 entry associated with "jones". Similarly, the numeric value for the group ID is obtained by  
27519 extracting the group ID from the group database entry associated with "cnd". This example  
27520 assumes the calling program has appropriate privileges.

27521 #include &lt;sys/types.h&gt;

27522 #include &lt;unistd.h&gt;

27523 #include &lt;fcntl.h&gt;

27524 #include &lt;pwd.h&gt;

27525 #include &lt;grp.h&gt;

27526 struct passwd \*pwd;

27527 struct group \*grp;

27528 int fildev;

27529 ...

27530 fildev = open("/home/cnd/mod1", O\_RDWR);

27531 pwd = getpwnam("jones");



```

27532         grp = getgrnam("cnd");
27533         fchown(fildes, pwd->pw_uid, grp->gr_gid);

```

#### 27534 APPLICATION USAGE

27535 None.

#### 27536 RATIONALE

27537 None.

#### 27538 FUTURE DIRECTIONS

27539 None.

#### 27540 SEE ALSO

27541 *chown()*

27542 XBD <unistd.h>

#### 27543 CHANGE HISTORY

27544 First released in Issue 4, Version 2.

#### 27545 Issue 5

27546 Moved from X/OPEN UNIX extension to BASE.

#### 27547 Issue 6

27548 The following changes were made to align with the IEEE P1003.1a draft standard:

- 27549 • Clarification is added that a call to *fchown()* may not be allowed on a pipe.

27550 The *fchown()* function is defined as mandatory.

#### 27551 Issue 7

27552 Functionality relating to XSI STREAMS is marked obsolescent.

27553 **NAME**

27554       fchownat — change owner and group of a file relative to directory file descriptor

27555 **SYNOPSIS**

27556       #include &lt;unistd.h&gt;

27557       int fchownat(int *fd*, const char \**path*, uid\_t *owner*, gid\_t *group*,  
27558                   int *flag*);27559 **DESCRIPTION**27560       Refer to *chown()*.

27561 **NAME**

27562       fclose — close a stream

27563 **SYNOPSIS**

27564       #include &lt;stdio.h&gt;

27565       int fclose(FILE \**stream*);27566 **DESCRIPTION**

27567 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 27568 conflict between the requirements described here and the ISO C standard is unintentional. This  
 27569 volume of POSIX.1-2008 defers to the ISO C standard.

27570       The *fclose()* function shall cause the stream pointed to by *stream* to be flushed and the associated  
 27571 file to be closed. Any unwritten buffered data for the stream shall be written to the file; any  
 27572 unread buffered data shall be discarded. Whether or not the call succeeds, the stream shall be  
 27573 disassociated from the file and any buffer set by the *setbuf()* or *setvbuf()* function shall be  
 27574 disassociated from the stream. If the associated buffer was automatically allocated, it shall be  
 27575 deallocated.

27576 CX       If the file is not already at EOF, and the file is one capable of seeking, the file offset of the  
 27577 underlying open file description shall be set to the file position of the stream if the stream is the  
 27578 active handle to the underlying file description.

27579       The *fclose()* function shall mark for update the last data modification and last file status change  
 27580 timestamps of the underlying file, if the stream was writable, and if buffered data remains that  
 27581 has not yet been written to the file. The *fclose()* function shall perform the equivalent of a *close()*  
 27582 on the file descriptor that is associated with the stream pointed to by *stream*.

27583       After the call to *fclose()*, any use of *stream* results in undefined behavior.

27584 **RETURN VALUE**

27585 CX       Upon successful completion, *fclose()* shall return 0; otherwise, it shall return EOF and set *errno*  
 27586 to indicate the error.

27587 **ERRORS**

27588       The *fclose()* function shall fail if:

27589 CX       [EAGAIN]       The O\_NONBLOCK flag is set for the file descriptor underlying *stream* and  
 27590 the thread would be delayed in the write operation.

27591 CX       [EBADF]       The file descriptor underlying stream is not valid.

27592 CX       [EFBIG]       An attempt was made to write a file that exceeds the maximum file size.

27593 XSI       [EFBIG]       An attempt was made to write a file that exceeds the file size limit of the  
 27594 process.

27595 CX       [EFBIG]       The file is a regular file and an attempt was made to write at or beyond the  
 27596 offset maximum associated with the corresponding stream.

27597 CX       [EINTR]       The *fclose()* function was interrupted by a signal.

27598 CX       [EIO]        The process is a member of a background process group attempting to write to  
 27599 its controlling terminal, TOSTOP is set, the calling thread is not blocking  
 27600 SIGTTOU, the process is not ignoring SIGTTOU, and the process group of the  
 27601 process is orphaned. This error may also be returned under implementation-  
 27602 defined conditions.

27603 CX [ENOMEM] The underlying stream was created by *open\_memstream()* or  
 27604 *open\_wmemstream()* and insufficient memory is available.

27605 CX [ENOSPC] There was no free space remaining on the device containing the file or in the  
 27606 buffer used by the *fmemopen()* function.

27607 CX [EPIPE] An attempt is made to write to a pipe or FIFO that is not open for reading by  
 27608 any process. A SIGPIPE signal shall also be sent to the thread.

27609 The *fclose()* function may fail if:

27610 CX [ENXIO] A request was made of a nonexistent device, or the request was outside the  
 27611 capabilities of the device.

## 27612 EXAMPLES

27613 None.

## 27614 APPLICATION USAGE

27615 Since after the call to *fclose()* any use of stream results in undefined behavior, *fclose()* should not  
 27616 be used on *stdin*, *stdout*, or *stderr* except immediately before process termination (see XBD  
 27617 Section 3.303, on page 82), so as to avoid triggering undefined behavior in other standard  
 27618 interfaces that rely on these streams. If there are any *atexit()* handlers registered by the  
 27619 application, such a call to *fclose()* should not occur until the last handler is finishing. Once  
 27620 *fclose()* has been used to close *stdin*, *stdout*, or *stderr*, there is no standard way to reopen any of  
 27621 these streams.

27622 Use of *freopen()* to change *stdin*, *stdout*, or *stderr* instead of closing them avoids the danger of a  
 27623 file unexpectedly being opened as one of the special file descriptors *STDIN\_FILENO*,  
 27624 *STDOUT\_FILENO*, or *STDERR\_FILENO* at a later time in the application.

## 27625 RATIONALE

27626 None.

## 27627 FUTURE DIRECTIONS

27628 None.

## 27629 SEE ALSO

27630 Section 2.5 (on page 495), *atexit()*, *close()*, *fmemopen()*, *fopen()*, *freopen()*, *getrlimit()*,  
 27631 *open\_memstream()*, *ulimit()*

27632 XBD <stdio.h>

## 27633 CHANGE HISTORY

27634 First released in Issue 1. Derived from Issue 1 of the SVID.

### 27635 Issue 5

27636 Large File Summit extensions are added.

### 27637 Issue 6

27638 Extensions beyond the ISO C standard are marked.

27639 The following new requirements on POSIX implementations derive from alignment with the  
 27640 Single UNIX Specification:

- 27641 • The [EFBIG] error is added as part of the large file support extensions.
- 27642 • The [ENXIO] optional error condition is added.

27643 The DESCRIPTION is updated to note that the stream and any buffer are disassociated whether  
 27644 or not the call succeeds. This is for alignment with the ISO/IEC 9899:1999 standard.

27645 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/28 is applied, updating the [EAGAIN]  
27646 error in the ERRORS section from “the process would be delayed” to “the thread would be  
27647 delayed”.

27648 **Issue 7**

27649 Austin Group Interpretation 1003.1-2001 #002 is applied, clarifying the interaction of file  
27650 descriptors and streams.

27651 The [ENOSPC] error condition is updated and the [ENOMEM] error is added from The Open  
27652 Group Technical Standard, 2006, Extended API Set Part 1.

27653 Changes are made related to support for finegrained timestamps.

27654 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0113 [87], XSH/TC1-2008/0114 [79],  
27655 and XSH/TC1-2008/0115 [14] are applied.

## 27656 NAME

27657 fcntl — file control

## 27658 SYNOPSIS

27659 #include &lt;fcntl.h&gt;

27660 int fcntl(int *fildes*, int *cmd*, ...);

## 27661 DESCRIPTION

27662 The *fcntl()* function shall perform the operations described below on open files. The *fildes*  
 27663 argument is a file descriptor.

27664 The available values for *cmd* are defined in <fcntl.h> and are as follows:

27665 F\_DUPFD Return a new file descriptor which shall be allocated as described in |  
 27666 Section 2.14 (on page 549), except that it shall be the lowest numbered |  
 27667 available file descriptor greater than or equal to the third argument, *arg*,  
 27668 taken as an integer of type **int**. The new file descriptor shall refer to the  
 27669 same open file description as the original file descriptor, and shall share  
 27670 any locks. The FD\_CLOEXEC flag associated with the new file descriptor  
 27671 shall be cleared to keep the file open across calls to one of the *exec*  
 27672 functions.

27673 F\_DUPFD\_CLOEXEC

27674 Like F\_DUPFD, but the FD\_CLOEXEC flag associated with the new file  
 27675 descriptor shall be set.

27676 F\_GETFD

27677 Get the file descriptor flags defined in <fcntl.h> that are associated with  
 27678 the file descriptor *fildes*. File descriptor flags are associated with a single  
 27679 file descriptor and do not affect other file descriptors that refer to the  
 same file.

27680 F\_SETFD

27681 Set the file descriptor flags defined in <fcntl.h>, that are associated with  
 27682 *fildes*, to the third argument, *arg*, taken as type **int**. If the FD\_CLOEXEC  
 27683 flag in the third argument is 0, the file descriptor shall remain open across  
 27684 the *exec* functions; otherwise, the file descriptor shall be closed upon  
 successful execution of one of the *exec* functions.

27685 F\_GETFL

27686 Get the file status flags and file access modes, defined in <fcntl.h>, for the  
 27687 file description associated with *fildes*. The file access modes can be  
 27688 extracted from the return value using the mask O\_ACCMODE, which is  
 27689 defined in <fcntl.h>. File status flags and file access modes are associated  
 27690 with the file description and do not affect other file descriptors that refer  
 27691 to the same file with different open file descriptions. The flags returned  
 27692 may include non-standard file status flags which the application did not  
 27693 set, provided that these additional flags do not alter the behavior of a  
 conforming application.

27694 F\_SETFL

27695 Set the file status flags, defined in <fcntl.h>, for the file description  
 27696 associated with *fildes* from the corresponding bits in the third argument,  
 27697 *arg*, taken as type **int**. Bits corresponding to the file access mode and the  
 27698 file creation flags, as defined in <fcntl.h>, that are set in *arg* shall be  
 27699 ignored. If any bits in *arg* other than those mentioned here are changed by  
 27700 the application, the result is unspecified. If *fildes* does not support non-  
 27701 blocking operations, it is unspecified whether the O\_NONBLOCK flag  
 will be ignored.

27702	F_GETOWN	If <i>fildest</i> refers to a socket, get the process ID or process group ID specified	
27703		to receive SIGURG signals when out-of-band data is available. Positive	
27704		values shall indicate a process ID; negative values, other than -1, shall	
27705		indicate a process group ID; the value zero shall indicate that no SIGURG	
27706		signals are to be sent. If <i>fildest</i> does not refer to a socket, the results are	
27707		unspecified.	
27708	F_SETOWN	If <i>fildest</i> refers to a socket, set the process ID or process group ID specified	
27709		to receive SIGURG signals when out-of-band data is available, using the	
27710		value of the third argument, <i>arg</i> , taken as type <b>int</b> . Positive values shall	
27711		indicate a process ID; negative values, other than -1, shall indicate a	
27712		process group ID; the value zero shall indicate that no SIGURG signals	
27713		are to be sent. Each time a SIGURG signal is sent to the specified process	
27714		or process group, permission checks equivalent to those performed by	
27715		<i>kill()</i> shall be performed, as if <i>kill()</i> were called by a process with the	
27716		same real user ID, effective user ID, and privileges that the process calling	
27717		<i>fcntl()</i> has at the time of the call; if the <i>kill()</i> call would fail, no signal shall	
27718		be sent.	
27719		These permission checks may also be performed by the <i>fcntl()</i> call. If the	
27720		process specified by <i>arg</i> later terminates, or the process group specified by	
27721		<i>arg</i> later becomes empty, while still being specified to receive SIGURG	
27722		signals when out-of-band data is available from <i>fildest</i> , then no signals	
27723		shall be sent to any subsequently created process that has the same	
27724		process ID or process group ID, regardless of permission; it is unspecified	
27725		whether this is achieved by the equivalent of a <i>fcntl(fildest, F_SETOWN, 0)</i>	
27726		call at the time the process terminates or is waited for or the process	
27727		group becomes empty, or by other means. If <i>fildest</i> does not refer to a	
27728		socket, the results are unspecified.	
27729	The following values for <i>cmd</i> are available for advisory record locking. Record locking shall be		
27730	supported for regular files, and may be supported for other files.		
27731	F_GETLK	Get any lock which blocks the lock description pointed to by the third	
27732		argument, <i>arg</i> , taken as a pointer to type <b>struct flock</b> , defined in	
27733		<b>&lt;fcntl.h&gt;</b> . The information retrieved shall overwrite the information	
27734		passed to <i>fcntl()</i> in the structure <b>flock</b> . If no lock is found that would	
27735		prevent this lock from being created, then the structure shall be left	
27736		unchanged except for the lock type which shall be set to F_UNLCK.	
27737	F_SETLK	Set or clear a file segment lock according to the lock description pointed	
27738		to by the third argument, <i>arg</i> , taken as a pointer to type <b>struct flock</b> ,	
27739		defined in <b>&lt;fcntl.h&gt;</b> . F_SETLK can establish shared (or read) locks	
27740		(F_RDLCK) or exclusive (or write) locks (F_WRLCK), as well as to	
27741		remove either type of lock (F_UNLCK). F_RDLCK, F_WRLCK, and	
27742		F_UNLCK are defined in <b>&lt;fcntl.h&gt;</b> . If a shared or exclusive lock cannot	
27743		be set, <i>fcntl()</i> shall return immediately with a return value of -1.	
27744	F_SETLKW	This command shall be equivalent to F_SETLK except that if a shared or	
27745		exclusive lock is blocked by other locks, the thread shall wait until the	
27746		request can be satisfied. If a signal that is to be caught is received while	
27747		<i>fcntl()</i> is waiting for a region, <i>fcntl()</i> shall be interrupted. Upon return	
27748		from the signal handler, <i>fcntl()</i> shall return -1 with <i>errno</i> set to [EINTR],	
27749		and the lock operation shall not be done.	

27750 Additional implementation-defined values for *cmd* may be defined in `<fcntl.h>`. Their names  
27751 shall start with `F_`.

27752 When a shared lock is set on a segment of a file, other processes shall be able to set shared locks  
27753 on that segment or a portion of it. A shared lock prevents any other process from setting an  
27754 exclusive lock on any portion of the protected area. A request for a shared lock shall fail if the  
27755 file descriptor was not opened with read access.

27756 An exclusive lock shall prevent any other process from setting a shared lock or an exclusive lock  
27757 on any portion of the protected area. A request for an exclusive lock shall fail if the file  
27758 descriptor was not opened with write access.

27759 The structure **lock** describes the type (*l\_type*), starting offset (*l\_whence*), relative offset (*l\_start*),  
27760 size (*l\_len*), and process ID (*l\_pid*) of the segment of the file to be affected.

27761 The value of *l\_whence* is `SEEK_SET`, `SEEK_CUR`, or `SEEK_END`, to indicate that the relative  
27762 offset *l\_start* bytes shall be measured from the start of the file, current position, or end of the file,  
27763 respectively. The value of *l\_len* is the number of consecutive bytes to be locked. The value of *l\_len*  
27764 may be negative (where the definition of **off\_t** permits negative values of *l\_len*). The *l\_pid* field  
27765 is only used with `F_GETLK` to return the process ID of the process holding a blocking lock. After  
27766 a successful `F_GETLK` request, when a blocking lock is found, the values returned in the **lock**  
27767 structure shall be as follows:

27768 *l\_type*           Type of blocking lock found.

27769 *l\_whence*       `SEEK_SET`.

27770 *l\_start*          Start of the blocking lock.

27771 *l\_len*           Length of the blocking lock.

27772 *l\_pid*           Process ID of the process that holds the blocking lock.

27773 If the command is `F_SETLK` and the process must wait for another process to release a lock,  
27774 then the range of bytes to be locked shall be determined before the *fcntl()* function blocks. If the  
27775 file size or file descriptor seek offset change while *fcntl()* is blocked, this shall not affect the  
27776 range of bytes locked.

27777 If *l\_len* is positive, the area affected shall start at *l\_start* and end at *l\_start+l\_len-1*. If *l\_len* is  
27778 negative, the area affected shall start at *l\_start+l\_len* and end at *l\_start-1*. Locks may start and  
27779 extend beyond the current end of a file, but shall not extend before the beginning of the file. A  
27780 lock shall be set to extend to the largest possible value of the file offset for that file by setting  
27781 *l\_len* to 0. If such a lock also has *l\_start* set to 0 and *l\_whence* is set to `SEEK_SET`, the whole file  
27782 shall be locked.

27783 There shall be at most one type of lock set for each byte in the file. Before a successful return  
27784 from an `F_SETLK` or an `F_SETLKW` request when the calling process has previously existing  
27785 locks on bytes in the region specified by the request, the previous lock type for each byte in the  
27786 specified region shall be replaced by the new lock type. As specified above under the  
27787 descriptions of shared locks and exclusive locks, an `F_SETLK` or an `F_SETLKW` request  
27788 (respectively) shall fail or block when another process has existing locks on bytes in the specified  
27789 region and the type of any of those locks conflicts with the type specified in the request.

27790 All locks associated with a file for a given process shall be removed when a file descriptor for  
27791 that file is closed by that process or the process holding that file descriptor terminates. Locks are  
27792 not inherited by a child process.

27793 A potential for deadlock occurs if a process controlling a locked region is put to sleep by  
27794 attempting to lock the locked region of another process. If the system detects that sleeping until



27795 a locked region is unlocked would cause a deadlock, *fcntl()* shall fail with an [EDEADLK] error.

27796 An unlock (F\_UNLCK) request in which *l\_len* is non-zero and the offset of the last byte of the  
 27797 requested segment is the maximum value for an object of type **off\_t**, when the process has an  
 27798 existing lock in which *l\_len* is 0 and which includes the last byte of the requested segment, shall  
 27799 be treated as a request to unlock from the start of the requested segment with an *l\_len* equal to 0.  
 27800 Otherwise, an unlock (F\_UNLCK) request shall attempt to unlock only the requested segment.

27801 SHM When the file descriptor *fildev* refers to a shared memory object, the behavior of *fcntl()* shall be  
 27802 the same as for a regular file except the effect of the following values for the argument *cmd* shall  
 27803 be unspecified: F\_SETFL, F\_GETLK, F\_SETLK, and F\_SETLKW.

27804 TYM If *fildev* refers to a typed memory object, the result of the *fcntl()* function is unspecified.

## 27805 RETURN VALUE

27806 Upon successful completion, the value returned shall depend on *cmd* as follows:

27807 F\_DUPFD A new file descriptor.

27808 F\_DUPFD\_CLOEXEC

27809 A new file descriptor.

27810 F\_GETFD Value of flags defined in **<fcntl.h>**. The return value shall not be negative.

27811 F\_SETFD Value other than -1.

27812 F\_GETFL Value of file status flags and access modes. The return value is not negative.

27813 F\_SETFL Value other than -1.

27814 F\_GETLK Value other than -1.

27815 F\_SETLK Value other than -1.

27816 F\_SETLKW Value other than -1.

27817 F\_GETOWN Value of the socket owner process or process group; this will not be -1.

27818 F\_SETOWN Value other than -1.

27819 Otherwise, -1 shall be returned and *errno* set to indicate the error.

## 27820 ERRORS

27821 The *fcntl()* function shall fail if:

27822 [EACCES] or [EAGAIN]

27823 The *cmd* argument is F\_SETLK; the type of lock (*l\_type*) is a shared (F\_RDLCK)  
 27824 or exclusive (F\_WRLCK) lock and the segment of a file to be locked is already  
 27825 exclusive-locked by another process, or the type is an exclusive lock and some  
 27826 portion of the segment of a file to be locked is already shared-locked or  
 27827 exclusive-locked by another process.

27828 [EBADF] The *fildev* argument is not a valid open file descriptor, or the argument *cmd* is  
 27829 F\_SETLK or F\_SETLKW, the type of lock, *l\_type*, is a shared lock (F\_RDLCK),  
 27830 and *fildev* is not a valid file descriptor open for reading, or the type of lock,  
 27831 *l\_type*, is an exclusive lock (F\_WRLCK), and *fildev* is not a valid file descriptor  
 27832 open for writing.

27833 [EINTR] The *cmd* argument is F\_SETLKW and the function was interrupted by a signal.

27834	[EINVAL]	The <i>cmd</i> argument is invalid, or the <i>cmd</i> argument is F_DUPFD or F_DUPFD_CLOEXEC and <i>arg</i> is negative or greater than or equal to {OPEN_MAX}, or the <i>cmd</i> argument is F_GETLK, F_SETLK, or F_SETLKW and the data pointed to by <i>arg</i> is not valid, or <i>fildev</i> refers to a file that does not support locking.	
27835			
27836			
27837			
27838			
27839	[EMFILE]	The argument <i>cmd</i> is F_DUPFD or F_DUPFD_CLOEXEC and all file descriptors available to the process are currently open, or no file descriptors greater than or equal to <i>arg</i> are available.	
27840			
27841			
27842	[ENOLCK]	The argument <i>cmd</i> is F_SETLK or F_SETLKW and satisfying the lock or unlock request would result in the number of locked regions in the system exceeding a system-imposed limit.	
27843			
27844			
27845	[EOVERFLOW]	One of the values to be returned cannot be represented correctly.	
27846	[EOVERFLOW]	The <i>cmd</i> argument is F_GETLK, F_SETLK, or F_SETLKW and the smallest or, if <i>l_len</i> is non-zero, the largest offset of any byte in the requested segment cannot be represented correctly in an object of type <b>off_t</b> .	
27847			
27848			
27849	[ESRCH]	The <i>cmd</i> argument is F_SETOWN and no process or process group can be found corresponding to that specified by <i>arg</i> .	+
27850			
27851	The <i>fcntl()</i> function may fail if:		
27852	[EDEADLK]	The <i>cmd</i> argument is F_SETLKW, the lock is blocked by a lock from another process, and putting the calling process to sleep to wait for that lock to become free would cause a deadlock.	
27853			
27854			
27855	[EINVAL]	The <i>cmd</i> argument is F_SETOWN and the value of the argument is not valid as a process or process group identifier.	+
27856			+
27857	[EPERM]	The <i>cmd</i> argument is F_SETOWN and the calling process does not have permission to send a SIGURG signal to any process specified by <i>arg</i> .	+
27858			

## EXAMPLES

### Locking and Unlocking a File

The following example demonstrates how to place a lock on bytes 100 to 109 of a file and then later remove it. F\_SETLK is used to perform a non-blocking lock request so that the process does not have to wait if an incompatible lock is held by another process; instead the process can take some other action.

```
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <stdio.h>

int
main(int argc, char *argv[])
{
    int fd;
    struct flock fl;

    fd = open("testfile", O_RDWR);
    if (fd == -1)
        /* Handle error */;
```

```

27878      /* Make a non-blocking request to place a write lock
27879         on bytes 100-109 of testfile */

27880      fl.l_type = F_WRLCK;
27881      fl.l_whence = SEEK_SET;
27882      fl.l_start = 100;
27883      fl.l_len = 10;

27884      if (fcntl(fd, F_SETLK, &fl) == -1) {
27885          if (errno == EACCES || errno == EAGAIN) {
27886              printf("Already locked by another process\n");
27887              /* We can't get the lock at the moment */
27888          } else {
27889              /* Handle unexpected error */;
27890          }
27891      } else { /* Lock was granted... */

27892          /* Perform I/O on bytes 100 to 109 of file */

27893          /* Unlock the locked bytes */

27894          fl.l_type = F_UNLCK;
27895          fl.l_whence = SEEK_SET;
27896          fl.l_start = 100;
27897          fl.l_len = 10;
27898          if (fcntl(fd, F_SETLK, &fl) == -1)
27899              /* Handle error */;
27900      }
27901      exit(EXIT_SUCCESS);
27902  } /* main */

```

### 27903 Setting the Close-on-Exec Flag

27904 The following example demonstrates how to set the close-on-exec flag for the file descriptor *fd*.

```

27905 #include <unistd.h>
27906 #include <fcntl.h>
27907 ...
27908     int flags;

27909     flags = fcntl(fd, F_GETFD);
27910     if (flags == -1)
27911         /* Handle error */;
27912     flags |= FD_CLOEXEC;
27913     if (fcntl(fd, F_SETFD, flags) == -1)
27914         /* Handle error */;

```

### 27915 APPLICATION USAGE

27916 The *arg* values to F\_GETFD, F\_SETFD, F\_GETFL, and F\_SETFL all represent flag values to allow  
27917 for future growth. Applications using these functions should do a read-modify-write operation  
27918 on them, rather than assuming that only the values defined by this volume of POSIX.1-2008 are  
27919 valid. It is a common error to forget this, particularly in the case of F\_SETFD. Some  
27920 implementations set additional file status flags to advise the application of default behavior,  
27921 even though the application did not request these flags.

On systems which do not perform permission checks at the time of an *fcntl()* call with *F\_SETOWN*, if the permission checks performed at the time the signal is sent disallow sending the signal to any process, the process that called *fcntl()* has no way of discovering that this has happened. A call to *kill()* with signal 0 can be used as a prior check of permissions, although this is no guarantee that permission will be granted at the time a signal is sent, since the target process(es) could change user IDs or privileges in the meantime.

## RATIONALE

The ellipsis in the SYNOPSIS is the syntax specified by the ISO C standard for a variable number of arguments. It is used because System V uses pointers for the implementation of file locking functions.

This volume of POSIX.1-2008 permits concurrent read and write access to file data using the *fcntl()* function; this is a change from the 1984 /usr/group standard and early proposals. Without concurrency controls, this feature may not be fully utilized without occasional loss of data.

Data losses occur in several ways. One case occurs when several processes try to update the same record, without sequencing controls; several updates may occur in parallel and the last writer “wins”. Another case is a bit-tree or other internal list-based database that is undergoing reorganization. Without exclusive use to the tree segment by the updating process, other reading processes chance getting lost in the database when the index blocks are split, condensed, inserted, or deleted. While *fcntl()* is useful for many applications, it is not intended to be overly general and does not handle the bit-tree example well.

This facility is only required for regular files because it is not appropriate for many devices such as terminals and network connections.

Since *fcntl()* works with “any file descriptor associated with that file, however it is obtained”, the file descriptor may have been inherited through a *fork()* or *exec* operation and thus may affect a file that another process also has open.

The use of the open file description to identify what to lock requires extra calls and presents problems if several processes are sharing an open file description, but there are too many implementations of the existing mechanism for this volume of POSIX.1-2008 to use different specifications.

Another consequence of this model is that closing any file descriptor for a given file (whether or not it is the same open file description that created the lock) causes the locks on that file to be relinquished for that process. Equivalently, any close for any file/process pair relinquishes the locks owned on that file for that process. But note that while an open file description may be shared through *fork()*, locks are not inherited through *fork()*. Yet locks may be inherited through one of the *exec* functions.

The identification of a machine in a network environment is outside the scope of this volume of POSIX.1-2008. Thus, an *l\_sysid* member, such as found in System V, is not included in the locking structure.

Changing of lock types can result in a previously locked region being split into smaller regions.

Mandatory locking was a major feature of the 1984 /usr/group standard.

For advisory file record locking to be effective, all processes that have access to a file must cooperate and use the advisory mechanism before doing I/O on the file. Enforcement-mode record locking is important when it cannot be assumed that all processes are cooperating. For example, if one user uses an editor to update a file at the same time that a second user executes another process that updates the same file and if only one of the two processes is using advisory

locking, the processes are not cooperating. Enforcement-mode record locking would protect against accidental collisions.

Secondly, advisory record locking requires a process using locking to bracket each I/O operation with lock (or test) and unlock operations. With enforcement-mode file and record locking, a process can lock the file once and unlock when all I/O operations have been completed. Enforcement-mode record locking provides a base that can be enhanced; for example, with sharable locks. That is, the mechanism could be enhanced to allow a process to lock a file so other processes could read it, but none of them could write it.

Mandatory locks were omitted for several reasons:

1. Mandatory lock setting was done by multiplexing the set-group-ID bit in most implementations; this was confusing, at best.
2. The relationship to file truncation as supported in 4.2 BSD was not well specified.
3. Any publicly readable file could be locked by anyone. Many historical implementations keep the password database in a publicly readable file. A malicious user could thus prohibit logins. Another possibility would be to hold open a long-distance telephone line.
4. Some demand-paged historical implementations offer memory mapped files, and enforcement cannot be done on that type of file.

Since sleeping on a region is interrupted with any signal, *alarm()* may be used to provide a timeout facility in applications requiring it. This is useful in deadlock detection. Since implementation of full deadlock detection is not always feasible, the [EDEADLK] error was made optional.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*alarm()*, *close()*, *exec*, *kill()*, *open()*, *sigaction()*

XBD *<fcntl.h>*, *<signal.h>*

## CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

### Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

Large File Summit extensions are added.

### Issue 6

In the SYNOPSIS, the optional include of the *<sys/types.h>* header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include *<sys/types.h>* has been removed. Although *<sys/types.h>* was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- In the DESCRIPTION, sentences describing behavior when *l\_len* is negative are now mandated, and the description of unlock (F\_UNLOCK) when *l\_len* is non-negative is mandated.

- In the ERRORS section, the [EINVAL] error condition has the case mandated when the *cmd* is invalid, and two [EOVERFLOW] error conditions are added.

The F\_GETOWN and F\_SETOWN values are added for sockets.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Clarification is added that the extent of the bytes locked is determined prior to the blocking action.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that *fcntl()* results are unspecified for typed memory objects.

The normative text is updated to avoid use of the term “must” for application requirements.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/29 is applied, adding the example to the EXAMPLES section.

#### Issue 7

Austin Group Interpretation 1003.1-2001 #150 is applied, clarifying the file status flags returned when *cmd* is F\_GETFL.

Austin Group Interpretation 1003.1-2001 #171 is applied, adding support to set the FD\_CLOEXEC flag atomically at *open()*, and adding the F\_DUPFD\_CLOEXEC flag.

The optional **<unistd.h>** header is removed from this function, since **<fcntl.h>** now defines SEEK\_SET, SEEK\_CUR, and SEEK\_END as part of the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0116 [141] is applied.

28029 **NAME**28030 `fdatasync` — synchronize the data of a file (**REALTIME**)28031 **SYNOPSIS**

```
28032 SIO      #include <unistd.h>
28033          int fdatasync(int fildes);
```

28034 **DESCRIPTION**

28035 The `fdatasync()` function shall force all currently queued I/O operations associated with the file  
 28036 indicated by file descriptor *fil*des to the synchronized I/O completion state.

28037 The functionality shall be equivalent to `fsync()` with the symbol `_POSIX_SYNCHRONIZED_IO`  
 28038 defined, with the exception that all I/O operations shall be completed as defined for  
 28039 synchronized I/O data integrity completion.

28040 **RETURN VALUE**

28041 If successful, the `fdatasync()` function shall return the value 0; otherwise, the function shall return  
 28042 the value `-1` and set *errno* to indicate the error. If the `fdatasync()` function fails, outstanding I/O  
 28043 operations are not guaranteed to have been completed.

28044 **ERRORS**

28045 The `fdatasync()` function shall fail if:

28046 [EBADF]           The *fil*des argument is not a valid file descriptor. |

28047 [EINVAL]          This implementation does not support synchronized I/O for this file.

28048 In the event that any of the queued I/O operations fail, `fdatasync()` shall return the error  
 28049 conditions defined for `read()` and `write()`.

28050 **EXAMPLES**

28051 None.

28052 **APPLICATION USAGE**

28053 Note that even if the file descriptor is not open for writing, if there are any pending write |  
 28054 requests on the underlying file, then that I/O will be completed prior to the return of `fdatasync()`.

28055 **RATIONALE**

28056 None.

28057 **FUTURE DIRECTIONS**

28058 None.

28059 **SEE ALSO**

28060 [`aio\_fsync\(\)`](#), [`fcntl\(\)`](#), [`fsync\(\)`](#), [`open\(\)`](#), [`read\(\)`](#), [`write\(\)`](#)

28061 XBD [`<unistd.h>`](#)

28062 **CHANGE HISTORY**

28063 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

28064 **Issue 6**

28065 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
 28066 implementation does not support the Synchronized Input and Output option.

28067 The `fdatasync()` function is marked as part of the Synchronized Input and Output option.



28068 **NAME**28069 fdetach — detach a name from a STREAMS-based file descriptor (**STREAMS**)28070 **SYNOPSIS**

```
28071 OB XSR #include <stropts.h>
28072 int fdetach(const char *path);
```

28073 **DESCRIPTION**

28074 The *fdetach()* function shall detach a STREAMS-based file from the file to which it was attached  
28075 by a previous call to *fattach()*. The *path* argument points to the pathname of the attached  
28076 STREAMS file. The process shall have appropriate privileges or be the owner of the file. A  
28077 successful call to *fdetach()* shall cause all pathnames that named the attached STREAMS file to  
28078 again name the file to which the STREAMS file was attached. All subsequent operations on *path*  
28079 shall operate on the underlying file and not on the STREAMS file.

28080 All open file descriptions established while the STREAMS file was attached to the file referenced  
28081 by *path* shall still refer to the STREAMS file after the *fdetach()* has taken effect.

28082 If there are no open file descriptors or other references to the STREAMS file, then a successful  
28083 call to *fdetach()* shall be equivalent to performing the last *close()* on the attached file.

28084 **RETURN VALUE**

28085 Upon successful completion, *fdetach()* shall return 0; otherwise, it shall return -1 and set *errno* to  
28086 indicate the error.

28087 **ERRORS**

28088 The *fdetach()* function shall fail if:

- |       |                |   |
|-------|----------------|---|
| 28089 | [EACCES]       | Search permission is denied on a component of the path prefix.  |
| 28090 | [EINVAL]       | The <i>path</i> argument names a file that is not currently attached.   |
| 28091 | [ELOOP]        | A loop exists in symbolic links encountered during resolution of the <i>path</i><br>28092 argument.   |
| 28093 | [ENAMETOOLONG] | The length of a component of a pathname is longer than {NAME_MAX}.  |
| 28094 |                |   |
| 28095 | [ENOENT]       | A component of <i>path</i> does not name an existing file or <i>path</i> is an empty string.  |
| 28096 | [ENOTDIR]      | A component of the path prefix names an existing file that is neither a<br>directory nor a symbolic link to a directory, or the <i>path</i> argument contains at<br>least one non- <i>&lt;slash&gt;</i> character and ends with one or more trailing <i>&lt;slash&gt;</i><br>characters and the last pathname component names an existing file that is<br>neither a directory nor a symbolic link to a directory. |
| 28097 |                |   |
| 28098 |                |   |
| 28099 |                |   |
| 28100 |                |   |
| 28101 | [EPERM]        | The effective user ID is not the owner of <i>path</i> and the process does not have<br>28102 appropriate privileges.  |

28103 The *fdetach()* function may fail if:

- |       |                |   |
|-------|----------------|---|
| 28104 | [ELOOP]        | More than {SYMLOOP_MAX} symbolic links were encountered during<br>28105 resolution of the <i>path</i> argument.   |
| 28106 |                |   |
| 28107 | [ENAMETOOLONG] | The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a<br>symbolic link produced an intermediate result with a length that exceeds<br>{PATH_MAX}. |
| 28108 |                |   |
| 28109 |                |   |



28110 **EXAMPLES**28111 **Detaching a File**

28112 The following example detaches the STREAMS-based file **/tmp/named-STREAM** from the file to  
28113 which it was attached by a previous, successful call to *fattach()*. Subsequent calls to open this  
28114 file refer to the underlying file, not to the STREAMS file.

```
28115 #include <stropts.h>
28116 ...
28117     char *pathname = "/tmp/named-STREAM";
28118     int ret;
28119     ret = fdetach(pathname);
```

28120 **APPLICATION USAGE**

28121 None.

28122 **RATIONALE**

28123 None.

28124 **FUTURE DIRECTIONS**

28125 The *fdetach()* function may be removed in a future version.

28126 **SEE ALSO**

28127 *fattach()*

28128 XBD <stropts.h>

28129 **CHANGE HISTORY**

28130 First released in Issue 4, Version 2.

28131 **Issue 5**

28132 Moved from X/OPEN UNIX extension to BASE.

28133 **Issue 6**

28134 The normative text is updated to avoid use of the term “must” for application requirements.

28135 The wording of the mandatory [ELOOP] error condition is updated, and a second optional  
28136 [ELOOP] error condition is added.

28137 **Issue 7**

28138 Austin Group Interpretation 1003.1-2001 #143 is applied.

28139 The *fdetach()* function is marked obsolescent.

28140 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a  
28141 pathname exists but is not a directory or a symbolic link to a directory.

28142 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0117 [324] and XSH/TC1-2008/0118  
28143 [291] are applied.

28144 **NAME**

28145 fdim, fdimf, fdiml — compute positive difference between two floating-point numbers

28146 **SYNOPSIS**

```
28147 #include <math.h>
28148 double fdim(double x, double y);
28149 float fdimf(float x, float y);
28150 long double fdiml(long double x, long double y);
```

28151 **DESCRIPTION**

28152 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 28153 conflict between the requirements described here and the ISO C standard is unintentional. This  
 28154 volume of POSIX.1-2008 defers to the ISO C standard.

28155 These functions shall determine the positive difference between their arguments. If  $x$  is greater  
 28156 than  $y$ ,  $x-y$  is returned. If  $x$  is less than or equal to  $y$ , +0 is returned.

28157 An application wishing to check for error situations should set *errno* to zero and call  
 28158 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 28159 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 28160 zero, an error has occurred.

28161 **RETURN VALUE**

28162 Upon successful completion, these functions shall return the positive difference value.

28163 If  $x-y$  is positive and overflows, a range error shall occur and *fdim*(), *fdimf*(), and *fdiml*() shall  
 28164 return the value of the macro HUGE\_VAL, HUGE\_VALF, and HUGE\_VALL, respectively.

28165 If the correct value would cause underflow, a range error may occur, and *fdim*(), *fdimf*(), and  
 28166 MXX *fdiml*() shall return the correct value, or (if the IEC 60559 Floating-Point option is not  
 28167 supported) an implementation-defined value no greater in magnitude than DBL\_MIN,  
 28168 FLT\_MIN, and LDBL\_MIN, respectively.

28169 MX If  $x$  or  $y$  is NaN, a NaN shall be returned.

28170 **ERRORS**

28171 The *fdim*() function shall fail if:

28172 Range Error The result overflows.

28173 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 28174 then *errno* shall be set to [ERANGE]. If the integer expression  
 28175 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the overflow  
 28176 floating-point exception shall be raised.

28177 The *fdim*() function may fail if:

28178 Range Error The result underflows.

28179 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 28180 then *errno* shall be set to [ERANGE]. If the integer expression  
 28181 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 28182 floating-point exception shall be raised.

**28183 EXAMPLES**

28184       None.

**28185 APPLICATION USAGE**

28186       On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
28187       MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

**28188 RATIONALE**

28189       None.

**28190 FUTURE DIRECTIONS**

28191       None.

**28192 SEE ALSO**

28193       *feclearexcept()*, *fetestexcept()*, *fmax()*, *fmin()*

28194       Section 4.20 (on page 117), <math.h>

**28195 CHANGE HISTORY**

28196       First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

**28197 Issue 7**

28198       POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0119 [68,428] and  
28199       XSH/TC1-2008/0120 [68,428] are applied.

28200 **NAME**

28201 fdopen — associate a stream with a file descriptor

28202 **SYNOPSIS**

```
28203 CX      #include <stdio.h>
28204      FILE *fdopen(int fil-des, const char *mode);
```

28205 **DESCRIPTION**28206 The *fdopen()* function shall associate a stream with a file descriptor.28207 The *mode* argument is a character string having one of the following values:28208 *r* or *rb* Open a file for reading.28209 *w* or *wb* Open a file for writing.28210 *a* or *ab* Open a file for writing at end-of-file.28211 *r+* or *rb+* or *r+b* Open a file for update (reading and writing).28212 *w+* or *wb+* or *w+b* Open a file for update (reading and writing).28213 *a+* or *ab+* or *a+b* Open a file for update (reading and writing) at end-of-file.

28214 The meaning of these flags is exactly as specified in *fopen()*, except that modes beginning with *w*  
 28215 shall not cause truncation of the file.

28216 Additional values for the *mode* argument may be supported by an implementation.

28217 The application shall ensure that the mode of the stream as expressed by the *mode* argument is  
 28218 allowed by the file access mode of the open file description to which *fil-des* refers. The file  
 28219 position indicator associated with the new stream is set to the position indicated by the file offset  
 28220 associated with the file descriptor.

28221 The error and end-of-file indicators for the stream shall be cleared. The *fdopen()* function may  
 28222 cause the last data access timestamp of the underlying file to be marked for update.

28223 SHM If *fil-des* refers to a shared memory object, the result of the *fdopen()* function is unspecified.28224 TYM If *fil-des* refers to a typed memory object, the result of the *fdopen()* function is unspecified.

28225 The *fdopen()* function shall preserve the offset maximum previously set for the open file  
 28226 description corresponding to *fil-des*.

28227 **RETURN VALUE**

28228 Upon successful completion, *fdopen()* shall return a pointer to a stream; otherwise, a null pointer  
 28229 shall be returned and *errno* set to indicate the error.

28230 **ERRORS**28231 The *fdopen()* function shall fail if:

28232 [EMFILE] {STREAM\_MAX} streams are currently open in the calling process.

28233 The *fdopen()* function may fail if:28234 [EBADF] The *fil-des* argument is not a valid file descriptor.28235 [EINVAL] The *mode* argument is not a valid mode.

28236 [EMFILE] {FOPEN\_MAX} streams are currently open in the calling process.

28237 [ENOMEM] Insufficient space to allocate a buffer.

## 28238 EXAMPLES

28239 None.

## 28240 APPLICATION USAGE

28241 File descriptors are obtained from calls like *open()*, *dup()*, *creat()*, or *pipe()*, which open files but  
28242 do not return streams.

## 28243 RATIONALE

28244 The file descriptor may have been obtained from *open()*, *creat()*, *pipe()*, *dup()*, *fcntl()*, or *socket()*;  
28245 inherited through *fork()*, *posix\_spawn()*, or *exec*; or perhaps obtained by other means.

28246 The meanings of the *mode* arguments of *fdopen()* and *fopen()* differ. With *fdopen()*, open for write  
28247 (*w* or *w+*) does not truncate, and append (*a* or *a+*) cannot create for writing. The *mode* argument  
28248 formats that include a *b* are allowed for consistency with the ISO C standard function *fopen()*.  
28249 The *b* has no effect on the resulting stream. Although not explicitly required by this volume of  
28250 POSIX.1-2008, a good implementation of append (*a*) mode would cause the O\_APPEND flag to  
28251 be set.

## 28252 FUTURE DIRECTIONS

28253 None.

## 28254 SEE ALSO

28255 [Section 2.5.1](#) (on page 497), *fclose()*, *fnemopen()*, *fopen()*, *open()*, *open\_memstream()*,  
28256 *posix\_spawn()*, *socket()*

28257 XBD [<stdio.h>](#)

## 28258 CHANGE HISTORY

28259 First released in Issue 1. Derived from Issue 1 of the SVID.

### 28260 Issue 5

28261 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension.

28262 Large File Summit extensions are added.

### 28263 Issue 6

28264 The following new requirements on POSIX implementations derive from alignment with the  
28265 Single UNIX Specification:

- 28266 • In the DESCRIPTION, the use and setting of the *mode* argument are changed to include  
28267 binary streams.
- 28268 • In the DESCRIPTION, text is added for large file support to indicate setting of the offset  
28269 maximum in the open file description.
- 28270 • All errors identified in the ERRORS section are added.
- 28271 • In the DESCRIPTION, text is added that the *fdopen()* function may cause *st\_atime* to be  
28272 updated.

28273 The following changes were made to align with the IEEE P1003.1a draft standard:

- 28274 • Clarification is added that it is the responsibility of the application to ensure that the mode  
28275 is compatible with the open file descriptor.

28276 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that  
28277 *fdopen()* results are unspecified for typed memory objects.

- 28278 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/30 is applied, making corrections to the  
28279 RATIONALE.
- 28280 **Issue 7**
- 28281 SD5-XSH-ERN-149 is applied, adding the {STREAM\_MAX} [EMFILE] error condition.
- 28282 Changes are made related to support for finegrained timestamps.
- 28283 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0121 [409] is applied.

28284 **NAME**

28285 fdopendir, opendir — open directory associated with file descriptor

28286 **SYNOPSIS**

28287 #include &lt;dirent.h&gt;

28288 DIR \*fdopendir(int *fd*);28289 DIR \*opendir(const char \**dirname*);28290 **DESCRIPTION**

28291 The *fdopendir()* function shall be equivalent to the *opendir()* function except that the directory is  
 28292 specified by a file descriptor rather than by a name. The file offset associated with the file  
 28293 descriptor at the time of the call determines which entries are returned.

28294 Upon successful return from *fdopendir()*, the file descriptor is under the control of the system,  
 28295 and if any attempt is made to close the file descriptor, or to modify the state of the associated  
 28296 XSI description, other than by means of *closedir()*, *readdir()*, *readdir\_r()*, *rewinddir()*, or *seekdir()*, the  
 28297 behavior is undefined. Upon calling *closedir()* the file descriptor shall be closed.

28298 It is unspecified whether the FD\_CLOEXEC flag will be set on the file descriptor by a successful  
 28299 call to *fdopendir()*.

28300 The *opendir()* function shall open a directory stream corresponding to the directory named by  
 28301 the *dirname* argument. The directory stream is positioned at the first entry. If the type **DIR** is  
 28302 implemented using a file descriptor, applications shall only be able to open up to a total of  
 28303 {OPEN\_MAX} files and directories.

28304 If the type **DIR** is implemented using a file descriptor, the descriptor shall be obtained as if the  
 28305 O\_DIRECTORY flag was passed to *open()*.

28306 **RETURN VALUE**

28307 Upon successful completion, these functions shall return a pointer to an object of type **DIR**.  
 28308 Otherwise, these functions shall return a null pointer and set *errno* to indicate the error.

28309 **ERRORS**28310 The *fdopendir()* function shall fail if:28311 [EBADF] The *fd* argument is not a valid file descriptor open for reading.28312 [ENOTDIR] The descriptor *fd* is not associated with a directory.28313 The *opendir()* function shall fail if:

28314 [EACCES] Search permission is denied for the component of the path prefix of *dirname* or  
 28315 read permission is denied for *dirname*.

28316 [ELOOP] A loop exists in symbolic links encountered during resolution of the *dirname*  
 28317 argument.

28318 [ENAMETOOLONG]  
 28319 The length of a component of a pathname is longer than {NAME\_MAX}.

28320 [ENOENT] A component of *dirname* does not name an existing directory or *dirname* is an  
 28321 empty string.

28322 [ENOTDIR] A component of *dirname* names an existing file that is neither a directory nor a  
 28323 symbolic link to a directory.

28324 The *opendir()* function may fail if:

28325 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
28326 resolution of the *dirname* argument.

28327 [EMFILE] All file descriptors available to the process are currently open.

28328 [ENAMETOOLONG]  
28329 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
28330 symbolic link produced an intermediate result with a length that exceeds  
28331 {PATH\_MAX}.

28332 [ENFILE] Too many files are currently open in the system.

## 28333 EXAMPLES

### 28334 Open a Directory Stream

28335 The following program fragment demonstrates how the *opendir()* function is used.

```
28336 #include <dirent.h>
28337 ...
28338     DIR *dir;
28339     struct dirent *dp;
28340 ...
28341     if ((dir = opendir(".")) == NULL) {
28342         perror("Cannot open .");
28343         exit(1);
28344     }
28345     while ((dp = readdir(dir)) != NULL) {
28346 ...
```

### 28347 Find And Open a File

28348 The following program searches through a given directory looking for files whose name does  
28349 not begin with a dot and whose size is larger than 1 MiB.

```
28350 #include <stdio.h>
28351 #include <dirent.h>
28352 #include <fcntl.h>
28353 #include <sys/stat.h>
28354 #include <stdint.h>
28355 #include <stdlib.h>
28356 #include <unistd.h>
28357
28358 int
28359 main(int argc, char *argv[])
28360 {
28361     struct stat statbuf;
28362     DIR *d;
28363     struct dirent *dp;
28364     int dfd, ffd;
28365
28366     if ((d = fdopendir((dfd = open("./tmp", O_RDONLY))) == NULL) {
28367         fprintf(stderr, "Cannot open ./tmp directory\n");
28368         exit(1);
28369     }
```



```

28367     }
28368     while ((dp = readdir(d)) != NULL) {
28369         if (dp->d_name[0] == '.')
28370             continue;
28371         /* there is a possible race condition here as the file
28372          * could be renamed between the readdir and the open */
28373         if ((ffd = openat(dfd, dp->d_name, O_RDONLY)) == -1) {
28374             perror(dp->d_name);
28375             continue;
28376         }
28377         if (fstat(ffd, &statbuf) == 0 && statbuf.st_size > (1024*1024)) {
28378             /* found it ... */
28379             printf("%s: %jdK\n", dp->d_name,
28380                 (intmax_t)(statbuf.st_size / 1024));
28381         }
28382         close(ffd);
28383     }
28384     closedir(d); // note this implicitly closes dfd
28385     return 0;
28386 }

```

## APPLICATION USAGE

The *opendir()* function should be used in conjunction with *readdir()*, *closedir()*, and *rewinddir()* to examine the contents of the directory (see the EXAMPLES section in *readdir()*). This method is recommended for portability.

## RATIONALE

The purpose of the *fdopendir()* function is to enable opening files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *opendir()*, resulting in unspecified behavior.

Based on historical implementations, the rules about file descriptors apply to directory streams as well. However, this volume of POSIX.1-2008 does not mandate that the directory stream be implemented using file descriptors. The description of *closedir()* clarifies that if a file descriptor is used for the directory stream, it is mandatory that *closedir()* deallocate the file descriptor. When a file descriptor is used to implement the directory stream, it behaves as if the *FD\_CLOEXEC* had been set for the file descriptor.

The directory entries for dot and dot-dot are optional. This volume of POSIX.1-2008 does not provide a way to test *a priori* for their existence because an application that is portable must be written to look for (and usually ignore) those entries. Writing code that presumes that they are the first two entries does not always work, as many implementations permit them to be other than the first two entries, with a “normal” entry preceding them. There is negligible value in providing a way to determine what the implementation does because the code to deal with dot and dot-dot must be written in any case and because such a flag would add to the list of those flags (which has proven in itself to be objectionable) and might be abused.

Since the structure and buffer allocation, if any, for directory operations are defined by the implementation, this volume of POSIX.1-2008 imposes no portability requirements for erroneous program constructs, erroneous data, or the use of unspecified values such as the use or referencing of a *dirp* value or a **dirent** structure value after a directory stream has been closed or after a *fork()* or one of the *exec* function calls.

28414 **FUTURE DIRECTIONS**

28415 None.

28416 **SEE ALSO**28417 *closedir()*, *dirfd()*, *fstatat()*, *open()*, *readdir()*, *rewinddir()*, *symlink()*28418 XBD **<dirent.h>**, **<sys/types.h>**28419 **CHANGE HISTORY**

28420 First released in Issue 2.

28421 **Issue 6**28422 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.28423 The following new requirements on POSIX implementations derive from alignment with the  
28424 Single UNIX Specification:

- 28425 • The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was  
28426 required for conforming implementations of previous POSIX specifications, it was not  
28427 required for UNIX applications.
- 28428 • The [ELOOP] mandatory error condition is added.
- 28429 • A second [ENAMETOOLONG] is added as an optional error condition.

28430 The following changes were made to align with the IEEE P1003.1a draft standard:

- 28431 • The [ELOOP] optional error condition is added.

28432 **Issue 7**

28433 Austin Group Interpretation 1003.1-2001 #143 is applied.

28434 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

28435 The *fdopendir()* function is added from The Open Group Technical Standard, 2006, Extended API  
28436 Set Part 2.

28437 An additional example is added.

28438 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0122 [422] and XSH/TC1-2008/0123  
28439 [324] are applied.

28440 **NAME**28441        `feclearexcept` — clear floating-point exception28442 **SYNOPSIS**28443        `#include <fenv.h>`28444        `int feclearexcept(int excepts);`28445 **DESCRIPTION**

28446 **CX**        The functionality described on this reference page is aligned with the ISO C standard. Any  
28447 conflict between the requirements described here and the ISO C standard is unintentional. This  
28448 volume of POSIX.1-2008 defers to the ISO C standard.

28449        The `feclearexcept()` function shall attempt to clear the supported floating-point exceptions  
28450 represented by *excepts*.

28451 **RETURN VALUE**

28452        If the argument is zero or if all the specified exceptions were successfully cleared, `feclearexcept()`  
28453 shall return zero. Otherwise, it shall return a non-zero value.

28454 **ERRORS**

28455        No errors are defined.

28456 **EXAMPLES**

28457        None.

28458 **APPLICATION USAGE**

28459        None.

28460 **RATIONALE**

28461        None.

28462 **FUTURE DIRECTIONS**

28463        None.

28464 **SEE ALSO**28465        [\*fegetexceptflag\(\)\*](#), [\*feraiseexcept\(\)\*](#), [\*fetestexcept\(\)\*](#)28466        XBD [\*<fenv.h>\*](#)28467 **CHANGE HISTORY**

28468        First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

28469        ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

28470 **NAME**

28471 fegetenv, fesetenv — get and set current floating-point environment

28472 **SYNOPSIS**

```
28473 #include <fenv.h>
28474 int fegetenv(fenv_t *envp);
28475 int fesetenv(const fenv_t *envp);
```

28476 **DESCRIPTION**

28477 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
28478 conflict between the requirements described here and the ISO C standard is unintentional. This  
28479 volume of POSIX.1-2008 defers to the ISO C standard.

28480 The *fegetenv()* function shall attempt to store the current floating-point environment in the object  
28481 pointed to by *envp*.

28482 The *fesetenv()* function shall attempt to establish the floating-point environment represented by  
28483 the object pointed to by *envp*. The argument *envp* shall point to an object set by a call to  
28484 *fegetenv()* or *fehldexcept()*, or equal a floating-point environment macro. The *fesetenv()* function  
28485 does not raise floating-point exceptions, but only installs the state of the floating-point status  
28486 flags represented through its argument.

28487 **RETURN VALUE**

28488 If the representation was successfully stored, *fegetenv()* shall return zero. Otherwise, it shall  
28489 return a non-zero value. If the environment was successfully established, *fesetenv()* shall return  
28490 zero. Otherwise, it shall return a non-zero value.

28491 **ERRORS**

28492 No errors are defined.

28493 **EXAMPLES**

28494 None.

28495 **APPLICATION USAGE**

28496 None.

28497 **RATIONALE**

28498 None.

28499 **FUTURE DIRECTIONS**

28500 None.

28501 **SEE ALSO**28502 *fehldexcept()*, *feupdateenv()*28503 XBD **<fenv.h>**28504 **CHANGE HISTORY**

28505 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

28506 ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

28507 **NAME**

28508 fegetexceptflag, fesetexceptflag — get and set floating-point status flags

28509 **SYNOPSIS**

28510 #include &lt;fenv.h&gt;

28511 int fegetexceptflag(fexcept\_t \*flagp, int excepts);

28512 int fesetexceptflag(const fexcept\_t \*flagp, int excepts);

28513 **DESCRIPTION**

28514 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 28515 conflict between the requirements described here and the ISO C standard is unintentional. This  
 28516 volume of POSIX.1-2008 defers to the ISO C standard.

28517 The *fegetexceptflag()* function shall attempt to store an implementation-defined representation of  
 28518 the states of the floating-point status flags indicated by the argument *excepts* in the object  
 28519 pointed to by the argument *flagp*.

28520 The *fesetexceptflag()* function shall attempt to set the floating-point status flags indicated by the  
 28521 argument *excepts* to the states stored in the object pointed to by *flagp*. The value pointed to by  
 28522 *flagp* shall have been set by a previous call to *fegetexceptflag()* whose second argument  
 28523 represented at least those floating-point exceptions represented by the argument *excepts*. This  
 28524 function does not raise floating-point exceptions, but only sets the state of the flags.

28525 **RETURN VALUE**

28526 If the representation was successfully stored, *fegetexceptflag()* shall return zero. Otherwise, it  
 28527 shall return a non-zero value. If the *excepts* argument is zero or if all the specified exceptions  
 28528 were successfully set, *fesetexceptflag()* shall return zero. Otherwise, it shall return a non-zero  
 28529 value.

28530 **ERRORS**

28531 No errors are defined.

28532 **EXAMPLES**

28533 None.

28534 **APPLICATION USAGE**

28535 None.

28536 **RATIONALE**

28537 None.

28538 **FUTURE DIRECTIONS**

28539 None.

28540 **SEE ALSO**28541 *feclearexcept()*, *feraiseexcept()*, *fetestexcept()*

28542 XBD &lt;fenv.h&gt;

28543 **CHANGE HISTORY**

28544 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

28545 ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

**NAME**

fegetround, fesetround — get and set current rounding direction

**SYNOPSIS**

```
#include <fenv.h>

int fegetround(void);
int fesetround(int round);
```

**DESCRIPTION**

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *fegetround()* function shall get the current rounding direction.

The *fesetround()* function shall establish the rounding direction represented by its argument *round*. If the argument is not equal to the value of a rounding direction macro, the rounding direction is not changed.

**RETURN VALUE**

The *fegetround()* function shall return the value of the rounding direction macro representing the current rounding direction or a negative value if there is no such rounding direction macro or the current rounding direction is not determinable.

The *fesetround()* function shall return a zero value if and only if the requested rounding direction was established.

**ERRORS**

No errors are defined.

**EXAMPLES**

The following example saves, sets, and restores the rounding direction, reporting an error and aborting if setting the rounding direction fails:

```
#include <fenv.h>
#include <assert.h>
void f(int round_dir)
{
    #pragma STDC FENV_ACCESS ON
    int save_round;
    int setround_ok;
    save_round = fegetround();
    setround_ok = fesetround(round_dir);
    assert(setround_ok == 0);
    /* ... */
    fesetround(save_round);
    /* ... */
}
```

**APPLICATION USAGE**

None.

**RATIONALE**

None.

28589 **FUTURE DIRECTIONS**

28590 None.

28591 **SEE ALSO**

28592 XBD &lt;fenv.h&gt;

28593 **CHANGE HISTORY**

28594 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

28595 ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

28596 **NAME**

28597 feholdexcept — save current floating-point environment

28598 **SYNOPSIS**

28599 #include <fenv.h>

28600 int feholdexcept(fenv\_t \*envp);

28601 **DESCRIPTION**

28602 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
28603 conflict between the requirements described here and the ISO C standard is unintentional. This  
28604 volume of POSIX.1-2008 defers to the ISO C standard.

28605 The *feholdexcept()* function shall save the current floating-point environment in the object  
28606 pointed to by *envp*, clear the floating-point status flags, and then install a non-stop (continue on  
28607 floating-point exceptions) mode, if available, for all floating-point exceptions.

28608 **RETURN VALUE**

28609 The *feholdexcept()* function shall return zero if and only if non-stop floating-point exception  
28610 handling was successfully installed.

28611 **ERRORS**

28612 No errors are defined.

28613 **EXAMPLES**

28614 None.

28615 **APPLICATION USAGE**

28616 None.

28617 **RATIONALE**

28618 The *feholdexcept()* function should be effective on typical IEC 60559:1989 standard  
28619 implementations which have the default non-stop mode and at least one other mode for trap  
28620 handling or aborting. If the implementation provides only the non-stop mode, then installing the  
28621 non-stop mode is trivial.

28622 **FUTURE DIRECTIONS**

28623 None.

28624 **SEE ALSO**

28625 *fegetenv()*, *feupdateenv()*

28626 XBD <fenv.h>

28627 **CHANGE HISTORY**

28628 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.



28629 **NAME**

28630 feof — test end-of-file indicator on a stream

28631 **SYNOPSIS**

28632 #include &lt;stdio.h&gt;

28633 int feof(FILE \*stream);

28634 **DESCRIPTION**

28635 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 28636 conflict between the requirements described here and the ISO C standard is unintentional. This  
 28637 volume of POSIX.1-2008 defers to the ISO C standard.

28638 The *feof()* function shall test the end-of-file indicator for the stream pointed to by *stream*.28639 CX The *feof()* function shall not change the setting of *errno* if *stream* is valid.28640 **RETURN VALUE**28641 The *feof()* function shall return non-zero if and only if the end-of-file indicator is set for *stream*.28642 **ERRORS**

28643 No errors are defined.

28644 **EXAMPLES**

28645 None.

28646 **APPLICATION USAGE**

28647 None.

28648 **RATIONALE**

28649 None.

28650 **FUTURE DIRECTIONS**

28651 None.

28652 **SEE ALSO**28653 *clearerr()*, *ferror()*, *fopen()*

28654 XBD &lt;stdio.h&gt;

28655 **CHANGE HISTORY**

28656 First released in Issue 1. Derived from Issue 1 of the SVID.

28657 **Issue 7**

28658 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0124 [401] is applied.

28659 **NAME**

28660         feraiseexcept — raise floating-point exception

28661 **SYNOPSIS**

28662         #include <fenv.h>

28663         int feraiseexcept(int *excepts*);

28664 **DESCRIPTION**

28665 CX         The functionality described on this reference page is aligned with the ISO C standard. Any  
 28666         conflict between the requirements described here and the ISO C standard is unintentional. This  
 28667         volume of POSIX.1-2008 defers to the ISO C standard.

28668         The *feraiseexcept()* function shall attempt to raise the supported floating-point exceptions  
 28669         represented by the *excepts* argument. The order in which these floating-point exceptions are  
 28670 MX         raised is unspecified, except that if the *excepts* argument represents IEC 60559 valid coincident  
 28671         floating-point exceptions for atomic operations (namely overflow and inexact, or underflow and  
 28672         inexact), then overflow or underflow shall be raised before inexact. Whether the *feraiseexcept()*  
 28673         function additionally raises the inexact floating-point exception whenever it raises the overflow  
 28674         or underflow floating-point exception is implementation-defined.

28675 **RETURN VALUE**

28676         If the argument is zero or if all the specified exceptions were successfully raised, *feraiseexcept()*  
 28677         shall return zero. Otherwise, it shall return a non-zero value.

28678 **ERRORS**

28679         No errors are defined.

28680 **EXAMPLES**

28681         None.

28682 **APPLICATION USAGE**

28683         The effect is intended to be similar to that of floating-point exceptions raised by arithmetic  
 28684         operations. Hence, enabled traps for floating-point exceptions raised by this function are taken.

28685 **RATIONALE**

28686         Raising overflow or underflow is allowed to also raise inexact because on some architectures the  
 28687         only practical way to raise an exception is to execute an instruction that has the exception as a  
 28688         side-effect. The function is not restricted to accept only valid coincident expressions for atomic  
 28689         operations, so the function can be used to raise exceptions accrued over several operations.

28690 **FUTURE DIRECTIONS**

28691         None.

28692 **SEE ALSO**

28693         *feclearexcept()*, *fegetexceptflag()*, *fetestexcept()*

28694         XBD <fenv.h>

28695 **CHANGE HISTORY**

28696         First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

28697         ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

28698 **NAME**

28699       ferror — test error indicator on a stream

28700 **SYNOPSIS**

28701       #include &lt;stdio.h&gt;

28702       int ferror(FILE \**stream*);28703 **DESCRIPTION**

28704 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 28705       conflict between the requirements described here and the ISO C standard is unintentional. This  
 28706       volume of POSIX.1-2008 defers to the ISO C standard.

28707       The *ferror()* function shall test the error indicator for the stream pointed to by *stream*.28708 CX       The *ferror()* function shall not change the setting of *errno* if *stream* is valid.28709 **RETURN VALUE**28710       The *ferror()* function shall return non-zero if and only if the error indicator is set for *stream*.28711 **ERRORS**

28712       No errors are defined.

28713 **EXAMPLES**

28714       None.

28715 **APPLICATION USAGE**

28716       None.

28717 **RATIONALE**

28718       None.

28719 **FUTURE DIRECTIONS**

28720       None.

28721 **SEE ALSO**28722       *clearerr()*, *feof()*, *fopen()*

28723       XBD &lt;stdio.h&gt;

28724 **CHANGE HISTORY**

28725       First released in Issue 1. Derived from Issue 1 of the SVID.

28726 **Issue 7**

28727       POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0125 [401] is applied.

28728 **NAME**

28729       fesetenv — set current floating-point environment

28730 **SYNOPSIS**

28731       #include &lt;fenv.h&gt;

28732       int fesetenv(const fenv\_t \*envp);

28733 **DESCRIPTION**28734       Refer to *fegetenv()*.

28735 **NAME**

28736 fesetexceptflag — set floating-point status flags

28737 **SYNOPSIS**

28738 #include &lt;fenv.h&gt;

28739 int fesetexceptflag(const fexcept\_t \*flagp, int excepts);

28740 **DESCRIPTION**28741 Refer to *fegetexceptflag()*.

28742 **NAME**

28743       fesetround — set current rounding direction

28744 **SYNOPSIS**

28745       #include <fenv.h>

28746       int fesetround(int round);

28747 **DESCRIPTION**

28748       Refer to *fegetround()*.

28749 **NAME**

28750 fetestexcept — test floating-point exception flags

28751 **SYNOPSIS**

28752 #include &lt;fenv.h&gt;

28753 int fetestexcept(int *excepts*);28754 **DESCRIPTION**

28755 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
28756 conflict between the requirements described here and the ISO C standard is unintentional. This  
28757 volume of POSIX.1-2008 defers to the ISO C standard.

28758 The *fetestexcept()* function shall determine which of a specified subset of the floating-point  
28759 exception flags are currently set. The *excepts* argument specifies the floating-point status flags to  
28760 be queried.

28761 **RETURN VALUE**

28762 The *fetestexcept()* function shall return the value of the bitwise-inclusive OR of the floating-point  
28763 exception macros corresponding to the currently set floating-point exceptions included in  
28764 *excepts*.

28765 **ERRORS**

28766 No errors are defined.

28767 **EXAMPLES**

28768 The following example calls function *f()* if an invalid exception is set, and then function *g()* if an  
28769 overflow exception is set:

```
28770 #include <fenv.h>
28771 /* ... */
28772 {
28773     #pragma STDC FENV_ACCESS ON
28774     int set_excepts;
28775     feclearexcept(FE_INVALID | FE_OVERFLOW);
28776     // maybe raise exceptions
28777     set_excepts = fetestexcept(FE_INVALID | FE_OVERFLOW);
28778     if (set_excepts & FE_INVALID) f();
28779     if (set_excepts & FE_OVERFLOW) g();
28780     /* ... */
28781 }
```

28782 **APPLICATION USAGE**

28783 None.

28784 **RATIONALE**

28785 None.

28786 **FUTURE DIRECTIONS**

28787 None.

28788 **SEE ALSO**28789 *feclearexcept()*, *fegetexceptflag()*, *feraiseexcept()*

28790 XBD &lt;fenv.h&gt;

## 28791

28792



28793 **NAME**

28794 feupdateenv — update floating-point environment

28795 **SYNOPSIS**

28796 #include &lt;fenv.h&gt;

28797 int feupdateenv(const fenv\_t \*envp);

28798 **DESCRIPTION**

28799 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
28800 conflict between the requirements described here and the ISO C standard is unintentional. This  
28801 volume of POSIX.1-2008 defers to the ISO C standard.

28802 The *feupdateenv()* function shall attempt to save the currently raised floating-point exceptions in  
28803 its automatic storage, attempt to install the floating-point environment represented by the object  
28804 pointed to by *envp*, and then attempt to raise the saved floating-point exceptions. The argument  
28805 *envp* shall point to an object set by a call to *feholdexcept()* or *fegetenv()*, or equal a floating-point  
28806 environment macro.

28807 **RETURN VALUE**

28808 The *feupdateenv()* function shall return a zero value if and only if all the required actions were  
28809 successfully carried out.

28810 **ERRORS**

28811 No errors are defined.

28812 **EXAMPLES**

28813 The following example shows sample code to hide spurious underflow floating-point  
28814 exceptions:

```
28815 #include <fenv.h>
28816 double f(double x)
28817 {
28818     #pragma STDC FENV_ACCESS ON
28819     double result;
28820     fenv_t save_env;
28821     feholdexcept(&save_env);
28822     // compute result
28823     if (/* test spurious underflow */)
28824         feclearexcept(FE_UNDERFLOW);
28825     feupdateenv(&save_env);
28826     return result;
28827 }
```

28828 **APPLICATION USAGE**

28829 None.

28830 **RATIONALE**

28831 None.

28832 **FUTURE DIRECTIONS**

28833 None.

28834 **SEE ALSO**28835 *fegetenv()*, *feholdexcept()*

28836 XBD &lt;fenv.h&gt;

28837

CHANGE HISTORY

28838

First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

28839

ISO/IEC 9899: 1999 standard, Technical Corrigendum 1 is incorporated.

28840 **NAME**

28841 fexecve — execute a file

28842 **SYNOPSIS**

28843 #include &lt;unistd.h&gt;

28844 int fexecve(int *fd*, char \*const *argv*[], char \*const *envp*[]);28845 **DESCRIPTION**28846 Refer to *exec*.

28847 **NAME**

28848 fflush — flush a stream

28849 **SYNOPSIS**

28850 #include &lt;stdio.h&gt;

28851 int fflush(FILE \*stream);

28852 **DESCRIPTION**

28853 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 28854 conflict between the requirements described here and the ISO C standard is unintentional. This  
 28855 volume of POSIX.1-2008 defers to the ISO C standard.

28856 If *stream* points to an output stream or an update stream in which the most recent operation was  
 28857 CX not input, *fflush()* shall cause any unwritten data for that stream to be written to the file, and the  
 28858 last data modification and last file status change timestamps of the underlying file shall be  
 28859 marked for update.

28860 For a stream open for reading with an underlying file description, if the file is not already at +  
 28861 EOF, and the file is one capable of seeking, the file offset of the underlying open file description +  
 28862 shall be set to the file position of the stream, and any characters pushed back onto the stream by +  
 28863 *ungetc()* or *ungetwc()* that have not subsequently been read from the stream shall be discarded +  
 28864 (without further changing the file offset).

28865 If *stream* is a null pointer, *fflush()* shall perform this flushing action on all streams for which the  
 28866 behavior is defined above. -

28867 **RETURN VALUE**

28868 Upon successful completion, *fflush()* shall return 0; otherwise, it shall set the error indicator for  
 28869 CX the stream, return EOF, and set *errno* to indicate the error.

28870 **ERRORS**28871 The *fflush()* function shall fail if:

28872 CX [EAGAIN] The O\_NONBLOCK flag is set for the file descriptor underlying *stream* and  
 28873 the thread would be delayed in the write operation.

28874 CX [EBADF] The file descriptor underlying *stream* is not valid.

28875 CX [EFBIG] An attempt was made to write a file that exceeds the maximum file size.

28876 XSI [EFBIG] An attempt was made to write a file that exceeds the file size limit of the  
 28877 process.

28878 CX [EFBIG] The file is a regular file and an attempt was made to write at or beyond the  
 28879 offset maximum associated with the corresponding stream.

28880 CX [EINTR] The *fflush()* function was interrupted by a signal.

28881 CX [EIO] The process is a member of a background process group attempting to write to  
 28882 its controlling terminal, TOSTOP is set, the calling thread is not blocking  
 28883 SIGTTOU, the process is not ignoring SIGTTOU, and the process group of the  
 28884 process is orphaned. This error may also be returned under implementation-  
 28885 defined conditions.

28886 CX [ENOMEM] The underlying stream was created by *open\_memstream()* or  
 28887 *open\_wmemstream()* and insufficient memory is available.

28888 CX [ENOSPC] There was no free space remaining on the device containing the file or in the  
 28889 buffer used by the *fmemopen()* function.

28890 CX [EPIPE] An attempt is made to write to a pipe or FIFO that is not open for reading by  
 28891 any process. A SIGPIPE signal shall also be sent to the thread.

28892 The *fflush()* function may fail if:

28893 CX [ENXIO] A request was made of a nonexistent device, or the request was outside the  
 28894 capabilities of the device.

## 28895 EXAMPLES

### 28896 Sending Prompts to Standard Output

28897 The following example uses *printf()* calls to print a series of prompts for information the user  
 28898 must enter from standard input. The *fflush()* calls force the output to standard output. The  
 28899 *fflush()* function is used because standard output is usually buffered and the prompt may not  
 28900 immediately be printed on the output or terminal. The *getline()* function calls read strings from  
 28901 standard input and place the results in variables, for use later in the program.

```

28902 char *user;
28903 char *oldpasswd;
28904 char *newpasswd;
28905 ssize_t llen;
28906 size_t blen;
28907 struct termios term;
28908 tcflag_t saveflag;

28909 printf("User name: ");
28910 fflush(stdout);
28911 blen = 0;
28912 llen = getline(&user, &blen, stdin);
28913 user[llen-1] = 0;
28914 tcgetattr(fileno(stdin), &term);
28915 saveflag = term.c_lflag;
28916 term.c_lflag &= ~ECHO;
28917 tcsetattr(fileno(stdin), TCSANOW, &term);
28918 printf("Old password: ");
28919 fflush(stdout);
28920 blen = 0;
28921 llen = getline(&oldpasswd, &blen, stdin);
28922 oldpasswd[llen-1] = 0;

28923 printf("\nNew password: ");
28924 fflush(stdout);
28925 blen = 0;
28926 llen = getline(&newpasswd, &blen, stdin);
28927 newpasswd[llen-1] = 0;
28928 term.c_lflag = saveflag;
28929 tcsetattr(fileno(stdin), TCSANOW, &term);
28930 free(user);
28931 free(oldpasswd);
28932 free(newpasswd);

```

**28933 APPLICATION USAGE**

28934 None.

**28935 RATIONALE**

28936 Data buffered by the system may make determining the validity of the position of the current  
28937 file descriptor impractical. Thus, enforcing the repositioning of the file descriptor after *fflush()*  
28938 on streams open for *read()* is not mandated by POSIX.1-2008.

**28939 FUTURE DIRECTIONS**

28940 None.

**28941 SEE ALSO**

28942 [Section 2.5](#) (on page 495), *fmemopen()*, *getrlimit()*, *open\_memstream()*, *ulimit()*

28943 XBD [<stdio.h>](#)

**28944 CHANGE HISTORY**

28945 First released in Issue 1. Derived from Issue 1 of the SVID.

**28946 Issue 5**

28947 Large File Summit extensions are added.

**28948 Issue 6**

28949 Extensions beyond the ISO C standard are marked.

28950 The following new requirements on POSIX implementations derive from alignment with the  
28951 Single UNIX Specification:

- 28952 • The [EFBIG] error is added as part of the large file support extensions.
- 28953 • The [ENXIO] optional error condition is added.

28954 The RETURN VALUE section is updated to note that the error indicator shall be set for the  
28955 stream. This is for alignment with the ISO/IEC 9899:1999 standard.

28956 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/31 is applied, updating the [EAGAIN]  
28957 error in the ERRORS section from “the process would be delayed” to “the thread would be  
28958 delayed”.

**28959 Issue 7**

28960 Austin Group Interpretation 1003.1-2001 #002 is applied, clarifying the interaction of file  
28961 descriptors and streams.

28962 The [ENOSPC] error condition is updated and the [ENOMEM] error is added from The Open  
28963 Group Technical Standard, 2006, Extended API Set Part 1.

28964 The EXAMPLES section is revised.

28965 Changes are made related to support for finegrained timestamps.

28966 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0126 [87], XSH/TC1-2008/0127 [79],  
28967 and XSH/TC1-2008/0128 [14] are applied.

28968 **NAME**

28969       ffs — find first set bit

28970 **SYNOPSIS**

```
28971 XSI      #include <strings.h>
28972          int ffs(int i);
```

28973 **DESCRIPTION**

28974       The `ffs()` function shall find the first bit set (beginning with the least significant bit) in *i*, and  
 28975       return the index of that bit. Bits are numbered starting at one (the least significant bit).

28976 **RETURN VALUE**

28977       The `ffs()` function shall return the index of the first bit set. If *i* is 0, then `ffs()` shall return 0.

28978 **ERRORS**

28979       No errors are defined.

28980 **EXAMPLES**

28981       None.

28982 **APPLICATION USAGE**

28983       None.

28984 **RATIONALE**

28985       None.

28986 **FUTURE DIRECTIONS**

28987       None.

28988 **SEE ALSO**

28989       XBD [<strings.h>](#)

28990 **CHANGE HISTORY**

28991       First released in Issue 4, Version 2.

28992 **Issue 5**

28993       Moved from X/OPEN UNIX extension to BASE.

28994 **NAME**

28995       fgetc — get a byte from a stream

28996 **SYNOPSIS**

28997       #include &lt;stdio.h&gt;

28998       int fgetc(FILE \*stream);

28999 **DESCRIPTION**

29000 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 29001 conflict between the requirements described here and the ISO C standard is unintentional. This  
 29002 volume of POSIX.1-2008 defers to the ISO C standard.

29003       If the end-of-file indicator for the input stream pointed to by *stream* is not set and a next byte is  
 29004 present, the *fgetc()* function shall obtain the next byte as an **unsigned char** converted to an **int**,  
 29005 from the input stream pointed to by *stream*, and advance the associated file position indicator for  
 29006 the stream (if defined). Since *fgetc()* operates on bytes, reading a character consisting of multiple  
 29007 bytes (or “a multi-byte character”) may require multiple calls to *fgetc()*.

29008 CX       The *fgetc()* function may mark the last data access timestamp of the file associated with *stream*  
 29009 for update. The last data access timestamp shall be marked for update by the first successful  
 29010 execution of *fgetc()*, *fgets()*, *fread()*, *fscanf()*, *getc()*, *getchar()*, *getdelim()*, *getline()*, *gets()*, or  
 29011 *scanf()* using *stream* that returns data not supplied by a prior call to *ungetc()*.

29012 **RETURN VALUE**

29013       Upon successful completion, *fgetc()* shall return the next byte from the input stream pointed to  
 29014 by *stream*. If the end-of-file indicator for the stream is set, or if the stream is at end-of-file, the  
 29015 end-of-file indicator for the stream shall be set and *fgetc()* shall return EOF. If a read error occurs,  
 29016 CX       the error indicator for the stream shall be set, *fgetc()* shall return EOF, and shall set *errno* to  
 29017 indicate the error.

29018 **ERRORS**29019       The *fgetc()* function shall fail if data needs to be read and:

29020 CX       [EAGAIN]       The O\_NONBLOCK flag is set for the file descriptor underlying *stream* and  
 29021 the thread would be delayed in the *fgetc()* operation.

29022 CX       [EBADF]       The file descriptor underlying *stream* is not a valid file descriptor open for  
 29023 reading.

29024 CX       [EINTR]       The read operation was terminated due to the receipt of a signal, and no data  
 29025 was transferred.

29026 CX       [EIO]        A physical I/O error has occurred, or the process is in a background process  
 29027 group attempting to read from its controlling terminal, and either the calling  
 29028 thread is blocking SIGTIN or the process is ignoring SIGTIN or the process  
 29029 group of the process is orphaned. This error may also be generated for  
 29030 implementation-defined reasons.

29031 CX       [EOVERFLOW]   The file is a regular file and an attempt was made to read at or beyond the  
 29032 offset maximum associated with the corresponding stream.

29033       The *fgetc()* function may fail if:

29034 CX       [ENOMEM]       Insufficient storage space is available.

29035 CX       [ENXIO]        A request was made of a nonexistent device, or the request was outside the  
 29036 capabilities of the device.



**EXAMPLES**

None.

**APPLICATION USAGE**

If the integer value returned by *fgetc()* is stored into a variable of type **char** and then compared against the integer constant EOF, the comparison may never succeed, because sign-extension of a variable of type **char** on widening to integer is implementation-defined.

The *ferror()* or *feof()* functions must be used to distinguish between an error condition and an end-of-file condition.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

Section 2.5 (on page 495), *feof()*, *ferror()*, *fgets()*, *fread()*, *fscanf()*, *getchar()*, *getc()*, *gets()*, *ungetc()*

XBD <stdio.h>

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

Large File Summit extensions are added.

**Issue 6**

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EIO] and [EOVERFLOW] mandatory error conditions are added.
- The [ENOMEM] and [ENXIO] optional error conditions are added.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The DESCRIPTION is updated to clarify the behavior when the end-of-file indicator for the input stream is not set.
- The RETURN VALUE section is updated to note that the error indicator shall be set for the stream.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/32 is applied, updating the [EAGAIN] error in the ERRORS section from “the process would be delayed” to “the thread would be delayed”.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #051 is applied, updating the list of functions that mark the last data access timestamp for update.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0129 [79] and XSH/TC1-2008/0130 [14] are applied.

29075 **NAME**29076 `fgetpos` — get current file position information29077 **SYNOPSIS**29078 `#include <stdio.h>`29079 `int fgetpos(FILE *restrict stream, fpos_t *restrict pos);`29080 **DESCRIPTION**

29081 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 29082 conflict between the requirements described here and the ISO C standard is unintentional. This  
 29083 volume of POSIX.1-2008 defers to the ISO C standard.

29084 The `fgetpos()` function shall store the current values of the parse state (if any) and file position  
 29085 indicator for the stream pointed to by `stream` in the object pointed to by `pos`. The value stored  
 29086 contains unspecified information usable by `fsetpos()` for repositioning the stream to its position  
 29087 at the time of the call to `fgetpos()`.

29088 The `fgetpos()` function shall not change the setting of `errno` if successful.

29089 **RETURN VALUE**

29090 Upon successful completion, `fgetpos()` shall return 0; otherwise, it shall return a non-zero value  
 29091 and set `errno` to indicate the error.

29092 **ERRORS**

29093 The `fgetpos()` function shall fail if:

29094 CX [EBADF] The file descriptor underlying `stream` is not valid.

29095 CX [EOVERFLOW] The current value of the file position cannot be represented correctly in an  
 29096 object of type `fpos_t`.

29097 CX [ESPIPE] The file descriptor underlying `stream` is associated with a pipe, FIFO, or socket.

29098 **EXAMPLES**

29099 None.

29100 **APPLICATION USAGE**

29101 None.

29102 **RATIONALE**

29103 None.

29104 **FUTURE DIRECTIONS**

29105 None.

29106 **SEE ALSO**

29107 [Section 2.5](#) (on page 495), `fopen()`, `ftell()`, `rewind()`, `ungetc()`

29108 XBD `<stdio.h>`

29109 **CHANGE HISTORY**

29110 First released in Issue 4. Derived from the ISO C standard.

29111 **Issue 5**

29112 Large File Summit extensions are added.

29113 **Issue 6**

29114 Extensions beyond the ISO C standard are marked.

29115 The following new requirements on POSIX implementations derive from alignment with the  
29116 Single UNIX Specification:

- 29117 • The [EBADF] and [ESPIPE] optional error conditions are added.

29118 An additional [ESPIPE] error condition is added for sockets.

29119 The prototype for *fgetpos()* is changed for alignment with the ISO/IEC 9899:1999 standard.

29120 **Issue 7**

29121 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0131 [105], XSH/TC1-2008/0132 [122],  
29122 and XSH/TC1-2008/0133 [14] are applied.

29123 **NAME**

29124       fgets — get a string from a stream

29125 **SYNOPSIS**

29126       #include &lt;stdio.h&gt;

29127       char \*fgets(char \*restrict s, int n, FILE \*restrict stream);

29128 **DESCRIPTION**

29129 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 29130 conflict between the requirements described here and the ISO C standard is unintentional. This  
 29131 volume of POSIX.1-2008 defers to the ISO C standard.

29132       The *fgets()* function shall read bytes from *stream* into the array pointed to by *s* until *n*–1 bytes are  
 29133 read, or a <newline> is read and transferred to *s*, or an end-of-file condition is encountered. A  
 29134 null byte shall be written immediately after the last byte read into the array. If the end-of-file  
 29135 condition is encountered before any bytes are read, the contents of the array pointed to by *s* shall  
 29136 not be changed.

29137 CX       The *fgets()* function may mark the last data access timestamp of the file associated with *stream*  
 29138 for update. The last data access timestamp shall be marked for update by the first successful  
 29139 execution of *fgetc()*, *fgets()*, *fread()*, *fscanf()*, *getc()*, *getchar()*, *getdelim()*, *getline()*, *gets()*, or  
 29140 *scanf()* using *stream* that returns data not supplied by a prior call to *ungetc()*.

29141 **RETURN VALUE**

29142       Upon successful completion, *fgets()* shall return *s*. If the stream is at end-of-file, the end-of-file  
 29143 indicator for the stream shall be set and *fgets()* shall return a null pointer. If a read error occurs,  
 29144 CX       the error indicator for the stream shall be set, *fgets()* shall return a null pointer, and shall set  
 29145 *errno* to indicate the error.

29146 **ERRORS**29147       Refer to *fgetc()*.29148 **EXAMPLES**29149       **Reading Input**

29150       The following example uses *fgets()* to read lines of input. It assumes that the file it is reading is a  
 29151 text file and that lines in this text file are no longer than 16384 (or {LINE\_MAX} if it is less than  
 29152 16384 on the implementation where it is running) bytes long. (Note that the standard utilities  
 29153 have no line length limit if *sysconf*(\_SC\_LINE\_MAX) returns –1 without setting *errno*. This  
 29154 example assumes that *sysconf*(\_SC\_LINE\_MAX) will not fail.)

```

29155 #include <limits.h>
29156 #include <stdio.h>
29157 #include <unistd.h>
29158 #define MYLIMIT 16384

29159 char *line;
29160 int line_max;
29161 if (LINE_MAX >= MYLIMIT) {
29162     // Use maximum line size of MYLIMIT. If LINE_MAX is
29163     // bigger than our limit, sysconf() can't report a
29164     // smaller limit.
29165     line_max = MYLIMIT;
29166 } else {
29167     long limit = sysconf(_SC_LINE_MAX);
29168     line_max = (limit < 0 || limit > MYLIMIT) ? MYLIMIT : (int)limit;

```

```

29169     }
29170     // line_max + 1 leaves room for the null byte added by fgets().
29171     line = malloc(line_max + 1);
29172     if (line == NULL) {
29173         // out of space
29174         ...
29175         return error;
29176     }
29177     while (fgets(line, line_max + 1, fp) != NULL) {
29178         // Verify that a full line has been read ...
29179         // If not, report an error or prepare to treat the
29180         // next time through the loop as a read of a
29181         // continuation of the current line.
29182         ...
29183         // Process line ...
29184         ...
29185     }
29186     free(line);
29187     ...

```

29188 **APPLICATION USAGE**

29189 None.

29190 **RATIONALE**

29191 None.

29192 **FUTURE DIRECTIONS**

29193 None.

29194 **SEE ALSO**

29195 [Section 2.5](#) (on page 495), [fgetc\(\)](#), [fopen\(\)](#), [fread\(\)](#), [fscanf\(\)](#), [getc\(\)](#), [getchar\(\)](#), [getdelim\(\)](#), [gets\(\)](#),  
29196 [ungetc\(\)](#)

29197 XBD [<stdio.h>](#)

29198 **CHANGE HISTORY**

29199 First released in Issue 1. Derived from Issue 1 of the SVID.

29200 **Issue 6**

29201 Extensions beyond the ISO C standard are marked.

29202 The prototype for `fgets()` is changed for alignment with the ISO/IEC 9899:1999 standard.

29203 **Issue 7**

29204 Austin Group Interpretation 1003.1-2001 #051 is applied, updating the list of functions that mark  
29205 the last data access timestamp for update.

29206 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0134 [182] and XSH/TC1-2008/0135  
29207 [14] are applied.

29208 **NAME**

29209 fgetwc — get a wide-character code from a stream

29210 **SYNOPSIS**

29211 #include &lt;stdio.h&gt;

29212 #include &lt;wchar.h&gt;

29213 wint\_t fgetwc(FILE \*stream);

29214 **DESCRIPTION**

29215 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 29216 conflict between the requirements described here and the ISO C standard is unintentional. This  
 29217 volume of POSIX.1-2008 defers to the ISO C standard.

29218 The *fgetwc()* function shall obtain the next character (if present) from the input stream pointed to  
 29219 by *stream*, convert that to the corresponding wide-character code, and advance the associated file  
 29220 position indicator for the stream (if defined).

29221 If an error occurs, the resulting value of the file position indicator for the stream is unspecified.

29222 CX The *fgetwc()* function may mark the last data access timestamp of the file associated with *stream*  
 29223 for update. The last data access timestamp shall be marked for update by the first successful  
 29224 execution of *fgetwc()*, *fgetws()*, *fwscanf()*, *getwc()*, *getwchar()*, *vfwscanf()*, *vfwscanf()*, or *wscanf()*  
 29225 using *stream* that returns data not supplied by a prior call to *ungetwc()*.

29226 The *fgetwc()* function shall not change the setting of *errno* if successful.

29227 **RETURN VALUE**

29228 Upon successful completion, the *fgetwc()* function shall return the wide-character code of the  
 29229 character read from the input stream pointed to by *stream* converted to a type **wint\_t**. If the end-  
 29230 of-file indicator for the stream is set, or if the stream is at end-of-file, the end-of-file indicator for  
 29231 the stream shall be set and *fgetwc()* shall return WEOF. If a read error occurs, the error indicator  
 29232 CX for the stream shall be set, *fgetwc()* shall return WEOF, and shall set *errno* to indicate the error. If  
 29233 an encoding error occurs, the error indicator for the stream shall be set, *fgetwc()* shall return  
 29234 WEOF, and shall set *errno* to indicate the error.

29235 **ERRORS**

29236 The *fgetwc()* function shall fail if data needs to be read and:

29237 CX [EAGAIN] The O\_NONBLOCK flag is set for the file descriptor underlying *stream* and  
 29238 the thread would be delayed in the *fgetwc()* operation.

29239 CX [EBADF] The file descriptor underlying *stream* is not a valid file descriptor open for  
 29240 reading.

29241 [EILSEQ] The data obtained from the input stream does not form a valid character.

29242 CX [EINTR] The read operation was terminated due to the receipt of a signal, and no data  
 29243 was transferred.

29244 CX [EIO] A physical I/O error has occurred, or the process is in a background process  
 29245 group attempting to read from its controlling terminal, and either the calling  
 29246 thread is blocking SIGTTIN or the process is ignoring SIGTTIN or the process  
 29247 group of the process is orphaned. This error may also be generated for  
 29248 implementation-defined reasons.

29249 CX [EOVERFLOW] The file is a regular file and an attempt was made to read at or beyond the  
 29250 offset maximum associated with the corresponding stream.

29251 The *fgetwc()* function may fail if:

29252 CX [ENOMEM] Insufficient storage space is available.

29253 CX [ENXIO] A request was made of a nonexistent device, or the request was outside the  
29254 capabilities of the device.

## 29255 EXAMPLES

29256 None.

## 29257 APPLICATION USAGE

29258 The *ferror()* or *feof()* functions must be used to distinguish between an error condition and an  
29259 end-of-file condition.

## 29260 RATIONALE

29261 None.

## 29262 FUTURE DIRECTIONS

29263 None.

## 29264 SEE ALSO

29265 [Section 2.5](#) (on page 495), *feof()*, *ferror()*, *fopen()*

29266 XBD [<stdio.h>](#), [<wchar.h>](#)

## 29267 CHANGE HISTORY

29268 First released in Issue 4. Derived from the MSE working draft.

### 29269 Issue 5

29270 The Optional Header (OH) marking is removed from [<stdio.h>](#).

29271 Large File Summit extensions are added.

### 29272 Issue 6

29273 Extensions beyond the ISO C standard are marked.

29274 The following new requirements on POSIX implementations derive from alignment with the  
29275 Single UNIX Specification:

- 29276 • The [EIO] and [EOVERFLOW] mandatory error conditions are added.
- 29277 • The [ENOMEM] and [ENXIO] optional error conditions are added.

29278 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/33 is applied, updating the [EAGAIN]  
29279 error in the ERRORS section from “the process would be delayed” to “the thread would be  
29280 delayed”.

### 29281 Issue 7

29282 Austin Group Interpretation 1003.1-2001 #051 is applied, clarifying the RETURN VALUE section.

29283 Changes are made related to support for finegrained timestamps.

29284 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0136 [105], XSH/TC1-2008/0137 [79],  
29285 and XSH/TC1-2008/0138 [14] are applied.

29286 **NAME**

29287 fgetws — get a wide-character string from a stream

29288 **SYNOPSIS**

29289 #include &lt;stdio.h&gt;

29290 #include &lt;wchar.h&gt;

29291 wchar\_t \*fgetws(wchar\_t \*restrict ws, int n,

29292 FILE \*restrict stream);

29293 **DESCRIPTION**

29294 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 29295 conflict between the requirements described here and the ISO C standard is unintentional. This  
 29296 volume of POSIX.1-2008 defers to the ISO C standard.

29297 The *fgetws()* function shall read characters from the *stream*, convert these to the corresponding  
 29298 wide-character codes, place them in the **wchar\_t** array pointed to by *ws*, until *n*–1 characters are  
 29299 read, or a <newline> is read, converted, and transferred to *ws*, or an end-of-file condition is  
 29300 encountered. The wide-character string, *ws*, shall then be terminated with a null wide-character  
 29301 code.

29302 If an error occurs, the resulting value of the file position indicator for the stream is unspecified.

29303 CX The *fgetws()* function may mark the last data access timestamp of the file associated with *stream*  
 29304 for update. The last data access timestamp shall be marked for update by the first successful  
 29305 execution of *fgetwc()*, *fgetws()*, *fwscanf()*, *getwc()*, *getwchar()*, *vfwscanf()*, *vwscanf()*, or *wscanf()*  
 29306 using *stream* that returns data not supplied by a prior call to *ungetwc()*.

29307 **RETURN VALUE**

29308 Upon successful completion, *fgetws()* shall return *ws*. If the end-of-file indicator for the stream is  
 29309 set, or if the stream is at end-of-file, the end-of-file indicator for the stream shall be set and  
 29310 *fgetws()* shall return a null pointer. If a read error occurs, the error indicator for the stream shall  
 29311 CX be set, *fgetws()* shall return a null pointer, and shall set *errno* to indicate the error.

29312 **ERRORS**29313 Refer to *fgetwc()*.29314 **EXAMPLES**

29315 None.

29316 **APPLICATION USAGE**

29317 None.

29318 **RATIONALE**

29319 None.

29320 **FUTURE DIRECTIONS**

29321 None.

29322 **SEE ALSO**29323 Section 2.5 (on page 495), *fopen()*, *fread()*

29324 XBD &lt;stdio.h&gt;, &lt;wchar.h&gt;

29325 **CHANGE HISTORY**

29326 First released in Issue 4. Derived from the MSE working draft.



29327 **Issue 5**

29328 The Optional Header (OH) marking is removed from `<stdio.h>`.

29329 **Issue 6**

29330 Extensions beyond the ISO C standard are marked.

29331 The prototype for `fgetws()` is changed for alignment with the ISO/IEC 9899:1999 standard.

29332 **Issue 7**

29333 Austin Group Interpretation 1003.1-2001 #051 is applied, clarifying the RETURN VALUE section.

29334 Changes are made related to support for finegrained timestamps.

29335 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0139 [14] is applied.

## NAME

fileno — map a stream pointer to a file descriptor

## SYNOPSIS

```
CX      #include <stdio.h>
29340    int fileno(FILE *stream);
```

## DESCRIPTION

The *fileno()* function shall return the integer file descriptor associated with the stream pointed to by *stream*.

## RETURN VALUE

Upon successful completion, *fileno()* shall return the integer value of the file descriptor associated with *stream*. Otherwise, the value  $-1$  shall be returned and *errno* set to indicate the error.

## ERRORS

29348	The <i>fileno()</i> function shall fail if:	
29349		
29350	[EBADF]           The stream is not associated with a file.	+
29351	The <i>fileno()</i> function may fail if:	+
29352	[EBADF]           The file descriptor underlying <i>stream</i> is not a valid file descriptor.	

## EXAMPLES

None.

## APPLICATION USAGE

None.

## RATIONALE

Without some specification of which file descriptors are associated with these streams, it is impossible for an application to set up the streams for another application it starts with *fork()* and *exec*. In particular, it would not be possible to write a portable version of the *sh* command interpreter (although there may be other constraints that would prevent that portability).

## FUTURE DIRECTIONS

None.

## SEE ALSO

Section 2.5.1 (on page 497), *dirfd()*, *fdopen()*, *fopen()*, *stdin*  
XBD *<stdio.h>*

## CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

### Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EBADF] optional error condition is added.

### Issue 7

SD5-XBD-ERN-99 is applied, changing the definition of the [EBADF] error.

29375 **NAME**

29376 flockfile, ftrylockfile, funlockfile — stdio locking functions

29377 **SYNOPSIS**

```
29378 CX      #include <stdio.h>
29379          void flockfile(FILE *file);
29380          int ftrylockfile(FILE *file);
29381          void funlockfile(FILE *file);
```

29382 **DESCRIPTION**

29383 These functions shall provide for explicit application-level locking of stdio (**FILE \***) objects.  
 29384 These functions can be used by a thread to delineate a sequence of I/O statements that are  
 29385 executed as a unit.

29386 The *flockfile()* function shall acquire for a thread ownership of a (**FILE \***) object.

29387 The *ftrylockfile()* function shall acquire for a thread ownership of a (**FILE \***) object if the object is  
 29388 available; *ftrylockfile()* is a non-blocking version of *flockfile()*.

29389 The *funlockfile()* function shall relinquish the ownership granted to the thread. The behavior is  
 29390 undefined if a thread other than the current owner calls the *funlockfile()* function.

29391 The functions shall behave as if there is a lock count associated with each (**FILE \***) object. This  
 29392 count is implicitly initialized to zero when the (**FILE \***) object is created. The (**FILE \***) object is  
 29393 unlocked when the count is zero. When the count is positive, a single thread owns the (**FILE \***)  
 29394 object. When the *flockfile()* function is called, if the count is zero or if the count is positive and  
 29395 the caller owns the (**FILE \***) object, the count shall be incremented. Otherwise, the calling thread  
 29396 shall be suspended, waiting for the count to return to zero. Each call to *funlockfile()* shall  
 29397 decrement the count. This allows matching calls to *flockfile()* (or successful calls to *ftrylockfile()*)  
 29398 and *funlockfile()* to be nested.

29399 All functions that reference (**FILE \***) objects, except those with names ending in *\_unlocked*, shall  
 29400 behave as if they use *flockfile()* and *funlockfile()* internally to obtain ownership of these (**FILE \***)  
 29401 objects.

29402 **RETURN VALUE**

29403 None for *flockfile()* and *funlockfile()*.

29404 The *ftrylockfile()* function shall return zero for success and non-zero to indicate that the lock  
 29405 cannot be acquired.

29406 **ERRORS**

29407 No errors are defined.

29408 **EXAMPLES**

29409 None.

29410 **APPLICATION USAGE**

29411 Applications using these functions may be subject to priority inversion, as discussed in XBD  
 29412 [Section 3.291](#) (on page 80).

29413 A call to *exit()* can block until locked streams are unlocked because a thread having ownership +  
 29414 of a (**FILE\***) object blocks all function calls that reference that (**FILE\***) object (except those with +  
 29415 names ending in *\_unlocked*) from other threads, including calls to *exit()*.

**RATIONALE**

The *flockfile()* and *funlockfile()* functions provide an orthogonal mutual-exclusion lock for each **FILE**. The *ftrylockfile()* function provides a non-blocking attempt to acquire a file lock, analogous to *pthread\_mutex\_trylock()*.

These locks behave as if they are the same as those used internally by *stdio* for thread-safety. This both provides thread-safety of these functions without requiring a second level of internal locking and allows functions in *stdio* to be implemented in terms of other *stdio* functions.

Application developers and implementors should be aware that there are potential deadlock problems on **FILE** objects. For example, the line-buffered flushing semantics of *stdio* (requested via `{_IOLBF}`) require that certain input operations sometimes cause the buffered contents of implementation-defined line-buffered output streams to be flushed. If two threads each hold the lock on the other's **FILE**, deadlock ensues. This type of deadlock can be avoided by acquiring **FILE** locks in a consistent order. In particular, the line-buffered output stream deadlock can typically be avoided by acquiring locks on input streams before locks on output streams if a thread would be acquiring both.

In summary, threads sharing *stdio* streams with other threads can use *flockfile()* and *funlockfile()* to cause sequences of I/O performed by a single thread to be kept bundled. The only case where the use of *flockfile()* and *funlockfile()* is required is to provide a scope protecting uses of the *\*\_unlocked* functions/macros. This moves the cost/performance tradeoff to the optimal point.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*exit()*, *getc\_unlocked()*

XBD Section 3.291 (on page 80), `<stdio.h>`

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

**Issue 6**

These functions are marked as part of the Thread-Safe Functions option.

**Issue 7**

The *flockfile()*, *ftrylockfile()*, and *funlockfile()* functions are moved from the Thread-Safe Functions option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0140 [118] is applied.

29448 **NAME**

29449 floor, floorf, floorl — floor function

29450 **SYNOPSIS**

```
29451 #include <math.h>
29452 double floor(double x);
29453 float floorf(float x);
29454 long double floorl(long double x);
```

29455 **DESCRIPTION**

29456 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 29457 conflict between the requirements described here and the ISO C standard is unintentional. This  
 29458 volume of POSIX.1-2008 defers to the ISO C standard.

29459 These functions shall compute the largest integral value not greater than  $x$ .29460 **RETURN VALUE**29461 MX The result shall have the same sign as  $x$ .

29462 Upon successful completion, these functions shall return the largest integral value not greater  
 29463 than  $x$ , expressed as a **double**, **float**, or **long double**, as appropriate for the return type of the  
 29464 function.

29465 MX If  $x$  is NaN, a NaN shall be returned.29466 If  $x$  is  $\pm 0$  or  $\pm \text{Inf}$ ,  $x$  shall be returned.29467 **ERRORS**

29468 No errors are defined.

29469 **EXAMPLES**

29470 None.

29471 **APPLICATION USAGE**

29472 The integral value returned by these functions might not be expressible as an **intmax\_t**. The  
 29473 return value should be tested before assigning it to an integer type to avoid the undefined  
 29474 results of an integer overflow.

29475 These functions may raise the inexact floating-point exception if the result differs in value from  
 29476 the argument.

29477 **RATIONALE**

29478 None.

29479 **FUTURE DIRECTIONS**

29480 None.

29481 **SEE ALSO**29482 [ceil\(\)](#), [feclearexcept\(\)](#), [fetestexcept\(\)](#), [isnan\(\)](#)29483 [Section 4.20](#) (on page 117), [<math.h>](#)29484 **CHANGE HISTORY**

29485 First released in Issue 1. Derived from Issue 1 of the SVID.

29486 **Issue 5**

29487 The DESCRIPTION is updated to indicate how an application should check for an error. This  
 29488 text was previously published in the APPLICATION USAGE section.

29489 **Issue 6**

29490 The *floorf()* and *floorl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

29491 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
29492 revised to align with the ISO/IEC 9899:1999 standard.

29493 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
29494 marked.

29495 **Issue 7**

29496 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0141 [346] is applied.

29497 **NAME**

29498 fma, fmaf, fmal — floating-point multiply-add

29499 **SYNOPSIS**

```
29500 #include <math.h>
29501 double fma(double x, double y, double z);
29502 float fmaf(float x, float y, float z);
29503 long double fmal(long double x, long double y, long double z);
```

29504 **DESCRIPTION**

29505 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 29506 conflict between the requirements described here and the ISO C standard is unintentional. This  
 29507 volume of POSIX.1-2008 defers to the ISO C standard.

29508 These functions shall compute  $(x * y) + z$ , rounded as one ternary operation: they shall compute  
 29509 the value (as if) to infinite precision and round once to the result format, according to the  
 29510 rounding mode characterized by the value of FLT\_ROUNDS.

29511 An application wishing to check for error situations should set *errno* to zero and call  
 29512 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 29513 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 29514 zero, an error has occurred.

29515 **RETURN VALUE**

29516 Upon successful completion, these functions shall return  $(x * y) + z$ , rounded as one ternary  
 29517 operation.

29518 MX If the result overflows or underflows, a range error may occur. On systems that support the IEC  
 29519 60559 Floating-Point option, if the result overflows a range error shall occur.

29520 If *x* or *y* are NaN, a NaN shall be returned.

29521 If *x* multiplied by *y* is an exact infinity and *z* is also an infinity but with the opposite sign, a  
 29522 domain error shall occur, and either a NaN (if supported), or an implementation-defined value  
 29523 shall be returned.

29524 If one of *x* and *y* is infinite, the other is zero, and *z* is not a NaN, a domain error shall occur, and  
 29525 either a NaN (if supported), or an implementation-defined value shall be returned.

29526 If one of *x* and *y* is infinite, the other is zero, and *z* is a NaN, a NaN shall be returned and a  
 29527 domain error may occur.

29528 If  $x*y$  is not  $0*\text{Inf}$  nor  $\text{Inf}*0$  and *z* is a NaN, a NaN shall be returned.

29529 **ERRORS**

29530 These functions shall fail if:

29531	MX	<b>Domain Error</b>	The value of $x*y+z$ is invalid, or the value $x*y$ is invalid and <i>z</i> is not a NaN.  If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero, then <i>errno</i> shall be set to [EDOM]. If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the invalid floating-point exception shall be raised.
-------	----	---------------------	---

29536	MX	<b>Range Error</b>	The result overflows.  If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero, then <i>errno</i> shall be set to [ERANGE]. If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the overflow floating-point exception shall be raised.
-------	----	--------------------	--

29541 These functions may fail if:

29542	MX	<b>Domain Error</b>	The value $x*y$ is invalid and $z$ is a NaN.
29543			If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
29544			then <i>errno</i> shall be set to [EDOM]. If the integer expression $(\text{math\_errhandling}$
29545			$\ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the invalid floating-point exception
29546			shall be raised.
29547		<b>Range Error</b>	The result underflows.
29548			If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
29549			then <i>errno</i> shall be set to [ERANGE]. If the integer expression
29550			$(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the underflow
29551			floating-point exception shall be raised.
29552		<b>Range Error</b>	The result overflows.
29553			If the integer expression $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$ is non-zero,
29554			then <i>errno</i> shall be set to [ERANGE]. If the integer expression
29555			$(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$ is non-zero, then the overflow
29556			floating-point exception shall be raised.

## 29557 EXAMPLES

29558 None.

## 29559 APPLICATION USAGE

29560 On error, the expressions  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$  and  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$  are independent of each other, but at least one of them must be non-zero.

## 29562 RATIONALE

29563 In many cases, clever use of floating (*fused*) multiply-add leads to much improved code; but its  
 29564 unexpected use by the compiler can undermine carefully written code. The FP\_CONTRACT  
 29565 macro can be used to disallow use of floating multiply-add; and the *fma()* function guarantees  
 29566 its use where desired. Many current machines provide hardware floating multiply-add  
 29567 instructions; software implementation can be used for others.

## 29568 FUTURE DIRECTIONS

29569 None.

## 29570 SEE ALSO

29571 *feclearexcept()*, *fetestexcept()*

29572 XBD Section 4.20 (on page 117), **<math.h>**

## 29573 CHANGE HISTORY

29574 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

## 29575 Issue 7

29576 ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #57 (SD5-XSH-ERN-69) is applied,  
 29577 adding a “may fail” range error for non-MX systems.



29578 **NAME**

29579 fmax, fmaxf, fmaxl — determine maximum numeric value of two floating-point numbers

29580 **SYNOPSIS**

```
29581 #include <math.h>
29582 double fmax(double x, double y);
29583 float fmaxf(float x, float y);
29584 long double fmaxl(long double x, long double y);
```

29585 **DESCRIPTION**

29586 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 29587 conflict between the requirements described here and the ISO C standard is unintentional. This  
 29588 volume of POSIX.1-2008 defers to the ISO C standard.

29589 MX These functions shall determine the maximum numeric value of their arguments. NaN  
 29590 arguments shall be treated as missing data: if one argument is a NaN and the other numeric,  
 29591 then these functions shall choose the numeric value.

29592 **RETURN VALUE**

29593 Upon successful completion, these functions shall return the maximum numeric value of their  
 29594 arguments.

29595 MX If just one argument is a NaN, the other argument shall be returned.

29596 If *x* and *y* are NaN, a NaN shall be returned.

29597 **ERRORS**

29598 No errors are defined.

29599 **EXAMPLES**

29600 None.

29601 **APPLICATION USAGE**

29602 None.

29603 **RATIONALE**

29604 None.

29605 **FUTURE DIRECTIONS**

29606 None.

29607 **SEE ALSO**

29608 *fdim()*, *fmin()*

29609 XBD <math.h>

29610 **CHANGE HISTORY**

29611 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

29612 **Issue 7**

29613 Austin Group Interpretation 1003.1-2001 #007 is applied.

29614 **NAME**

29615 fmemopen — open a memory buffer stream

29616 **SYNOPSIS**

```
29617 CX    #include <stdio.h>
29618      FILE *fmemopen(void *restrict buf, size_t size,
29619                    const char *restrict mode);
```

29620 **DESCRIPTION**

29621 The *fmemopen()* function shall associate the buffer given by the *buf* and *size* arguments with a  
 29622 stream. The *buf* argument shall be either a null pointer or point to a buffer that is at least *size*  
 29623 bytes long.

29624 The *mode* argument points to a string. If the string is one of the following, the stream shall be  
 29625 opened in the indicated mode. Otherwise, the behavior is undefined.

- |       |           |  |
|-------|-----------|--|
| 29626 | <i>r</i>  | Open the stream for reading.   |
| 29627 | <i>w</i>  | Open the stream for writing.   |
| 29628 | <i>a</i>  | Append; open the stream for writing at the first null byte.                              |
| 29629 | <i>r+</i> | Open the stream for update (reading and writing).  |
| 29630 | <i>w+</i> | Open the stream for update (reading and writing). Truncate the buffer contents.          |
| 29631 | <i>a+</i> | Append; open the stream for update (reading and writing); the initial position is at the |
| 29632 |           | first null byte.   |

29633 Implementations shall accept all mode strings allowed by *fopen()*, but the use of the character  
 29634 'b' shall produce implementation-defined results, where the resulting **FILE \*** need not behave  
 29635 the same as if 'b' were omitted.

29636 If a null pointer is specified as the *buf* argument, *fmemopen()* shall allocate *size* bytes of memory  
 29637 as if by a call to *malloc()*. This buffer shall be automatically freed when the stream is closed.  
 29638 Because this feature is only useful when the stream is opened for updating (because there is no  
 29639 way to get a pointer to the buffer) the *fmemopen()* call may fail if the *mode* argument does not  
 29640 include a '+'.

29641 The stream shall maintain a current position in the buffer. This position shall be initially set to  
 29642 either the beginning of the buffer (for *r* and *w* modes) or to the first null byte in the buffer (for *a*  
 29643 modes). If no null byte is found in append mode, the initial position shall be set to one byte after  
 29644 the end of the buffer.

29645 If *buf* is a null pointer, the initial position shall always be set to the beginning of the buffer.

29646 The stream shall also maintain the size of the current buffer contents; use of *fseek()* or *fseeko()* on  
 29647 the stream with *SEEK\_END* shall seek relative to this size. For modes *r* and *r+* the size shall be  
 29648 set to the value given by the *size* argument. For modes *w* and *w+* the initial size shall be zero and  
 29649 for modes *a* and *a+* the initial size shall be:

- |       |  |  |
|-------|--|--|
| 29650 | • Zero, if <i>buf</i> is a null pointer  |  |
| 29651 | • The position of the first null byte in the buffer, if one is found                                   |  |
| 29652 | • The value of the <i>size</i> argument, if <i>buf</i> is not a null pointer and no null byte is found |  |

29653 A read operation on the stream shall not advance the current buffer position beyond the current  
 29654 buffer size. Reaching the buffer size in a read operation shall count as “end-of-file”. Null bytes in  
 29655 the buffer shall have no special meaning for reads. The read operation shall start at the current

29656 buffer position of the stream.

29657 A write operation shall start either at the current position of the stream (if *mode* has not specified  
29658 'a' as the first character) or at the current size of the stream (if *mode* had 'a' as the first  
29659 character). If the current position at the end of the write is larger than the current buffer size, the  
29660 current buffer size shall be set to the current position. A write operation on the stream shall not  
29661 advance the current buffer size beyond the size given in the *size* argument.

29662 When a stream open for writing is flushed or closed, a null byte shall be written at the current  
29663 position or at the end of the buffer, depending on the size of the contents. If a stream open for  
29664 update is flushed or closed and the last write has advanced the current buffer size, a null byte  
29665 shall be written at the end of the buffer if it fits.

29666 An attempt to seek a memory buffer stream to a negative position or to a position larger than the  
29667 buffer size given in the *size* argument shall fail.

## 29668 RETURN VALUE

29669 Upon successful completion, *fmemopen()* shall return a pointer to the object controlling the  
29670 stream. Otherwise, a null pointer shall be returned, and *errno* shall be set to indicate the error.

## 29671 ERRORS

29672 The *fmemopen()* function shall fail if:

29673 [EMFILE] {STREAM\_MAX} streams are currently open in the calling process. |

29674 The *fmemopen()* function may fail if:

29675 [EINVAL] The value of the *mode* argument is not valid.

29676 [EINVAL] The *buf* argument is a null pointer and the *mode* argument does not include a  
29677 '+' character.

29678 [EINVAL] The *size* argument specifies a buffer size of zero and the implementation does +  
29679 not support this.

29680 [ENOMEM] The *buf* argument is a null pointer and the allocation of a buffer of length *size*  
29681 has failed.

29682 [EMFILE] {FOPEN\_MAX} streams are currently open in the calling process.

## 29683 EXAMPLES

```
29684 #include <stdio.h>
29685 #include <string.h>
29686 static char buffer[] = "foobar";
29687 int
29688 main (void)
29689 {
29690     int ch;
29691     FILE *stream;
29692     stream = fmemopen(buffer, strlen (buffer), "r");
29693     if (stream == NULL)
29694         /* handle error */;
29695     while ((ch = fgetc(stream)) != EOF)
29696         printf("Got %c\n", ch);
29697     fclose(stream);
```

```
29698         return (0);
29699     }
```

29700 This program produces the following output:

```
29701     Got f
29702     Got o
29703     Got o
29704     Got b
29705     Got a
29706     Got r
```

#### 29707 APPLICATION USAGE

29708 None.

#### 29709 RATIONALE

29710 This interface has been introduced to eliminate many of the errors encountered in the  
29711 construction of strings, notably overflowing of strings. This interface prevents overflow.

#### 29712 FUTURE DIRECTIONS

29713 A future version of this standard may mandate specific behavior when the *mode* argument |  
29714 includes 'b'.

29715 A future version of this standard may require support of zero-length buffer streams explicitly. +

#### 29716 SEE ALSO

29717 *fdopen()*, *fopen()*, *freopen()*, *fseek()*, *malloc()*, *open\_memstream()*

29718 XBD <stdio.h>

#### 29719 CHANGE HISTORY

29720 First released in Issue 7.

29721 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0142 [461], XSH/TC1-2008/0143 [396],  
29722 XSH/TC1-2008/0144 [396], XSH/TC1-2008/0145 [461], XSH/TC1-2008/0146 [461],  
29723 XSH/TC1-2008/0147 [461], XSH/TC1-2008/0148 [461], XSH/TC1-2008/0149 [461], and  
29724 XSH/TC1-2008/0150 [396] are applied.

29725 **NAME**

29726 fmin, fminf, fminl — determine minimum numeric value of two floating-point numbers

29727 **SYNOPSIS**

29728 #include &lt;math.h&gt;

29729 double fmin(double x, double y);

29730 float fminf(float x, float y);

29731 long double fminl(long double x, long double y);

29732 **DESCRIPTION**

29733 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 29734 conflict between the requirements described here and the ISO C standard is unintentional. This  
 29735 volume of POSIX.1-2008 defers to the ISO C standard.

29736 MX These functions shall determine the minimum numeric value of their arguments. NaN  
 29737 arguments shall be treated as missing data: if one argument is a NaN and the other numeric,  
 29738 then these functions shall choose the numeric value.

29739 **RETURN VALUE**

29740 Upon successful completion, these functions shall return the minimum numeric value of their  
 29741 arguments.

29742 MX If just one argument is a NaN, the other argument shall be returned.

29743 If *x* and *y* are NaN, a NaN shall be returned.

29744 **ERRORS**

29745 No errors are defined.

29746 **EXAMPLES**

29747 None.

29748 **APPLICATION USAGE**

29749 None.

29750 **RATIONALE**

29751 None.

29752 **FUTURE DIRECTIONS**

29753 None.

29754 **SEE ALSO**29755 *fdim()*, *fmax()*

29756 XBD &lt;math.h&gt;

29757 **CHANGE HISTORY**

29758 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

29759 **Issue 7**

29760 Austin Group Interpretation 1003.1-2001 #008 is applied.

29761 **NAME**

29762 fmod, fmodf, fmodl — floating-point remainder value function

29763 **SYNOPSIS**

```
29764 #include <math.h>
29765 double fmod(double x, double y);
29766 float fmodf(float x, float y);
29767 long double fmodl(long double x, long double y);
```

29768 **DESCRIPTION**

29769 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 29770 conflict between the requirements described here and the ISO C standard is unintentional. This  
 29771 volume of POSIX.1-2008 defers to the ISO C standard.

29772 These functions shall return the floating-point remainder of the division of  $x$  by  $y$ .

29773 An application wishing to check for error situations should set *errno* to zero and call  
 29774 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 29775 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 29776 zero, an error has occurred.

29777 **RETURN VALUE**

29778 These functions shall return the value  $x - i * y$ , for some integer  $i$  such that, if  $y$  is non-zero, the  
 29779 result has the same sign as  $x$  and magnitude less than the magnitude of  $y$ .

29780 MXX If the correct value would cause underflow, and is not representable, a range error may occur,  
 29781 MXX and *fmod*(), *modf*(), and *fmodl*() shall return 0.0, or (if the IEC 60559 Floating-Point option is not  
 29782 supported) an implementation-defined value no greater in magnitude than DBL\_MIN,  
 29783 FLT\_MIN, and LDBL\_MIN, respectively.

29784 MX If  $x$  or  $y$  is NaN, a NaN shall be returned, and none of the conditions below shall be considered.

29785 If  $y$  is zero, a domain error shall occur, and a NaN shall be returned.

29786 If  $x$  is infinite, a domain error shall occur, and a NaN shall be returned.

29787 If  $x$  is  $\pm 0$  and  $y$  is not zero,  $\pm 0$  shall be returned.

29788 If  $x$  is not infinite and  $y$  is  $\pm \text{Inf}$ ,  $x$  shall be returned.

29789 MXX If the correct value would cause underflow, and is representable, a range error may occur and  
 29790 the correct value shall be returned.

29791 **ERRORS**

29792 These functions shall fail if:

29793 MX **Domain Error** The  $x$  argument is infinite or  $y$  is zero.

29794 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 29795 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 29796 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 29797 shall be raised.

29798 These functions may fail if:

29799 **Range Error** The result underflows.

29800 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 29801 then *errno* shall be set to [ERANGE]. If the integer expression  
 29802 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 29803 floating-point exception shall be raised.

29804 **EXAMPLES**

29805 None.

29806 **APPLICATION USAGE**

29807 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 29808 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

29809 **RATIONALE**

29810 None.

29811 **FUTURE DIRECTIONS**

29812 None.

29813 **SEE ALSO**29814 *feclearexcept()*, *fetestexcept()*, *isnan()*

29815 Section 4.20 (on page 117), &lt;math.h&gt;

29816 **CHANGE HISTORY**

29817 First released in Issue 1. Derived from Issue 1 of the SVID.

29818 **Issue 5**

29819 The DESCRIPTION is updated to indicate how an application should check for an error. This  
 29820 text was previously published in the APPLICATION USAGE section.

29821 **Issue 6**29822 The behavior for when the *y* argument is zero is now defined.

29823 The *fmodf()* and *fmodl()* functions are added for alignment with the ISO/IEC 9899:1999  
 29824 standard.

29825 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
 29826 revised to align with the ISO/IEC 9899:1999 standard.

29827 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
 29828 marked.

29829 **Issue 7**

29830 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0151 [68], XSH/TC1-2008/0152 [320],  
 29831 and XSH/TC1-2008/0153 [68] are applied.

29832 **NAME**

29833       fmtmsg — display a message in the specified format on standard error and/or a system console

29834 **SYNOPSIS**

```

29835 XSI      #include <fmtmsg.h>
29836
29837      int fmtmsg(long classification, const char *label, int severity,
                const char *text, const char *action, const char *tag);

```

29838 **DESCRIPTION**

29839       The *fmtmsg()* function shall display messages in a specified format instead of the traditional  
 29840       *printf()* function.

29841       Based on a message's classification component, *fmtmsg()* shall write a formatted message either  
 29842       to standard error, to the console, or to both.

29843       A formatted message consists of up to five components as defined below. The component  
 29844       *classification* is not part of a message displayed to the user, but defines the source of the message  
 29845       and directs the display of the formatted message.

29846       *classification*       Contains the sum of identifying values constructed from the constants defined  
 29847       below. Any one identifier from a subclass may be used in combination with a  
 29848       single identifier from a different subclass. Two or more identifiers from the  
 29849       same subclass should not be used together, with the exception of identifiers  
 29850       from the display subclass. (Both display subclass identifiers may be used so  
 29851       that messages can be displayed to both standard error and the system  
 29852       console.)

29853       **Major Classifications**

29854               Identifies the source of the condition. Identifiers are: MM\_HARD  
 29855               (hardware), MM\_SOFT (software), and MM\_FIRM (firmware).

29856       **Message Source Subclassifications**

29857               Identifies the type of software in which the problem is detected.  
 29858               Identifiers are: MM\_APPL (application), MM\_UTIL (utility), and  
 29859               MM\_OPSYS (operating system).

29860       **Display Subclassifications**

29861               Indicates where the message is to be displayed. Identifiers are:  
 29862               MM\_PRINT to display the message on the standard error stream,  
 29863               MM\_CONSOLE to display the message on the system console. One or  
 29864               both identifiers may be used.

29865       **Status Subclassifications**

29866               Indicates whether the application can recover from the condition.  
 29867               Identifiers are: MM\_RECOVER (recoverable) and MM\_NRECOV (non-  
 29868               recoverable).

29869       An additional identifier, MM\_NULLMC, indicates that no classification  
 29870       component is supplied for the message.

29871       *label*               Identifies the source of the message. The format is two fields separated by a  
 29872       <colon>. The first field is up to 10 bytes, the second is up to 14 bytes.

29873       *severity*           Indicates the seriousness of the condition. Identifiers for the levels of *severity*  
 29874       are:



29875		MM_HALT	Indicates that the application has encountered a severe fault and is halting. Produces the string "HALT".
29876			
29877		MM_ERROR	Indicates that the application has detected a fault. Produces the string "ERROR".
29878			
29879		MM_WARNING	Indicates a condition that is out of the ordinary, that might be a problem, and should be watched. Produces the string "WARNING".
29880			
29881			
29882		MM_INFO	Provides information about a condition that is not in error. Produces the string "INFO".
29883			
29884		MM_NOSEV	Indicates that no severity level is supplied for the message.
29885	<i>text</i>		Describes the error condition that produced the message. The character string is not limited to a specific size. If the character string is empty, then the text produced is unspecified.
29886			
29887			
29888	<i>action</i>		Describes the first step to be taken in the error-recovery process. The <i>fmtmsg()</i> function precedes the action string with the prefix: "TO FIX:". The <i>action</i> string is not limited to a specific size.
29889			
29890			
29891	<i>tag</i>		An identifier that references on-line documentation for the message. Suggested usage is that <i>tag</i> includes the <i>label</i> and a unique identifying number. A sample <i>tag</i> is "XSI:cat:146".
29892			
29893			

29894 The *MSGVERB* environment variable (for message verbosity) shall determine for *fmtmsg()*  
 29895 which message components it is to select when writing messages to standard error. The value of  
 29896 *MSGVERB* shall be a <colon>-separated list of optional keywords. Valid keywords are: *label*,  
 29897 *severity*, *text*, *action*, and *tag*. If *MSGVERB* contains a keyword for a component and the  
 29898 component's value is not the component's null value, *fmtmsg()* shall include that component in  
 29899 the message when writing the message to standard error. If *MSGVERB* does not include a  
 29900 keyword for a message component, that component shall not be included in the display of the  
 29901 message. The keywords may appear in any order. If *MSGVERB* is not defined, if its value is the  
 29902 null string, if its value is not of the correct format, or if it contains keywords other than the valid  
 29903 ones listed above, *fmtmsg()* shall select all components.

29904 *MSGVERB* shall determine which components are selected for display to standard error. All  
 29905 message components shall be included in console messages.

## 29906 RETURN VALUE

29907 The *fmtmsg()* function shall return one of the following values:

29908	MM_OK	The function succeeded.
29909	MM_NOTOK	The function failed completely.
29910	MM_NOMSG	The function was unable to generate a message on standard error, but otherwise succeeded.
29911		
29912	MM_NOCON	The function was unable to generate a console message, but otherwise succeeded.
29913		

## 29914 ERRORS

29915 None.

29916 **EXAMPLES**29917 1. The following example of *fmtmsg()*:29918 `fmtmsg(MM_PRINT, "XSI:cat", MM_ERROR, "illegal option",`  
29919 `"refer to cat in user's reference manual", "XSI:cat:001")`

29920 produces a complete message in the specified message format:

29921 `XSI:cat: ERROR: illegal option`29922 `TO FIX: refer to cat in user's reference manual XSI:cat:001`29923 2. When the environment variable *MSGVERB* is set as follows:29924 `MSGVERB=severity:text:action`29925 and Example 1 is used, *fmtmsg()* produces:29926 `ERROR: illegal option`29927 `TO FIX: refer to cat in user's reference manual`29928 **APPLICATION USAGE**29929 One or more message components may be systematically omitted from messages generated by  
29930 an application by using the null value of the argument for that component.29931 **RATIONALE**

29932 None.

29933 **FUTURE DIRECTIONS**

29934 None.

29935 **SEE ALSO**29936 *fprintf()*

29937 XBD &lt;fmtmsg.h&gt;

29938 **CHANGE HISTORY**

29939 First released in Issue 4, Version 2.

29940 **Issue 5**

29941 Moved from X/OPEN UNIX extension to BASE.

**NAME**

fnmatch — match a filename string or a pathname

**SYNOPSIS**

```
#include <fnmatch.h>
```

```
int fnmatch(const char *pattern, const char *string, int flags);
```

**DESCRIPTION**

The *fnmatch()* function shall match patterns as described in XCU [Section 2.13.1](#) (on page 2372) and [Section 2.13.2](#) (on page 2373). It checks the string specified by the *string* argument to see if it matches the pattern specified by the *pattern* argument.

The *flags* argument shall modify the interpretation of *pattern* and *string*. It is the bitwise-inclusive OR of zero or more of the flags defined in **<fnmatch.h>**. If the FNM\_PATHNAME flag is set in *flags*, then a <slash> character ( ' / ' ) in *string* shall be explicitly matched by a <slash> in *pattern*; it shall not be matched by either the <asterisk> or <question-mark> special characters, nor by a bracket expression. If the FNM\_PATHNAME flag is not set, the <slash> character shall be treated as an ordinary character.

If FNM\_NOESCAPE is not set in *flags*, a <backslash> character in *pattern* followed by any other character shall match that second character in *string*. In particular, "\\\" shall match a <backslash> in *string*. If *pattern* ends with an unescaped <backslash>, *fnmatch()* shall return a non-zero value (indicating either no match or an error). If FNM\_NOESCAPE is set, a <backslash> character shall be treated as an ordinary character.

If FNM\_PERIOD is set in *flags*, then a leading <period> ( ' . ' ) in *string* shall match a <period> in *pattern*; as described by rule 2 in XCU [Section 2.13.3](#) (on page 2373) where the location of “leading” is indicated by the value of FNM\_PATHNAME:

- If FNM\_PATHNAME is set, a <period> is “leading” if it is the first character in *string* or if it immediately follows a <slash>.
- If FNM\_PATHNAME is not set, a <period> is “leading” only if it is the first character of *string*.

If FNM\_PERIOD is not set, then no special restrictions are placed on matching a period.

**RETURN VALUE**

If *string* matches the pattern specified by *pattern*, then *fnmatch()* shall return 0. If there is no match, *fnmatch()* shall return FNM\_NOMATCH, which is defined in **<fnmatch.h>**. If an error occurs, *fnmatch()* shall return another non-zero value.

**ERRORS**

No errors are defined.

**EXAMPLES**

None.

**APPLICATION USAGE**

The *fnmatch()* function has two major uses. It could be used by an application or utility that needs to read a directory and apply a pattern against each entry. The *find* utility is an example of this. It can also be used by the *pax* utility to process its *pattern* operands, or by applications that need to match strings in a similar manner.

The name *fnmatch()* is intended to imply *filename* match, rather than *pathname* match. The default action of this function is to match filename strings, rather than pathnames, since it gives no special significance to the <slash> character. With the FNM\_PATHNAME flag, *fnmatch()* does match pathnames, but without tilde expansion, parameter expansion, or special treatment for a

29987 <period> at the beginning of a filename.

#### 29988 **RATIONALE**

29989 This function replaced the REG\_FILENAME flag of *regcomp()* in early proposals of this volume  
29990 of POSIX.1-2008. It provides virtually the same functionality as the *regcomp()* and *regexec()*  
29991 functions using the REG\_FILENAME and REG\_FSLASH flags (the REG\_FSLASH flag was  
29992 proposed for *regcomp()*, and would have had the opposite effect from FNM\_PATHNAME), but  
29993 with a simpler function and less system overhead.

#### 29994 **FUTURE DIRECTIONS**

29995 None.

#### 29996 **SEE ALSO**

29997 *glob()*, Section 2.6

29998 XBD <fnmatch.h>

#### 29999 **CHANGE HISTORY**

30000 First released in Issue 4. Derived from the ISO POSIX-2 standard.

#### 30001 **Issue 5**

30002 Moved from POSIX2 C-language Binding to BASE.

#### 30003 **Issue 7**

30004 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0154 [291] and XSH/TC1-2008/0155  
30005 [291] are applied.

30006 **NAME**

30007       fopen — open a stream

30008 **SYNOPSIS**

30009       #include &lt;stdio.h&gt;

30010       FILE \*fopen(const char \*restrict pathname, const char \*restrict mode);

30011 **DESCRIPTION**

30012 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 30013 conflict between the requirements described here and the ISO C standard is unintentional. This  
 30014 volume of POSIX.1-2008 defers to the ISO C standard.

30015       The *fopen()* function shall open the file whose pathname is the string pointed to by *pathname*,  
 30016 and associates a stream with it.

30017       The *mode* argument points to a string. If the string is one of the following, the file shall be opened  
 30018 in the indicated mode. Otherwise, the behavior is undefined.

30019	<i>r</i> or <i>rb</i>	Open file for reading.
30020	<i>w</i> or <i>wb</i>	Truncate to zero length or create file for writing.
30021	<i>a</i> or <i>ab</i>	Append; open or create file for writing at end-of-file.
30022	<i>r+</i> or <i>rb+</i> or <i>r+b</i>	Open file for update (reading and writing).
30023	<i>w+</i> or <i>wb+</i> or <i>w+b</i>	Truncate to zero length or create file for update.
30024	<i>a+</i> or <i>ab+</i> or <i>a+b</i>	Append; open or create file for update, writing at end-of-file.

30025 CX       The character 'b' shall have no effect, but is allowed for ISO C standard conformance. Opening  
 30026 a file with read mode (*r* as the first character in the *mode* argument) shall fail if the file does not  
 30027 exist or cannot be read.

30028       Opening a file with append mode (*a* as the first character in the *mode* argument) shall cause all  
 30029 subsequent writes to the file to be forced to the then current end-of-file, regardless of intervening  
 30030 calls to *fseek()*.

30031       When a file is opened with update mode ('+' as the second or third character in the *mode*  
 30032 argument), both input and output may be performed on the associated stream. However, the  
 30033 application shall ensure that output is not directly followed by input without an intervening call  
 30034 to *fflush()* or to a file positioning function (*fseek()*, *fsetpos()*, or *rewind()*), and input is not directly  
 30035 followed by output without an intervening call to a file positioning function, unless the input  
 30036 operation encounters end-of-file.

30037       When opened, a stream is fully buffered if and only if it can be determined not to refer to an  
 30038 interactive device. The error and end-of-file indicators for the stream shall be cleared.

30039 CX       If *mode* is *w*, *wb*, *a*, *ab*, *w+*, *wb+*, *w+b*, *a+*, *ab+*, or *a+b*, and the file did not previously exist, upon  
 30040 successful completion, *fopen()* shall mark for update the last data access, last data modification,  
 30041 and last file status change timestamps of the file and the last file status change and last data  
 30042 modification timestamps of the parent directory.

30043       If *mode* is *w*, *wb*, *a*, *ab*, *w+*, *wb+*, *w+b*, *a+*, *ab+*, or *a+b*, and the file did not previously exist, the  
 30044 *fopen()* function shall create a file as if it called the *creat()* function with a value appropriate for  
 30045 the *path* argument interpreted from *pathname* and a value of S\_IRUSR | S\_IWUSR | S\_IRGRP |  
 30046 S\_IWGRP | S\_IROTH | S\_IWOTH for the *mode* argument.

30047       If *mode* is *w*, *wb*, *w+*, *wb+*, or *w+b*, and the file did previously exist, upon successful completion,  
 30048 *fopen()* shall mark for update the last data modification and last file status change timestamps of

the file.

After a successful call to the *fopen()* function, the orientation of the stream shall be cleared, the encoding rule shall be cleared, and the associated **mbstate\_t** object shall be set to describe an initial conversion state.

The file descriptor associated with the opened stream shall be allocated and opened as if by a call to *open()* with the following flags:

<i>fopen()</i> Mode	<i>open()</i> Flags
<i>r</i> or <i>rb</i>	O_RDONLY
<i>w</i> or <i>wb</i>	O_WRONLY O_CREAT O_TRUNC
<i>a</i> or <i>ab</i>	O_WRONLY O_CREAT O_APPEND
<i>r+</i> or <i>rb+</i> or <i>r+b</i>	O_RDWR
<i>w+</i> or <i>wb+</i> or <i>w+b</i>	O_RDWR O_CREAT O_TRUNC
<i>a+</i> or <i>ab+</i> or <i>a+b</i>	O_RDWR O_CREAT O_APPEND

## RETURN VALUE

Upon successful completion, *fopen()* shall return a pointer to the object controlling the stream. Otherwise, a null pointer shall be returned, and *errno* shall be set to indicate the error.

## ERRORS

The *fopen()* function shall fail if:

**[EACCES]** Search permission is denied on a component of the path prefix, or the file exists and the permissions specified by *mode* are denied, or the file does not exist and write permission is denied for the parent directory of the file to be created.

**[EINTR]** A signal was caught during *fopen()*.

**[EISDIR]** The named file is a directory and *mode* requires write access.

**[ELOOP]** A loop exists in symbolic links encountered during resolution of the *path* argument.

**[EMFILE]** All file descriptors available to the process are currently open.

**[EMFILE]** {STREAM\_MAX} streams are currently open in the calling process.

**[ENAMETOOLONG]** The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH\_MAX}.

**[ENFILE]** The maximum allowable number of files is currently open in the system.

**[ENOENT]** The *mode* string begins with 'r' and a component of *pathname* does not name an existing file, or *mode* begins with 'w' or 'a' and a component of the path prefix of *pathname* does not name an existing file, or *pathname* is an empty string.

**[ENOENT] or [ENOTDIR]** The *pathname* argument contains at least one non-*<slash>* character and ends with one or more trailing *<slash>* characters. If *pathname* without the trailing *<slash>* characters would name an existing file, an [ENOENT] error shall not occur.

30091	CX	[ENOSPC]	The directory or file system that would contain the new file cannot be expanded, the file does not exist, and the file was to be created.
30092			
30093	CX	[ENOTDIR]	A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>pathname</i> argument contains at least one non- <code>&lt;slash&gt;</code> character and ends with one or more trailing <code>&lt;slash&gt;</code> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.
30094			
30095			
30096			
30097			
30098	CX	[ENXIO]	The named file is a character special or block special file, and the device associated with this special file does not exist.
30099			
30100	CX	[EOVERFLOW]	The named file is a regular file and the size of the file cannot be represented correctly in an object of type <code>off_t</code> .
30101			
30102	CX	[EROFS]	The named file resides on a read-only file system and <i>mode</i> requires write access.
30103			
30104			The <i>fopen()</i> function may fail if:
30105	CX	[EINVAL]	The value of the <i>mode</i> argument is not valid.
30106	CX	[ELOOP]	More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the <i>path</i> argument.
30107			
30108	CX	[EMFILE]	{FOPEN_MAX} streams are currently open in the calling process.
30109	CX	[ENAMETOOLONG]	
30110			The length of a component of a pathname is longer than {NAME_MAX}.
30111	CX	[ENOMEM]	Insufficient storage space is available.
30112	CX	[ETXTBSY]	The file is a pure procedure (shared text) file that is being executed and <i>mode</i> requires write access.
30113			

## 30114 EXAMPLES

### 30115 Opening a File

30116 The following example tries to open the file named **file** for reading. The *fopen()* function returns  
 30117 a file pointer that is used in subsequent *fgets()* and *fclose()* calls. If the program cannot open the  
 30118 file, it just ignores it.

```

30119 #include <stdio.h>
30120 ...
30121 FILE *fp;
30122 ...
30123 void rgrep(const char *file)
30124 {
30125     ...
30126     if ((fp = fopen(file, "r")) == NULL)
30127         return;
30128     ...
30129 }
```



**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

Section 2.5 (on page 495), *creat()*, *fclose()*, *fdopen()*, *fmemopen()*, *freopen()*, *open\_memstream()*

XBD <stdio.h>

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

Large File Summit extensions are added.

**Issue 6**

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, text is added to indicate setting of the offset maximum in the open file description. This change is to support large files.
- In the ERRORS section, the [Eoverflow] condition is added. This change is to support large files.
- The [ELOOP] mandatory error condition is added.
- The [EINVAL], [EMFILE], [ENAMETOOLONG], [ENOMEM], and [ETXTBSY] optional error conditions are added.

The normative text is updated to avoid use of the term “must” for application requirements.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The prototype for *fopen()* is updated.
- The DESCRIPTION is updated to note that if the argument *mode* points to a string other than those listed, then the behavior is undefined.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #025 is applied, clarifying the file creation mode.

Austin Group Interpretation 1003.1-2001 #143 is applied.

Austin Group Interpretation 1003.1-2001 #159 is applied, clarifying requirements for the flags set on the open file description.

SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

SD5-XSH-ERN-149 is applied, changing the {STREAM\_MAX} [EMFILE] error condition from a “may fail” to a “shall fail”.

Changes are made related to support for finegrained timestamps.



30170  
30171

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0156 [291,433], XSH/TC1-2008/0157 [146,433], XSH/TC1-2008/0158 [324], and XSH/TC1-2008/0159 [14] are applied.

30172 **NAME**

30173 fork — create a new process

30174 **SYNOPSIS**

30175 #include &lt;unistd.h&gt;

30176 pid\_t fork(void);

30177 **DESCRIPTION**

30178 The *fork()* function shall create a new process. The new process (child process) shall be an exact  
 30179 copy of the calling process (parent process) except as detailed below:

- 30180 • The child process shall have a unique process ID.
- 30181 • The child process ID also shall not match any active process group ID.
- 30182 • The child process shall have a different parent process ID, which shall be the process ID of  
 30183 the calling process.
- 30184 • The child process shall have its own copy of the parent's file descriptors. Each of the  
 30185 child's file descriptors shall refer to the same open file description with the corresponding  
 30186 file descriptor of the parent.
- 30187 • The child process shall have its own copy of the parent's open directory streams. Each  
 30188 open directory stream in the child process may share directory stream positioning with the  
 30189 corresponding directory stream of the parent.
- 30190 • The child process shall have its own copy of the parent's message catalog descriptors.
- 30191 • The child process values of *tms\_utime*, *tms\_stime*, *tms\_cutime*, and *tms\_cstime* shall be set to  
 30192 0.
- 30193 • The time left until an alarm clock signal shall be reset to zero, and the alarm, if any, shall be  
 30194 canceled; see *alarm()*.
- 30195 XSI • All *semadj* values shall be cleared.
- 30196 • File locks set by the parent process shall not be inherited by the child process.
- 30197 • The set of signals pending for the child process shall be initialized to the empty set.
- 30198 XSI • Interval timers shall be reset in the child process.
- 30199 • Any semaphores that are open in the parent process shall also be open in the child process.
- 30200 ML • The child process shall not inherit any address space memory locks established by the  
 30201 parent process via calls to *mlockall()* or *mlock()*.
- 30202 • Memory mappings created in the parent shall be retained in the child process.  
 30203 MAP\_PRIVATE mappings inherited from the parent shall also be MAP\_PRIVATE  
 30204 mappings in the child, and any modifications to the data in these mappings made by the  
 30205 parent prior to calling *fork()* shall be visible to the child. Any modifications to the data in  
 30206 MAP\_PRIVATE mappings made by the parent after *fork()* returns shall be visible only to  
 30207 the parent. Modifications to the data in MAP\_PRIVATE mappings made by the child shall  
 30208 be visible only to the child.
- 30209 PS • For the SCHED\_FIFO and SCHED\_RR scheduling policies, the child process shall inherit  
 30210 the policy and priority settings of the parent process during a *fork()* function. For other  
 30211 scheduling policies, the policy and priority settings on *fork()* are implementation-defined.

- 30212 • Per-process timers created by the parent shall not be inherited by the child process.
- 30213 MSG • The child process shall have its own copy of the message queue descriptors of the parent.
- 30214 Each of the message descriptors of the child shall refer to the same open message queue
- 30215 description as the corresponding message descriptor of the parent.
- 30216 • No asynchronous input or asynchronous output operations shall be inherited by the child
- 30217 process. Any use of asynchronous control blocks created by the parent produces undefined
- 30218 behavior.
- 30219 • A process shall be created with a single thread. If a multi-threaded process calls *fork()*, the
- 30220 new process shall contain a replica of the calling thread and its entire address space,
- 30221 possibly including the states of mutexes and other resources. Consequently, to avoid
- 30222 errors, the child process may only execute async-signal-safe operations until such time as
- 30223 one of the *exec* functions is called.

30224 When the application calls *fork()* from a signal handler and any of the fork handlers  
 30225 registered by *pthread\_atfork()* calls a function that is not async-signal-safe, the behavior is  
 30226 undefined.

- 30227 OB TRC TRI • If the Trace option and the Trace Inherit option are both supported:
- 30228 If the calling process was being traced in a trace stream that had its inheritance policy set
- 30229 to *POSIX\_TRACE\_INHERITED*, the child process shall be traced into that trace stream,
- 30230 and the child process shall inherit the parent's mapping of trace event names to trace event
- 30231 type identifiers. If the trace stream in which the calling process was being traced had its
- 30232 inheritance policy set to *POSIX\_TRACE\_CLOSE\_FOR\_CHILD*, the child process shall not
- 30233 be traced into that trace stream. The inheritance policy is set by a call to the
- 30234 *posix\_trace\_attr\_setinherited()* function.

- 30235 OB TRC • If the Trace option is supported, but the Trace Inherit option is not supported:
- 30236 The child process shall not be traced into any of the trace streams of its parent process.

- 30237 OB TRC • If the Trace option is supported, the child process of a trace controller process shall not
- 30238 control the trace streams controlled by its parent process.

- 30239 CPT • The initial value of the CPU-time clock of the child process shall be set to zero.
- 30240 TCT • The initial value of the CPU-time clock of the single thread of the child process shall be set
- 30241 to zero.

30242 All other process characteristics defined by POSIX.1-2008 shall be the same in the parent and  
 30243 child processes. The inheritance of process characteristics not defined by POSIX.1-2008 is  
 30244 unspecified by POSIX.1-2008.

30245 After *fork()*, both the parent and the child processes shall be capable of executing independently  
 30246 before either one terminates.

## 30247 RETURN VALUE

30248 Upon successful completion, *fork()* shall return 0 to the child process and shall return the  
 30249 process ID of the child process to the parent process. Both processes shall continue to execute  
 30250 from the *fork()* function. Otherwise, -1 shall be returned to the parent process, no child process  
 30251 shall be created, and *errno* shall be set to indicate the error.

## 30252 ERRORS

30253 The *fork()* function shall fail if:

30254 [EAGAIN] The system lacked the necessary resources to create another process, or the  
 30255 system-imposed limit on the total number of processes under execution  
 30256 system-wide or by a single user {CHILD\_MAX} would be exceeded.

30257 The *fork()* function may fail if:

30258 [ENOMEM] Insufficient storage space is available.

#### 30259 EXAMPLES

30260 None.

#### 30261 APPLICATION USAGE

30262 None.

#### 30263 RATIONALE

30264 Many historical implementations have timing windows where a signal sent to a process group  
 30265 (for example, an interactive SIGINT) just prior to or during execution of *fork()* is delivered to the  
 30266 parent following the *fork()* but not to the child because the *fork()* code clears the child's set of  
 30267 pending signals. This volume of POSIX.1-2008 does not require, or even permit, this behavior.  
 30268 However, it is pragmatic to expect that problems of this nature may continue to exist in  
 30269 implementations that appear to conform to this volume of POSIX.1-2008 and pass available  
 30270 verification suites. This behavior is only a consequence of the implementation failing to make  
 30271 the interval between signal generation and delivery totally invisible. From the application's  
 30272 perspective, a *fork()* call should appear atomic. A signal that is generated prior to the *fork()*  
 30273 should be delivered prior to the *fork()*. A signal sent to the process group after the *fork()* should  
 30274 be delivered to both parent and child. The implementation may actually initialize internal data  
 30275 structures corresponding to the child's set of pending signals to include signals sent to the  
 30276 process group during the *fork()*. Since the *fork()* call can be considered as atomic from the  
 30277 application's perspective, the set would be initialized as empty and such signals would have  
 30278 arrived after the *fork()*; see also <signal.h>.

30279 One approach that has been suggested to address the problem of signal inheritance across *fork()*  
 30280 is to add an [EINTR] error, which would be returned when a signal is detected during the call.  
 30281 While this is preferable to losing signals, it was not considered an optimal solution. Although it  
 30282 is not recommended for this purpose, such an error would be an allowable extension for an  
 30283 implementation.

30284 The [ENOMEM] error value is reserved for those implementations that detect and distinguish  
 30285 such a condition. This condition occurs when an implementation detects that there is not enough  
 30286 memory to create the process. This is intended to be returned when [EAGAIN] is inappropriate  
 30287 because there can never be enough memory (either primary or secondary storage) to perform  
 30288 the operation. Since *fork()* duplicates an existing process, this must be a condition where there is  
 30289 sufficient memory for one such process, but not for two. Many historical implementations  
 30290 actually return [ENOMEM] due to temporary lack of memory, a case that is not generally  
 30291 distinct from [EAGAIN] from the perspective of a conforming application.

30292 Part of the reason for including the optional error [ENOMEM] is because the SVID specifies it  
 30293 and it should be reserved for the error condition specified there. The condition is not applicable  
 30294 on many implementations.

30295 IEEE Std 1003.1-1988 neglected to require concurrent execution of the parent and child of *fork()*.  
 30296 A system that single-threads processes was clearly not intended and is considered an  
 30297 unacceptable "toy implementation" of this volume of POSIX.1-2008. The only objection  
 30298 anticipated to the phrase "executing independently" is testability, but this assertion should be  
 30299 testable. Such tests require that both the parent and child can block on a detectable action of the  
 30300 other, such as a write to a pipe or a signal. An interactive exchange of such actions should be

possible for the system to conform to the intent of this volume of POSIX.1-2008.

The [EAGAIN] error exists to warn applications that such a condition might occur. Whether it occurs or not is not in any practical sense under the control of the application because the condition is usually a consequence of the user's use of the system, not of the application's code. Thus, no application can or should rely upon its occurrence under any circumstances, nor should the exact semantics of what concept of "user" is used be of concern to the application developer. Validation writers should be cognizant of this limitation.

There are two reasons why POSIX programmers call *fork()*. One reason is to create a new thread of control within the same program (which was originally only possible in POSIX by creating a new process); the other is to create a new process running a different program. In the latter case, the call to *fork()* is soon followed by a call to one of the *exec* functions.

The general problem with making *fork()* work in a multi-threaded world is what to do with all of the threads. There are two alternatives. One is to copy all of the threads into the new process. This causes the programmer or implementation to deal with threads that are suspended on system calls or that might be about to execute system calls that should not be executed in the new process. The other alternative is to copy only the thread that calls *fork()*. This creates the difficulty that the state of process-local resources is usually held in process memory. If a thread that is not calling *fork()* holds a resource, that resource is never released in the child process because the thread whose job it is to release the resource does not exist in the child process.

When a programmer is writing a multi-threaded program, the first described use of *fork()*, creating new threads in the same program, is provided by the *pthread\_create()* function. The *fork()* function is thus used only to run new programs, and the effects of calling functions that require certain resources between the call to *fork()* and the call to an *exec* function are undefined.

The addition of the *forkall()* function to the standard was considered and rejected. The *forkall()* function lets all the threads in the parent be duplicated in the child. This essentially duplicates the state of the parent in the child. This allows threads in the child to continue processing and allows locks and the state to be preserved without explicit *pthread\_atfork()* code. The calling process has to ensure that the threads processing state that is shared between the parent and child (that is, file descriptors or MAP\_SHARED memory) behaves properly after *forkall()*. For example, if a thread is reading a file descriptor in the parent when *forkall()* is called, then two threads (one in the parent and one in the child) are reading the file descriptor after the *forkall()*. If this is not desired behavior, the parent process has to synchronize with such threads before calling *forkall()*.

While the *fork()* function is async-signal-safe, there is no way for an implementation to determine whether the fork handlers established by *pthread\_atfork()* are async-signal-safe. The fork handlers may attempt to execute portions of the implementation that are not async-signal-safe, such as those that are protected by mutexes, leading to a deadlock condition. It is therefore undefined for the fork handlers to execute functions that are not async-signal-safe when *fork()* is called from a signal handler.

When *forkall()* is called, threads, other than the calling thread, that are in functions that can return with an [EINTR] error may have those functions return [EINTR] if the implementation cannot ensure that the function behaves correctly in the parent and child. In particular, *pthread\_cond\_wait()* and *pthread\_cond\_timedwait()* need to return in order to ensure that the condition has not changed. These functions can be awakened by a spurious condition wakeup rather than returning [EINTR].

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*alarm()*, *exec*, *fcntl()*, *posix\_trace\_attr\_getinherited()*, *posix\_trace\_eventid\_equal()*, *pthread\_atfork()*, *semop()*, *signal()*, *times()*

XBD Section 4.12 (on page 111), *<sys/types.h>*, *<unistd.h>*

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

The DESCRIPTION is changed for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

**Issue 6**

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include *<sys/types.h>* has been removed. Although *<sys/types.h>* was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The effect of *fork()* on a pending alarm call in the child process is clarified.

The description of CPU-time clock semantics is added for alignment with IEEE Std 1003.1d-1999.

The description of tracing semantics is added for alignment with IEEE Std 1003.1q-2000.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/17 is applied, adding text to the DESCRIPTION and RATIONALE relating to fork handlers registered by the *pthread\_atfork()* function and async-signal safety.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #080 is applied, clarifying the status of asynchronous input and asynchronous output operations and asynchronous control lists in the DESCRIPTION.

Functionality relating to the Asynchronous Input and Output, Memory Mapped Files, Timers, and Threads options is moved to the Base.

Functionality relating to message catalog descriptors is moved from the XSI option to the Base.

**NAME**

fpathconf, pathconf — get configurable pathname variables

**SYNOPSIS**

```
#include <unistd.h>
```

```
long fpathconf(int fildes, int name);
```

```
long pathconf(const char *path, int name);
```

**DESCRIPTION**

The *fpathconf()* and *pathconf()* functions shall determine the current value of a configurable limit or option (*variable*) that is associated with a file or directory.

For *pathconf()*, the *path* argument points to the pathname of a file or directory.

For *fpathconf()*, the *fildes* argument is an open file descriptor.

The *name* argument represents the variable to be queried relative to that file or directory. Implementations shall support all of the variables listed in the following table and may support others. The variables in the following table come from **<limits.h>** or **<unistd.h>** and the symbolic constants, defined in **<unistd.h>**, are the corresponding values used for *name*.

Variable	Value of <i>name</i>	Requirements
{FILESIZEBITS}	_PC_FILESIZEBITS	4,7
{LINK_MAX}	_PC_LINK_MAX	1
{MAX_CANON}	_PC_MAX_CANON	2
{MAX_INPUT}	_PC_MAX_INPUT	2
{NAME_MAX}	_PC_NAME_MAX	3,4
{PATH_MAX}	_PC_PATH_MAX	4,5
{PIPE_BUF}	_PC_PIPE_BUF	6
{POSIX2_SYMLINKS}	_PC_2_SYMLINKS	4
{POSIX_ALLOC_SIZE_MIN}	_PC_ALLOC_SIZE_MIN	10
{POSIX_REC_INCR_XFER_SIZE}	_PC_REC_INCR_XFER_SIZE	10
{POSIX_REC_MAX_XFER_SIZE}	_PC_REC_MAX_XFER_SIZE	10
{POSIX_REC_MIN_XFER_SIZE}	_PC_REC_MIN_XFER_SIZE	10
{POSIX_REC_XFER_ALIGN}	_PC_REC_XFER_ALIGN	10
{SYMLINK_MAX}	_PC_SYMLINK_MAX	4,9
_POSIX_CHOWN_RESTRICTED	_PC_CHOWN_RESTRICTED	7
_POSIX_NO_TRUNC	_PC_NO_TRUNC	3,4
_POSIX_VDISABLE	_PC_VDISABLE	2
_POSIX_ASYNC_IO	_PC_ASYNC_IO	8
_POSIX_PRIO_IO	_PC_PRIO_IO	8
_POSIX_SYNC_IO	_PC_SYNC_IO	8
_POSIX_TIMESTAMP_RESOLUTION	_PC_TIMESTAMP_RESOLUTION	1

**Requirements**

1. If *path* or *fildes* refers to a directory, the value returned shall apply to the directory itself.
2. If *path* or *fildes* does not refer to a terminal file, it is unspecified whether an implementation supports an association of the variable name with the specified file.
3. If *path* or *fildes* refers to a directory, the value returned shall apply to filenames within the directory.



4. If *path* or *fildes* does not refer to a directory, it is unspecified whether an implementation supports an association of the variable name with the specified file.
5. If *path* or *fildes* refers to a directory, the value returned shall be the maximum length of a relative pathname that would not cross any mount points when the specified directory is the working directory.
6. If *path* refers to a FIFO, or *fildes* refers to a pipe or FIFO, the value returned shall apply to the referenced object. If *path* or *fildes* refers to a directory, the value returned shall apply to any FIFO that exists or can be created within the directory. If *path* or *fildes* refers to any other type of file, it is unspecified whether an implementation supports an association of the variable name with the specified file.
7. If *path* or *fildes* refers to a directory, the value returned shall apply to any files, other than directories, that exist or can be created within the directory.
8. If *path* or *fildes* refers to a directory, it is unspecified whether an implementation supports an association of the variable name with the specified file.
9. If *path* or *fildes* refers to a directory, the value returned shall be the maximum length of the string that a symbolic link in that directory can contain.
10. If *path* or *fildes* does not refer to a regular file, it is unspecified whether an implementation supports an association of the variable name with the specified file. If an implementation supports such an association for other than a regular file, the value returned is unspecified.

#### RETURN VALUE

If *name* is an invalid value, both *pathconf()* and *fpathconf()* shall return `-1` and set *errno* to indicate the error.

If the variable corresponding to *name* is described in `<limits.h>` as a maximum or minimum value and the variable has no limit for the path or file descriptor, both *pathconf()* and *fpathconf()* shall return `-1` without changing *errno*. Note that indefinite limits do not imply infinite limits; see `<limits.h>`.

If the implementation needs to use *path* to determine the value of *name* and the implementation does not support the association of *name* with the file specified by *path*, or if the process did not have appropriate privileges to query the file specified by *path*, or *path* does not exist, *pathconf()* shall return `-1` and set *errno* to indicate the error.

If the implementation needs to use *fildes* to determine the value of *name* and the implementation does not support the association of *name* with the file specified by *fildes*, or if *fildes* is an invalid file descriptor, *fpathconf()* shall return `-1` and set *errno* to indicate the error.

Otherwise, *pathconf()* or *fpathconf()* shall return the current variable value for the file or directory without changing *errno*. The value returned shall not be more restrictive than the corresponding value available to the application when it was compiled with the implementation's `<limits.h>` or `<unistd.h>`.

If the variable corresponding to *name* is dependent on an unsupported option, the results are unspecified.

#### ERRORS

The *pathconf()* function shall fail if:

[EINVAL]           The value of *name* is not valid.



- 30462 [EOVERFLOW] The value of *name* is `_PC_TIMESTAMP_RESOLUTION` and the resolution is  
 30463 larger than `{LONG_MAX}`.
- 30464 The *pathconf()* function may fail if:
- 30465 [EACCES] Search permission is denied for a component of the path prefix.
- 30466 [EINVAL] The implementation does not support an association of the variable *name* with  
 30467 the specified file.
- 30468 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path* +  
 30469 argument. +
- 30470 [ELOOP] More than `{SYMLOOP_MAX}` symbolic links were encountered during  
 30471 resolution of the *path* argument.
- 30472 [ENAMETOOLONG]  
 30473 The length of a component of a pathname is longer than `{NAME_MAX}`.
- 30474 [ENAMETOOLONG]  
 30475 The length of a pathname exceeds `{PATH_MAX}`, or pathname resolution of a  
 30476 symbolic link produced an intermediate result with a length that exceeds  
 30477 `{PATH_MAX}`.
- 30478 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.
- 30479 [ENOTDIR] A component of the path prefix names an existing file that is neither a  
 30480 directory nor a symbolic link to a directory, or the *path* argument contains at  
 30481 least one non-`<slash>` character and ends with one or more trailing `<slash>`  
 30482 characters and the last pathname component names an existing file that is  
 30483 neither a directory nor a symbolic link to a directory.
- 30484 The *fpathconf()* function shall fail if:
- 30485 [EINVAL] The value of *name* is not valid.
- 30486 [EOVERFLOW] The value of *name* is `_PC_TIMESTAMP_RESOLUTION` and the resolution is  
 30487 larger than `{LONG_MAX}`.
- 30488 The *fpathconf()* function may fail if:
- 30489 [EBADF] The *fildev* argument is not a valid file descriptor.
- 30490 [EINVAL] The implementation does not support an association of the variable *name* with  
 30491 the specified file.

**EXAMPLES**

30492 None.

**APPLICATION USAGE**

30495 Application developers should check whether an option, such as `_POSIX_ADVISORY_INFO`, is  
 30496 supported prior to obtaining and using values for related variables such as  
 30497 `{POSIX_ALLOC_SIZE_MIN}`.

**RATIONALE**

30499 The *pathconf()* function was proposed immediately after the *sysconf()* function when it was  
 30500 realized that some configurable values may differ across file system, directory, or device  
 30501 boundaries.

30502 For example, `{NAME_MAX}` frequently changes between System V and BSD-based file systems;  
 30503 System V uses a maximum of 14, BSD 255. On an implementation that provides both types of file

systems, an application would be forced to limit all pathname components to 14 bytes, as this would be the value specified in `<limits.h>` on such a system.

Therefore, various useful values can be queried on any pathname or file descriptor, assuming that appropriate privileges are in place.

The value returned for the variable `{PATH_MAX}` indicates the longest relative pathname that could be given if the specified directory is the current working directory of the process. A process may not always be able to generate a name that long and use it if a subdirectory in the pathname crosses into a more restrictive file system. Note that implementations are allowed to accept pathnames longer than `{PATH_MAX}` bytes long, but are not allowed to return pathnames longer than this unless the user specifies a larger buffer using a function that provides a buffer size argument.

The value returned for the variable `_POSIX_CHOWN_RESTRICTED` also applies to directories that do not have file systems mounted on them. The value may change when crossing a mount point, so applications that need to know should check for each directory. (An even easier check is to try the `chown()` function and look for an error in case it happens.)

Unlike the values returned by `sysconf()`, the pathname-oriented variables are potentially more volatile and are not guaranteed to remain constant throughout the lifetime of the process. For example, in between two calls to `pathconf()`, the file system in question may have been unmounted and remounted with different characteristics.

Also note that most of the errors are optional. If one of the variables always has the same value on an implementation, the implementation need not look at *path* or *fildev* to return that value and is, therefore, not required to detect any of the errors except the meaning of `[EINVAL]` that indicates that the value of *name* is not valid for that variable, and the `[EOVERFLOW]` error that indicates the value to be returned is larger than `{LONG_MAX}`.

If the value of any of the limits is unspecified (logically infinite), they will not be defined in `<limits.h>` and the `pathconf()` and `fpathconf()` functions return `-1` without changing `errno`. This can be distinguished from the case of giving an unrecognized *name* argument because `errno` is set to `[EINVAL]` in this case.

Since `-1` is a valid return value for the `pathconf()` and `fpathconf()` functions, applications should set `errno` to zero before calling them and check `errno` only if the return value is `-1`.

For the case of `{SYMLINK_MAX}`, since both `pathconf()` and `open()` follow symbolic links, there is no way that *path* or *fildev* could refer to a symbolic link.

It was the intention of IEEE Std 1003.1d-1999 that the following variables:

```
{POSIX_ALLOC_SIZE_MIN}
{POSIX_REC_INCR_XFER_SIZE}
{POSIX_REC_MAX_XFER_SIZE}
{POSIX_REC_MIN_XFER_SIZE}
{POSIX_REC_XFER_ALIGN}
```

only applied to regular files, but Note 10 also permits implementation of the advisory semantics on other file types unique to an implementation (for example, a character special device).

The `[EOVERFLOW]` error for `_PC_TIMESTAMP_RESOLUTION` cannot occur on POSIX-compliant file systems because POSIX requires a timestamp resolution no larger than one second. Even on 32-bit systems, this can be represented without overflow.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*chown()*, *confstr()*, *sysconf()*

XBD *<limits.h>*, *<unistd.h>*

XCU *getconf*

**CHANGE HISTORY**

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

**Issue 5**

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension.

Large File Summit extensions are added.

**Issue 6**

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to include {FILESIZEBITS}.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The \_PC\_SYMLINK\_MAX entry is added to the table in the DESCRIPTION.

The following *pathconf()* variables and their associated names are added for alignment with IEEE Std 1003.1d-1999:

```
{POSIX_ALLOC_SIZE_MIN}
{POSIX_REC_INCR_XFER_SIZE}
{POSIX_REC_MAX_XFER_SIZE}
{POSIX_REC_MIN_XFER_SIZE}
{POSIX_REC_XFER_ALIGN}
```

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/18 is applied, changing the fourth paragraph of the DESCRIPTION and removing shading and margin markers from the table. This change is needed since implementations are required to support all of these symbols.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/34 is applied, adding the table entry for POSIX2\_SYMLINKS in the DESCRIPTION.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/35 is applied, updating the DESCRIPTION and RATIONALE sections to clarify behavior for the following variables:

```
{POSIX_ALLOC_SIZE_MIN}
{POSIX_REC_INCR_XFER_SIZE}
{POSIX_REC_MAX_XFER_SIZE}
{POSIX_REC_MIN_XFER_SIZE}
{POSIX_REC_XFER_ALIGN}
```

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/36 is applied, updating the RETURN VALUE and APPLICATION USAGE sections to state that the results are unspecified if a variable is dependent on an unsupported option, and advising application developers to check for

30588 supported options prior to obtaining and using such values.

30589 **Issue 7**

30590 Austin Group Interpretations 1003.1-2001 #143 and #160 are applied.

30591 Changes are made related to support for finegrained timestamps.

30592 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a  
30593 pathname exists but is not a directory or a symbolic link to a directory.

30594 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0160 [256,428], XSH/TC1-2008/0161  
30595 [256,428], and XSH/TC1-2008/0162 [324] are applied.

30596 **NAME**

30597 fpclassify — classify real floating type

30598 **SYNOPSIS**

30599 #include &lt;math.h&gt;

30600 int fpclassify(real-floating x);

30601 **DESCRIPTION**

30602 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
30603 conflict between the requirements described here and the ISO C standard is unintentional. This  
30604 volume of POSIX.1-2008 defers to the ISO C standard.

30605 The *fpclassify()* macro shall classify its argument value as NaN, infinite, normal, subnormal,  
30606 zero, or into another implementation-defined category. First, an argument represented in a  
30607 format wider than its semantic type is converted to its semantic type. Then classification is based  
30608 on the type of the argument.

30609 **RETURN VALUE**

30610 The *fpclassify()* macro shall return the value of the number classification macro appropriate to  
30611 the value of its argument.

30612 **ERRORS**

30613 No errors are defined.

30614 **EXAMPLES**

30615 None.

30616 **APPLICATION USAGE**

30617 None.

30618 **RATIONALE**

30619 None.

30620 **FUTURE DIRECTIONS**

30621 None.

30622 **SEE ALSO**30623 *isfinite()*, *isinf()*, *isnan()*, *isnormal()*, *signbit()*

30624 XBD &lt;math.h&gt;

30625 **CHANGE HISTORY**

30626 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

**NAME**

dprintf, fprintf, printf, snprintf, sprintf — print formatted output

**SYNOPSIS**

```
#include <stdio.h>
```

CX

```
int dprintf(int filides, const char *restrict format, ...);
int fprintf(FILE *restrict stream, const char *restrict format, ...);
int printf(const char *restrict format, ...);
int snprintf(char *restrict s, size_t n,
             const char *restrict format, ...);
int sprintf(char *restrict s, const char *restrict format, ...);
```

**DESCRIPTION**

CX

Excluding *dprintf()*: The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *fprintf()* function shall place output on the named output *stream*. The *printf()* function shall place output on the standard output stream *stdout*. The *sprintf()* function shall place output followed by the null byte, '\0', in consecutive bytes starting at *s*; it is the user's responsibility to ensure that enough space is available.

CX

The *dprintf()* function shall be equivalent to the *fprintf()* function, except that *dprintf()* shall write output to the file associated with the file descriptor specified by the *filides* argument rather than place output on a stream.

The *snprintf()* function shall be equivalent to *sprintf()*, with the addition of the *n* argument which states the size of the buffer referred to by *s*. If *n* is zero, nothing shall be written and *s* may be a null pointer. Otherwise, output bytes beyond the *n*-1st shall be discarded instead of being written to the array, and a null byte is written at the end of the bytes actually written into the array.

If copying takes place between objects that overlap as a result of a call to *sprintf()* or *snprintf()*, the results are undefined.

Each of these functions converts, formats, and prints its arguments under control of the *format*. The *format* is a character string, beginning and ending in its initial shift state, if any. The *format* is composed of zero or more directives: *ordinary characters*, which are simply copied to the output stream, and *conversion specifications*, each of which shall result in the fetching of zero or more arguments. The results are undefined if there are insufficient arguments for the *format*. If the *format* is exhausted while arguments remain, the excess arguments shall be evaluated but are otherwise ignored.

CX

Conversions can be applied to the *n*th argument after the *format* in the argument list, rather than to the next unused argument. In this case, the conversion specifier character % (see below) is replaced by the sequence "%*n*\$", where *n* is a decimal integer in the range [1,{NL\_ARGMAX}], giving the position of the argument in the argument list. This feature provides for the definition of format strings that select arguments in an order appropriate to specific languages (see the EXAMPLES section).

The *format* can contain either numbered argument conversion specifications (that is, "%*n*\$" and "%*m*\$"), or unnumbered argument conversion specifications (that is, % and \*), but not both. The only exception to this is that %% can be mixed with the "%*n*\$" form. The results of mixing numbered and unnumbered argument specifications in a *format* string are undefined. When numbered argument specifications are used, specifying the *N*th argument requires that all the leading arguments, from the first to the (*N*-1)th, are specified in the format string.

- 30674 In format strings containing the "%n\$" form of conversion specification, numbered arguments  
 30675 in the argument list can be referenced from the format string as many times as required.
- 30676 In format strings containing the % form of conversion specification, each conversion specification  
 30677 uses the first unused argument in the argument list.
- 30678 CX All forms of the *fprintf()* functions allow for the insertion of a language-dependent radix  
 30679 character in the output string. The radix character is defined in the current locale (category  
 30680 LC\_NUMERIC). In the POSIX locale, or in a locale where the radix character is not defined, the  
 30681 radix character shall default to a <period> ('.').
- 30682 CX Each conversion specification is introduced by the '%' character or by the character sequence  
 30683 "%n\$", after which the following appear in sequence:
- 30684 • Zero or more *flags* (in any order), which modify the meaning of the conversion  
 30685 specification.
  - 30686 • An optional minimum *field width*. If the converted value has fewer bytes than the field  
 30687 width, it shall be padded with <space> characters by default on the left; it shall be padded  
 30688 on the right if the left-adjustment flag ('-'), described below, is given to the field width.  
 30689 The field width takes the form of an <asterisk> ('\*'), described below, or a decimal  
 30690 integer.
  - 30691 • An optional *precision* that gives the minimum number of digits to appear for the d, i, o, u,  
 30692 x, and X conversion specifiers; the number of digits to appear after the radix character for  
 30693 the a, A, e, E, f, and F conversion specifiers; the maximum number of significant digits for  
 30694 the g and G conversion specifiers; or the maximum number of bytes to be printed from a  
 30695 XSI string in the s and S conversion specifiers. The precision takes the form of a <period>  
 30696 ('.') followed either by an <asterisk> ('\*'), described below, or an optional decimal digit  
 30697 string, where a null digit string is treated as zero. If a precision appears with any other  
 30698 conversion specifier, the behavior is undefined.
  - 30699 • An optional length modifier that specifies the size of the argument.
  - 30700 • A *conversion specifier* character that indicates the type of conversion to be applied.
- 30701 A field width, or precision, or both, may be indicated by an <asterisk> ('\*'). In this case an  
 30702 argument of type **int** supplies the field width or precision. Applications shall ensure that  
 30703 arguments specifying field width, or precision, or both appear in that order before the argument,  
 30704 if any, to be converted. A negative field width is taken as a '-' flag followed by a positive field  
 30705 CX width. A negative precision is taken as if the precision were omitted. In *format* strings  
 30706 containing the "%n\$" form of a conversion specification, a field width or precision may be  
 30707 indicated by the sequence "%m\$", where *m* is a decimal integer in the range [1,{NL\_ARGMAX}]  
 30708 giving the position in the argument list (after the *format* argument) of an integer argument  
 30709 containing the field width or precision, for example:
- ```
30710 printf("%1$d:%2$.*3$d:%4$.*3$d\n", hour, min, precision, sec);
```
- 30711 The flag characters and their meanings are:
- 30712 CX ' (The <apostrophe>.) The integer portion of the result of a decimal conversion (%i, %d,  
 30713 %u, %f, %F, %g, or %G) shall be formatted with thousands' grouping characters. For  
 30714 other conversions the behavior is undefined. The non-monetary grouping character is  
 30715 used.
  - 30716 – The result of the conversion shall be left-justified within the field. The conversion is  
 30717 right-justified if this flag is not specified.



30718 + The result of a signed conversion shall always begin with a sign ('+' or '-'). The  
 30719 conversion shall begin with a sign only when a negative value is converted if this flag is  
 30720 not specified.

30721 <space> If the first character of a signed conversion is not a sign or if a signed conversion results  
 30722 in no characters, a <space> shall be prefixed to the result. This means that if the  
 30723 <space> and '+' flags both appear, the <space> flag shall be ignored.

30724 # Specifies that the value is to be converted to an alternative form. For o conversion, it  
 30725 shall increase the precision, if and only if necessary, to force the first digit of the result  
 30726 to be a zero (if the value and precision are both 0, a single 0 is printed). For x or X  
 30727 conversion specifiers, a non-zero result shall have 0x (or 0X) prefixed to it. For a, A, e,  
 30728 E, f, F, g, and G conversion specifiers, the result shall always contain a radix character,  
 30729 even if no digits follow the radix character. Without this flag, a radix character appears  
 30730 in the result of these conversions only if a digit follows it. For g and G conversion  
 30731 specifiers, trailing zeros shall *not* be removed from the result as they normally are. For  
 30732 other conversion specifiers, the behavior is undefined.

30733 0 For d, i, o, u, x, X, a, A, e, E, f, F, g, and G conversion specifiers, leading zeros  
 30734 (following any indication of sign or base) are used to pad to the field width rather than  
 30735 performing space padding, except when converting an infinity or NaN. If the '0' and  
 30736 '-' flags both appear, the '0' flag is ignored. For d, i, o, u, x, and X conversion  
 30737 CX specifiers, if a precision is specified, the '0' flag shall be ignored. If the '0' and  
 30738 <apostrophe> flags both appear, the grouping characters are inserted before zero  
 30739 padding. For other conversions, the behavior is undefined.

30740 The length modifiers and their meanings are:

30741 hh Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **signed char**  
 30742 or **unsigned char** argument (the argument will have been promoted according to the  
 30743 integer promotions, but its value shall be converted to **signed char** or **unsigned char**  
 30744 before printing); or that a following n conversion specifier applies to a pointer to a  
 30745 **signed char** argument.

30746 h Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **short** or  
 30747 **unsigned short** argument (the argument will have been promoted according to the  
 30748 integer promotions, but its value shall be converted to **short** or **unsigned short** before  
 30749 printing); or that a following n conversion specifier applies to a pointer to a **short**  
 30750 argument.

30751 l (ell) Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **long** or  
 30752 **unsigned long** argument; that a following n conversion specifier applies to a pointer to  
 30753 a **long** argument; that a following c conversion specifier applies to a **wint\_t** argument;  
 30754 that a following s conversion specifier applies to a pointer to a **wchar\_t** argument; or  
 30755 has no effect on a following a, A, e, E, f, F, g, or G conversion specifier.

30756 ll (ell-ell) Specifies that a following d, i, o, u, x, or X conversion specifier applies to a **long long**  
 30757 or **unsigned long long** argument; or that a following n conversion specifier applies to a  
 30758 pointer to a **long long** argument.

30760 j Specifies that a following d, i, o, u, x, or X conversion specifier applies to an **intmax\_t**  
 30761 or **uintmax\_t** argument; or that a following n conversion specifier applies to a pointer  
 30762 to an **intmax\_t** argument.



|       |                                                                                                        |                                                                                                       |
|-------|--------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|
| 30763 | z                                                                                                      | Specifies that a following d, i, o, u, x, or X conversion specifier applies to a <b>size_t</b> or the |
| 30764 |                                                                                                        | corresponding signed integer type argument; or that a following n conversion specifier                |
| 30765 |                                                                                                        | applies to a pointer to a signed integer type corresponding to a <b>size_t</b> argument.              |
| 30766 | t                                                                                                      | Specifies that a following d, i, o, u, x, or X conversion specifier applies to a <b>ptrdiff_t</b> or  |
| 30767 |                                                                                                        | the corresponding <b>unsigned</b> type argument; or that a following n conversion specifier           |
| 30768 |                                                                                                        | applies to a pointer to a <b>ptrdiff_t</b> argument.                                                  |
| 30769 | L                                                                                                      | Specifies that a following a, A, e, E, f, F, g, or G conversion specifier applies to a <b>long</b>    |
| 30770 |                                                                                                        | <b>double</b> argument.                                                                               |
| 30771 | If a length modifier appears with any conversion specifier other than as specified above, the          |                                                                                                       |
| 30772 | behavior is undefined.                                                                                 |                                                                                                       |
| 30773 | The conversion specifiers and their meanings are:                                                      |                                                                                                       |
| 30774 | d, i                                                                                                   | The <b>int</b> argument shall be converted to a signed decimal in the style "[−]dddd". The            |
| 30775 |                                                                                                        | precision specifies the minimum number of digits to appear; if the value being                        |
| 30776 |                                                                                                        | converted can be represented in fewer digits, it shall be expanded with leading zeros.                |
| 30777 |                                                                                                        | The default precision is 1. The result of converting zero with an explicit precision of               |
| 30778 |                                                                                                        | zero shall be no characters.                                                                          |
| 30779 | o                                                                                                      | The <b>unsigned</b> argument shall be converted to unsigned octal format in the style                 |
| 30780 |                                                                                                        | "dddd". The precision specifies the minimum number of digits to appear; if the value                  |
| 30781 |                                                                                                        | being converted can be represented in fewer digits, it shall be expanded with leading                 |
| 30782 |                                                                                                        | zeros. The default precision is 1. The result of converting zero with an explicit precision           |
| 30783 |                                                                                                        | of zero shall be no characters.                                                                       |
| 30784 | u                                                                                                      | The <b>unsigned</b> argument shall be converted to unsigned decimal format in the style               |
| 30785 |                                                                                                        | "dddd". The precision specifies the minimum number of digits to appear; if the value                  |
| 30786 |                                                                                                        | being converted can be represented in fewer digits, it shall be expanded with leading                 |
| 30787 |                                                                                                        | zeros. The default precision is 1. The result of converting zero with an explicit precision           |
| 30788 |                                                                                                        | of zero shall be no characters.                                                                       |
| 30789 | x                                                                                                      | The <b>unsigned</b> argument shall be converted to unsigned hexadecimal format in the style           |
| 30790 |                                                                                                        | "dddd"; the letters "abcdef" are used. The precision specifies the minimum number                     |
| 30791 |                                                                                                        | of digits to appear; if the value being converted can be represented in fewer digits, it              |
| 30792 |                                                                                                        | shall be expanded with leading zeros. The default precision is 1. The result of                       |
| 30793 |                                                                                                        | converting zero with an explicit precision of zero shall be no characters.                            |
| 30794 | X                                                                                                      | Equivalent to the x conversion specifier, except that letters "ABCDEF" are used instead               |
| 30795 |                                                                                                        | of "abcdef".                                                                                          |
| 30796 | f, F                                                                                                   | The <b>double</b> argument shall be converted to decimal notation in the style                        |
| 30797 |                                                                                                        | "[−]ddd.ddd", where the number of digits after the radix character is equal to the                    |
| 30798 |                                                                                                        | precision specification. If the precision is missing, it shall be taken as 6; if the precision        |
| 30799 |                                                                                                        | is explicitly zero and no '#' flag is present, no radix character shall appear. If a radix            |
| 30800 |                                                                                                        | character appears, at least one digit appears before it. The low-order digit shall be                 |
| 30801 |                                                                                                        | rounded in an implementation-defined manner.                                                          |
| 30802 | A <b>double</b> argument representing an infinity shall be converted in one of the styles              |                                                                                                       |
| 30803 | "[−]inf" or "[−]infinity"; which style is implementation-defined. A <b>double</b>                      |                                                                                                       |
| 30804 | argument representing a NaN shall be converted in one of the styles "[−]nan( <i>n-char-sequence</i> )" |                                                                                                       |
| 30805 | or "[−]nan"; which style, and the meaning of any <i>n-char-sequence</i> ,                              |                                                                                                       |
| 30806 | is implementation-defined. The F conversion specifier produces "INF", "INFINITY",                      |                                                                                                       |
| 30807 | or "NAN" instead of "inf", "infinity", or "nan", respectively.                                         |                                                                                                       |

|       |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------|------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 30808 | e, E | The <b>double</b> argument shall be converted in the style " <code>[-]d.ddde±dd</code> ", where there is one digit before the radix character (which is non-zero if the argument is non-zero) and the number of digits after it is equal to the precision; if the precision is missing, it shall be taken as 6; if the precision is zero and no <code>'#'</code> flag is present, no radix character shall appear. The low-order digit shall be rounded in an implementation-defined manner. The <code>E</code> conversion specifier shall produce a number with <code>'E'</code> instead of <code>'e'</code> introducing the exponent. The exponent shall always contain at least two digits. If the value is zero, the exponent shall be zero.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 30809 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30810 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30811 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30812 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30813 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30814 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30815 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30816 |      | A <b>double</b> argument representing an infinity or NaN shall be converted in the style of an <code>f</code> or <code>F</code> conversion specifier.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 30817 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30818 | g, G | The <b>double</b> argument representing a floating-point number shall be converted in the style <code>f</code> or <code>e</code> (or in the style <code>F</code> or <code>E</code> in the case of a <code>G</code> conversion specifier), depending on the value converted and the precision. Let <code>P</code> equal the precision if non-zero, 6 if the precision is omitted, or 1 if the precision is zero. Then, if a conversion with style <code>E</code> would have an exponent of <code>X</code> :                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 30819 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30820 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30821 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30822 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30823 |      | — If $P > X \geq -4$ , the conversion shall be with style <code>f</code> (or <code>F</code> ) and precision $P - (X + 1)$ .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 30824 |      | — Otherwise, the conversion shall be with style <code>e</code> (or <code>E</code> ) and precision $P - 1$ .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 30825 |      | Finally, unless the <code>'#'</code> flag is used, any trailing zeros shall be removed from the fractional portion of the result and the decimal-point character shall be removed if there is no fractional portion remaining.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30826 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30827 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30828 |      | A <b>double</b> argument representing an infinity or NaN shall be converted in the style of an <code>f</code> or <code>F</code> conversion specifier.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 30829 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30830 | a, A | A <b>double</b> argument representing a floating-point number shall be converted in the style " <code>[-]0xh.hhhhp±d</code> ", where there is one hexadecimal digit (which shall be non-zero if the argument is a normalized floating-point number and is otherwise unspecified) before the decimal-point character and the number of hexadecimal digits after it is equal to the precision; if the precision is missing and <code>FLT_RADIX</code> is a power of 2, then the precision shall be sufficient for an exact representation of the value; if the precision is missing and <code>FLT_RADIX</code> is not a power of 2, then the precision shall be sufficient to distinguish values of type <b>double</b> , except that trailing zeros may be omitted; if the precision is zero and the <code>'#'</code> flag is not specified, no decimal-point character shall appear. The letters <code>"abcdef"</code> shall be used for a conversion and the letters <code>"ABCDEF"</code> for <code>A</code> conversion. The <code>A</code> conversion specifier produces a number with <code>'X'</code> and <code>'P'</code> instead of <code>'x'</code> and <code>'p'</code> . The exponent shall always contain at least one digit, and only as many more digits as necessary to represent the decimal exponent of 2. If the value is zero, the exponent shall be zero. |
| 30831 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30832 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30833 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30834 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30835 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30836 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30837 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30838 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30839 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30840 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30841 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30842 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30843 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30844 |      | A <b>double</b> argument representing an infinity or NaN shall be converted in the style of an <code>f</code> or <code>F</code> conversion specifier.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 30845 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30846 | c    | The <b>int</b> argument shall be converted to an <b>unsigned char</b> , and the resulting byte shall be written.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 30847 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30848 |      | If an <code>l</code> (ell) qualifier is present, the <b>wint_t</b> argument shall be converted as if by an <code>ls</code> conversion specification with no precision and an argument that points to a two-element array of type <b>wchar_t</b> , the first element of which contains the <b>wint_t</b> argument to the <code>ls</code> conversion specification and the second element contains a null wide character.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 30849 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30850 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30851 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 30852 |      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |

|       |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 30853 | s   | The argument shall be a pointer to an array of <b>char</b> . Bytes from the array shall be written up to (but not including) any terminating null byte. If the precision is specified, no more than that many bytes shall be written. If the precision is not specified or is greater than the size of the array, the application shall ensure that the array contains a null byte.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 30854 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30855 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30856 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30857 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30858 |     | If an <b>l</b> (ell) qualifier is present, the argument shall be a pointer to an array of type <b>wchar_t</b> . Wide characters from the array shall be converted to characters (each as if by a call to the <i>wcrtomb()</i> function, with the conversion state described by an <b>mbstate_t</b> object initialized to zero before the first wide character is converted) up to and including a terminating null wide character. The resulting characters shall be written up to (but not including) the terminating null character (byte). If no precision is specified, the application shall ensure that the array contains a null wide character. If a precision is specified, no more than that many characters (bytes) shall be written (including shift sequences, if any), and the array shall contain a null wide character if, to equal the character sequence length given by the precision, the function would need to access a wide character one past the end of the array. In no case shall a partial character be written. |
| 30859 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30860 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30861 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30862 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30863 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30864 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30865 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30866 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30867 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30868 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30869 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30870 | p   | The argument shall be a pointer to <b>void</b> . The value of the pointer is converted to a sequence of printable characters, in an implementation-defined manner.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 30871 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30872 | n   | The argument shall be a pointer to an integer into which is written the number of bytes written to the output so far by this call to one of the <i>fprintf()</i> functions. No argument is converted.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 30873 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30874 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30875 | XSI | <b>C</b> Equivalent to <b>lc</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 30876 | XSI | <b>S</b> Equivalent to <b>ls</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 30877 | %   | Print a ' % ' character; no argument is converted. The complete conversion specification shall be %%.<br>If a conversion specification does not match one of the above forms, the behavior is undefined. If any argument is not the correct type for the corresponding conversion specification, the behavior is undefined.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 30878 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30879 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30880 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30881 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30882 |     | In no case shall a nonexistent or small field width cause truncation of a field; if the result of a conversion is wider than the field width, the field shall be expanded to contain the conversion result. Characters generated by <i>fprintf()</i> and <i>printf()</i> are printed as if <i>fputc()</i> had been called.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 30883 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30884 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30885 |     | For the <b>a</b> and <b>A</b> conversion specifiers, if <b>FLT_RADIX</b> is a power of 2, the value shall be correctly rounded to a hexadecimal floating number with the given precision.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 30886 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30887 |     | For <b>a</b> and <b>A</b> conversions, if <b>FLT_RADIX</b> is not a power of 2 and the result is not exactly representable in the given precision, the result should be one of the two adjacent numbers in hexadecimal floating style with the given precision, with the extra stipulation that the error should have a correct sign for the current rounding direction.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 30888 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30889 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30890 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30891 |     | For the <b>e</b> , <b>E</b> , <b>f</b> , <b>F</b> , <b>g</b> , and <b>G</b> conversion specifiers, if the number of significant decimal digits is at most <b>DECIMAL_DIG</b> , then the result should be correctly rounded. If the number of significant decimal digits is more than <b>DECIMAL_DIG</b> but the source value is exactly representable with <b>DECIMAL_DIG</b> digits, then the result should be an exact representation with trailing zeros. Otherwise, the source value is bounded by two adjacent decimal strings $L < U$ , both having <b>DECIMAL_DIG</b> significant digits; the value of the resultant decimal string $D$ should satisfy $L \leq D \leq U$ , with the extra stipulation that the error should have a correct sign for the current rounding direction.                                                                                                                                                                                                                                                   |
| 30892 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30893 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30894 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30895 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30896 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30897 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 30898 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

30899 CX The last data modification and last file status change timestamps of the file shall be marked for  
30900 update:

- 30901 1. Between the call to a successful execution of *fprintf()* or *printf()* and the next successful  
30902 completion of a call to *fflush()* or *fclose()* on the same stream or a call to *exit()* or *abort()*
- 30903 2. Upon successful completion of a call to *dprintf()*

#### 30904 RETURN VALUE

30905 CX Upon successful completion, the *dprintf()*, *fprintf()*, and *printf()* functions shall return the  
30906 number of bytes transmitted.

30907 Upon successful completion, the *sprintf()* function shall return the number of bytes written to *s*,  
30908 excluding the terminating null byte.

30909 Upon successful completion, the *snprintf()* function shall return the number of bytes that would  
30910 be written to *s* had *n* been sufficiently large excluding the terminating null byte.

30911 CX If an output error was encountered, these functions shall return a negative value and set *errno* to  
30912 indicate the error.

30913 If the value of *n* is zero on a call to *snprintf()*, nothing shall be written, the number of bytes that  
30914 would have been written had *n* been sufficiently large excluding the terminating null shall be  
30915 returned, and *s* may be a null pointer.

#### 30916 ERRORS

30917 CX For the conditions under which *dprintf()*, *fprintf()*, and *printf()* fail and may fail, refer to *fputc()*  
30918 or *fputwc()*.

30919 In addition, all forms of *fprintf()* shall fail if:

30920 CX [EILSEQ] A wide-character code that does not correspond to a valid character has been  
30921 detected.

30922 CX [EOVERFLOW] The value to be returned is greater than {INT\_MAX}.

30923 The *dprintf()* function may fail if:

30924 [EBADF] The *filides* argument is not a valid file descriptor.

30925 CX The *dprintf()*, *fprintf()*, and *printf()* functions may fail if:

30926 CX [ENOMEM] Insufficient storage space is available.

30927 The *snprintf()* function shall fail if:

30928 CX [EOVERFLOW] The value of *n* is greater than {INT\_MAX}.

#### 30929 EXAMPLES

##### 30930 Printing Language-Independent Date and Time

30931 The following statement can be used to print date and time using a language-independent  
30932 format:

30933 `printf(format, weekday, month, day, hour, min);`

30934 For American usage, *format* could be a pointer to the following string:

30935 `"%s, %s %d, %d:%.2d\n"`

30936 This example would produce the following message:

30937 Sunday, July 3, 10:02  
 30938 For German usage, *format* could be a pointer to the following string:  
 30939 "%1\$s, %3\$d. %2\$s, %4\$d:%5\$.2d\n"  
 30940 This definition of *format* would produce the following message:  
 30941 Sonntag, 3. Juli, 10:02

### 30942 Printing File Information

30943 The following example prints information about the type, permissions, and number of links of a  
 30944 specific file in a directory.

30945 The first two calls to *printf()* use data decoded from a previous *stat()* call. The user-defined  
 30946 *strperm()* function shall return a string similar to the one at the beginning of the output for the  
 30947 following command:

30948 `ls -l`

30949 The next call to *printf()* outputs the owner's name if it is found using *getpwuid()*; the *getpwuid()*  
 30950 function shall return a **passwd** structure from which the name of the user is extracted. If the user  
 30951 name is not found, the program instead prints out the numeric value of the user ID.

30952 The next call prints out the group name if it is found using *getgrgid()*; *getgrgid()* is very similar  
 30953 to *getpwuid()* except that it shall return group information based on the group number. Once  
 30954 again, if the group is not found, the program prints the numeric value of the group for the entry.

30955 The final call to *printf()* prints the size of the file.

```
30956 #include <stdio.h>
30957 #include <sys/types.h>
30958 #include <pwd.h>
30959 #include <grp.h>
30960 char *strperm (mode_t);
30961 ...
30962 struct stat statbuf;
30963 struct passwd *pwd;
30964 struct group *grp;
30965 ...
30966 printf("%10.10s", strperm (statbuf.st_mode));
30967 printf("%4d", statbuf.st_nlink);
30968 if ((pwd = getpwuid(statbuf.st_uid)) != NULL)
30969     printf(" %-8.8s", pwd->pw_name);
30970 else
30971     printf(" %-8ld", (long) statbuf.st_uid);
30972 if ((grp = getgrgid(statbuf.st_gid)) != NULL)
30973     printf(" %-8.8s", grp->gr_name);
30974 else
30975     printf(" %-8ld", (long) statbuf.st_gid);
30976 printf("%9jd", (intmax_t) statbuf.st_size);
30977 ...
```

**Printing a Localized Date String**

The following example gets a localized date string. The `nl_langinfo()` function shall return the localized date string, which specifies the order and layout of the date. The `strftime()` function takes this information and, using the `tm` structure for values, places the date and time information into `datestring`. The `printf()` function then outputs `datestring` and the name of the entry.

```
#include <stdio.h>
#include <time.h>
#include <langinfo.h>
...
struct dirent *dp;
struct tm *tm;
char datestring[256];
...
strftime(datestring, sizeof(datestring), nl_langinfo (D_T_FMT), tm);
printf(" %s %s\n", datestring, dp->d_name);
...
```

**Printing Error Information**

The following example uses `fprintf()` to write error information to standard error.

In the first group of calls, the program tries to open the password lock file named **LOCKFILE**. If the file already exists, this is an error, as indicated by the `O_EXCL` flag on the `open()` function. If the call fails, the program assumes that someone else is updating the password file, and the program exits.

The next group of calls saves a new password file as the current password file by creating a link between **LOCKFILE** and the new password file **PASSWDFILE**.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <errno.h>

#define LOCKFILE "/etc/ptmp"
#define PASSWDFILE "/etc/passwd"
...
int pfd;
...
if ((pfd = open(LOCKFILE, O_WRONLY | O_CREAT | O_EXCL,
    S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)) == -1)
{
    fprintf(stderr, "Cannot open /etc/ptmp. Try again later.\n");
    exit(1);
}
...
if (link(LOCKFILE, PASSWDFILE) == -1) {
```

```

31024         fprintf(stderr, "Link error: %s\n", strerror(errno));
31025         exit(1);
31026     }
31027     ...

```

### Printing Usage Information

The following example checks to make sure the program has the necessary arguments, and uses *fprintf()* to print usage information if the expected number of arguments is not present.

```

31031 #include <stdio.h>
31032 #include <stdlib.h>
31033 ...
31034 char *Options = "hdbtl";
31035 ...
31036 if (argc < 2) {
31037     fprintf(stderr, "Usage: %s -%s <file\n", argv[0], Options); exit(1);
31038 }
31039 ...

```

### Formatting a Decimal String

The following example prints a key and data pair on *stdout*. Note use of the <asterisk> ('\*') in the format string; this ensures the correct number of decimal places for the element based on the number of elements requested.

```

31044 #include <stdio.h>
31045 ...
31046 long i;
31047 char *keyststr;
31048 int elementlen, len;
31049 ...
31050 while (len < elementlen) {
31051     ...
31052     printf("%s Element%0*ld\n", keyststr, elementlen, i);
31053     ...
31054 }

```

### Creating a Pathname

The following example creates a pathname using information from a previous *getpwnam()* function that returned the password database entry of the user.

```

31058 #include <stdint.h>
31059 #include <stdio.h>
31060 #include <stdlib.h>
31061 #include <string.h>
31062 #include <sys/types.h>
31063 #include <unistd.h>
31064 ...
31065 char *pathname;
31066 struct passwd *pw;
31067 size_t len;

```



```

31068     ...
31069     // digits required for pid_t is number of bits times
31070     // log2(10) = approx 10/33
31071     len = strlen(pw->pw_dir) + 1 + 1+(sizeof(pid_t)*80+32)/33 +
31072         sizeof ".out";
31073     pathname = malloc(len);
31074     if (pathname != NULL)
31075     {
31076         snprintf(pathname, len, "%s/%jd.out", pw->pw_dir,
31077             (intmax_t) getpid());
31078         ...
31079     }

```

### Reporting an Event

The following example loops until an event has timed out. The *pause()* function waits forever unless it receives a signal. The *fprintf()* statement should never occur due to the possible return values of *pause()*.

```

31084     #include <stdio.h>
31085     #include <unistd.h>
31086     #include <string.h>
31087     #include <errno.h>
31088     ...
31089     while (!event_complete) {
31090     ...
31091         if (pause() != -1 || errno != EINTR)
31092             fprintf(stderr, "pause: unknown error: %s\n", strerror(errno));
31093     }
31094     ...

```

### Printing Monetary Information

The following example uses *strfmon()* to convert a number and store it as a formatted monetary string named *convbuf*. If the first number is printed, the program prints the format and the description; otherwise, it just prints the number.

```

31099     #include <monetary.h>
31100     #include <stdio.h>
31101     ...
31102     struct tblfmt {
31103         char *format;
31104         char *description;
31105     };
31106     struct tblfmt table[] = {
31107         { "%n", "default formatting" },
31108         { "%11n", "right align within an 11 character field" },
31109         { "%#5n", "aligned columns for values up to 99999" },
31110         { "%=*#5n", "specify a fill character" },
31111         { "%=0#5n", "fill characters do not use grouping" },
31112         { "%^#5n", "disable the grouping separator" },
31113         { "%^#5.0n", "round off to whole units" },

```



```

31114         { "%^#5.4n", "increase the precision" },
31115         { "%(#5n", "use an alternative pos/neg style" },
31116         { "%!(#5n", "disable the currency symbol" },
31117     };
31118     ...
31119     float input[3];
31120     int i, j;
31121     char convbuf[100];
31122     ...
31123     strfmon(convbuf, sizeof(convbuf), table[i].format, input[j]);
31124
31125     if (j == 0) {
31126         printf("%s%s%s\n", table[i].format,
31127             convbuf, table[i].description);
31128     }
31129     else {
31130         printf("%s\n", convbuf);
31131     }
31132     ...

```

### Printing Wide Characters

The following example prints a series of wide characters. Suppose that "L'@'" expands to three bytes:

```

31135     wchar_t wz [3] = L"@@";           // Zero-terminated
31136     wchar_t wn [3] = L"@@";           // Unterminated
31137
31138     fprintf (stdout, "%ls", wz);        // Outputs 6 bytes
31139     fprintf (stdout, "%ls", wn);        // Undefined because wn has no terminator
31140     fprintf (stdout, "%4ls", wz);       // Outputs 3 bytes
31141     fprintf (stdout, "%4ls", wn);       // Outputs 3 bytes; no terminator needed
31142     fprintf (stdout, "%9ls", wz);       // Outputs 6 bytes
31143     fprintf (stdout, "%9ls", wn);       // Outputs 9 bytes; no terminator needed
31144     fprintf (stdout, "%10ls", wz);      // Outputs 6 bytes
31145     fprintf (stdout, "%10ls", wn);      // Undefined because wn has no terminator

```

In the last line of the example, after processing three characters, nine bytes have been output. The fourth character must then be examined to determine whether it converts to one byte or more. If it converts to more than one byte, the output is only nine bytes. Since there is no fourth character in the array, the behavior is undefined.

### APPLICATION USAGE

If the application calling *fprintf()* has any objects of type **wint\_t** or **wchar\_t**, it must also include the **<wchar.h>** header to have these objects defined.

### RATIONALE

If an implementation detects that there are insufficient arguments for the format, it is recommended that the function should fail and report an [EINVAL] error.

### FUTURE DIRECTIONS

None.

**SEE ALSO**

Section 2.5 (on page 495), *fputc()*, *fscanf()*, *setlocale()*, *strfmon()*, *wcrtomb()*

XBD Chapter 7 (on page 135), *<inttypes.h>*, *<stdio.h>*, *<wchar.h>*

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

Aligned with ISO/IEC 9899:1990/Amendment 1:1995 (E). Specifically, the *l* (ell) qualifier can now be used with *c* and *s* conversion specifiers.

The *snprintf()* function is new in Issue 5.

**Issue 6**

Extensions beyond the ISO C standard are marked.

The normative text is updated to avoid use of the term “must” for application requirements.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The prototypes for *fprintf()*, *printf()*, *snprintf()*, and *sprintf()* are updated, and the XSI shading is removed from *snprintf()*.
- The description of *snprintf()* is aligned with the ISO C standard. Note that this supersedes the *snprintf()* description in The Open Group Base Resolution bwg98-006, which changed the behavior from Issue 5.
- The DESCRIPTION is updated.

The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion specification” consistently.

ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

An example of printing wide characters is added.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #161 is applied, updating the DESCRIPTION of the *0* flag.

Austin Group Interpretation 1003.1-2001 #170 is applied.

ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #68 (SD5-XSH-ERN-70) is applied, revising the description of *g* and *G*.

SD5-XSH-ERN-174 is applied.

The *dprintf()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 1.

Functionality relating to the *%n\$* form of conversion specification and the *<apostrophe>* flag is moved from the XSI option to the Base.

Changes are made related to support for finegrained timestamps.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0163 [302], XSH/TC1-2008/0164 [316], XSH/TC1-2008/0165 [316], XSH/TC1-2008/0166 [451,291], and XSH/TC1-2008/0167 [14] are applied.

31195 **NAME**

31196       fputc — put a byte on a stream

31197 **SYNOPSIS**

31198       #include &lt;stdio.h&gt;

31199       int fputc(int *c*, FILE \**stream*);31200 **DESCRIPTION**

31201 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 31202 conflict between the requirements described here and the ISO C standard is unintentional. This  
 31203 volume of POSIX.1-2008 defers to the ISO C standard.

31204       The *fputc()* function shall write the byte specified by *c* (converted to an **unsigned char**) to the  
 31205 output stream pointed to by *stream*, at the position indicated by the associated file-position  
 31206 indicator for the stream (if defined), and shall advance the indicator appropriately. If the file  
 31207 cannot support positioning requests, or if the stream was opened with append mode, the byte  
 31208 shall be appended to the output stream.

31209 CX       The last data modification and last file status change timestamps of the file shall be marked for  
 31210 update between the successful execution of *fputc()* and the next successful completion of a call  
 31211 to *fflush()* or *fclose()* on the same stream or a call to *exit()* or *abort()*.

31212 **RETURN VALUE**

31213       Upon successful completion, *fputc()* shall return the value it has written. Otherwise, it shall  
 31214 CX       return EOF, the error indicator for the stream shall be set, and *errno* shall be set to indicate the  
 31215 error.

31216 **ERRORS**

31217       The *fputc()* function shall fail if either the *stream* is unbuffered or the *stream*'s buffer needs to be  
 31218 flushed, and:

31219 CX       [EAGAIN]       The O\_NONBLOCK flag is set for the file descriptor underlying *stream* and  
 31220 the thread would be delayed in the write operation.

31221 CX       [EBADF]       The file descriptor underlying *stream* is not a valid file descriptor open for  
 31222 writing.

31223 CX       [EFBIG]       An attempt was made to write to a file that exceeds the maximum file size.

31224 XSI       [EFBIG]       An attempt was made to write to a file that exceeds the file size limit of the  
 31225 process.

31226 CX       [EFBIG]       The file is a regular file and an attempt was made to write at or beyond the  
 31227 offset maximum.

31228 CX       [EINTR]       The write operation was terminated due to the receipt of a signal, and no data  
 31229 was transferred.

31230 CX       [EIO]        A physical I/O error has occurred, or the process is a member of a background  
 31231 process group attempting to write to its controlling terminal, TOSTOP is set,  
 31232 the calling thread is not blocking SIGTTOU, the process is not ignoring  
 31233 SIGTTOU, and the process group of the process is orphaned. This error may  
 31234 also be returned under implementation-defined conditions.

31235 CX       [ENOSPC]       There was no free space remaining on the device containing the file.

31236 CX       [EPIPE]       An attempt is made to write to a pipe or FIFO that is not open for reading by  
 31237 any process. A SIGPIPE signal shall also be sent to the thread.

31238 The `fputc()` function may fail if:

31239 CX [ENOMEM] Insufficient storage space is available.

31240 CX [ENXIO] A request was made of a nonexistent device, or the request was outside the  
31241 capabilities of the device.

#### 31242 EXAMPLES

31243 None.

#### 31244 APPLICATION USAGE

31245 None.

#### 31246 RATIONALE

31247 None.

#### 31248 FUTURE DIRECTIONS

31249 None.

#### 31250 SEE ALSO

31251 Section 2.5 (on page 495), `ferror()`, `fopen()`, `getrlimit()`, `putc()`, `puts()`, `setbuf()`, `ulimit()`

31252 XBD `<stdio.h>`

#### 31253 CHANGE HISTORY

31254 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 31255 Issue 5

31256 Large File Summit extensions are added.

#### 31257 Issue 6

31258 Extensions beyond the ISO C standard are marked.

31259 The following new requirements on POSIX implementations derive from alignment with the  
31260 Single UNIX Specification:

- 31261 • The [EIO] and [EFBIG] mandatory error conditions are added.
- 31262 • The [ENOMEM] and [ENXIO] optional error conditions are added.

31263 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/37 is applied, updating the [EAGAIN]  
31264 error in the ERRORS section from “the process would be delayed” to “the thread would be  
31265 delayed”.

#### 31266 Issue 7

31267 Changes are made related to support for finegrained timestamps.

31268 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0168 [79] and XSH/TC1-2008/0169  
31269 [14] are applied.

31270 **NAME**

31271 fputs — put a string on a stream

31272 **SYNOPSIS**

31273 #include &lt;stdio.h&gt;

31274 int fputs(const char \*restrict s, FILE \*restrict stream);

31275 **DESCRIPTION**

31276 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 31277 conflict between the requirements described here and the ISO C standard is unintentional. This  
 31278 volume of POSIX.1-2008 defers to the ISO C standard.

31279 The *fputs()* function shall write the null-terminated string pointed to by *s* to the stream pointed  
 31280 to by *stream*. The terminating null byte shall not be written.

31281 CX The last data modification and last file status change timestamps of the file shall be marked for  
 31282 update between the successful execution of *fputs()* and the next successful completion of a call  
 31283 to *fflush()* or *fclose()* on the same stream or a call to *exit()* or *abort()*.

31284 **RETURN VALUE**

31285 Upon successful completion, *fputs()* shall return a non-negative number. Otherwise, it shall  
 31286 CX return EOF, set an error indicator for the stream, and set *errno* to indicate the error.

31287 **ERRORS**31288 Refer to *fputc()*.31289 **EXAMPLES**31290 **Printing to Standard Output**

31291 The following example gets the current time, converts it to a string using *localtime()* and  
 31292 *asctime()*, and prints it to standard output using *fputs()*. It then prints the number of minutes to  
 31293 an event for which it is waiting.

```
31294 #include <time.h>
31295 #include <stdio.h>
31296 ...
31297 time_t now;
31298 int minutes_to_event;
31299 ...
31300 time(&now);
31301 printf("The time is ");
31302 fputs(asctime(localtime(&now)), stdout);
31303 printf("There are still %d minutes to the event.\n",
31304        minutes_to_event);
31305 ...
```

31306 **APPLICATION USAGE**31307 The *puts()* function appends a <newline> while *fputs()* does not.

31308 This volume of POSIX.1-2008 requires that successful completion simply return a non-negative  
 31309 integer. There are at least three known different implementation conventions for this  
 31310 requirement:

- 31311 • Return a constant value.

- Return the last character written.
- Return the number of bytes written. Note that this implementation convention cannot be adhered to for strings longer than {INT\_MAX} bytes as the value would not be representable in the return type of the function. For backwards-compatibility, implementations can return the number of bytes for strings of up to {INT\_MAX} bytes, and return {INT\_MAX} for all longer strings.

**RATIONALE**

The `fputs()` function is one whose source code was specified in the referenced *The C Programming Language*. In the original edition, the function had no defined return value, yet many practical implementations would, as a side-effect, return the value of the last character written as that was the value remaining in the accumulator used as a return value. In the second edition of the book, either the fixed value 0 or EOF would be returned depending upon the return value of `ferror()`; however, for compatibility with extant implementations, several implementations would, upon success, return a positive value representing the last byte written.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 2.5](#) (on page 495), `fopen()`, `putc()`, `puts()`

XBD `<stdio.h>`

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 6**

Extensions beyond the ISO C standard are marked.

The `fputs()` prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

**Issue 7**

Changes are made related to support for finegrained timestamps.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0170 [174,412], XSH/TC1-2008/0171 [412], and XSH/TC1-2008/0172 [14] are applied.

31340 **NAME**31341 `fputwc` — put a wide-character code on a stream31342 **SYNOPSIS**31343 `#include <stdio.h>`31344 `#include <wchar.h>`31345 `wint_t fputwc(wchar_t wc, FILE *stream);`31346 **DESCRIPTION**

31347 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 31348 conflict between the requirements described here and the ISO C standard is unintentional. This  
 31349 volume of POSIX.1-2008 defers to the ISO C standard.

31350 The `fputwc()` function shall write the character corresponding to the wide-character code `wc` to  
 31351 the output stream pointed to by `stream`, at the position indicated by the associated file-position  
 31352 indicator for the stream (if defined), and advances the indicator appropriately. If the file cannot  
 31353 support positioning requests, or if the stream was opened with append mode, the character is  
 31354 appended to the output stream. If an error occurs while writing the character, the shift state of  
 31355 the output file is left in an undefined state.

31356 CX The last data modification and last file status change timestamps of the file shall be marked for  
 31357 update between the successful execution of `fputwc()` and the next successful completion of a call  
 31358 to `fflush()` or `fclose()` on the same stream or a call to `exit()` or `abort()`.

31359 The `fputwc()` function shall not change the setting of `errno` if successful.

31360 **RETURN VALUE**

31361 Upon successful completion, `fputwc()` shall return `wc`. Otherwise, it shall return WEOF, the error  
 31362 CX indicator for the stream shall be set, and `errno` shall be set to indicate the error.

31363 **ERRORS**

31364 The `fputwc()` function shall fail if either the stream is unbuffered or data in the `stream`'s buffer  
 31365 needs to be written, and:

31366 CX [EAGAIN] The O\_NONBLOCK flag is set for the file descriptor underlying `stream` and  
 31367 the thread would be delayed in the write operation.

31368 CX [EBADF] The file descriptor underlying `stream` is not a valid file descriptor open for  
 31369 writing.

31370 CX [EFBIG] An attempt was made to write to a file that exceeds the maximum file size or  
 31371 the file size limit of the process.

31372 CX [EFBIG] The file is a regular file and an attempt was made to write at or beyond the  
 31373 offset maximum associated with the corresponding stream.

31374 [EILSEQ] The wide-character code `wc` does not correspond to a valid character.

31375 CX [EINTR] The write operation was terminated due to the receipt of a signal, and no data  
 31376 was transferred.

31377 CX [EIO] A physical I/O error has occurred, or the process is a member of a background  
 31378 process group attempting to write to its controlling terminal, TOSTOP is set,  
 31379 the calling thread is not blocking SIGTTOU, the process is not ignoring  
 31380 SIGTTOU, and the process group of the process is orphaned. This error may  
 31381 also be returned under implementation-defined conditions.



31382 CX [ENOSPC] There was no free space remaining on the device containing the file.

31383 CX [EPIPE] An attempt is made to write to a pipe or FIFO that is not open for reading by

31384 any process. A SIGPIPE signal shall also be sent to the thread.

31385 The *fputc()* function may fail if:

31386 CX [ENOMEM] Insufficient storage space is available.

31387 CX [ENXIO] A request was made of a nonexistent device, or the request was outside the

31388 capabilities of the device.

#### 31389 EXAMPLES

31390 None.

#### 31391 APPLICATION USAGE

31392 None.

#### 31393 RATIONALE

31394 None.

#### 31395 FUTURE DIRECTIONS

31396 None.

#### 31397 SEE ALSO

31398 Section 2.5 (on page 495), *ferror()*, *fopen()*, *setbuf()*, *ulimit()*

31399 XBD `<stdio.h>`, `<wchar.h>`

#### 31400 CHANGE HISTORY

31401 First released in Issue 4. Derived from the MSE working draft.

#### 31402 Issue 5

31403 Aligned with ISO/IEC 9899:1990/Amendment 1:1995 (E). Specifically, the type of argument *wc*

31404 is changed from `wint_t` to `wchar_t`.

31405 The Optional Header (OH) marking is removed from `<stdio.h>`.

31406 Large File Summit extensions are added.

#### 31407 Issue 6

31408 Extensions beyond the ISO C standard are marked.

31409 The following new requirements on POSIX implementations derive from alignment with the

31410 Single UNIX Specification:

- 31411 • The [EFBIG] and [EIO] mandatory error conditions are added.
- 31412 • The [ENOMEM] and [ENXIO] optional error conditions are added.

31413 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/38 is applied, updating the [EAGAIN]

31414 error in the ERRORS section from “the process would be delayed” to “the thread would be

31415 delayed”.

#### 31416 Issue 7

31417 Changes are made related to support for finegrained timestamps.

31418 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0173 [105], XSH/TC1-2008/0174 [79],

31419 and XSH/TC1-2008/0175 [14] are applied.



31420 **NAME**31421 `fputws` — put a wide-character string on a stream31422 **SYNOPSIS**31423 `#include <stdio.h>`31424 `#include <wchar.h>`31425 `int fputws(const wchar_t *restrict ws, FILE *restrict stream);`31426 **DESCRIPTION**

31427 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 31428 conflict between the requirements described here and the ISO C standard is unintentional. This  
 31429 volume of POSIX.1-2008 defers to the ISO C standard.

31430 The `fputws()` function shall write a character string corresponding to the (null-terminated) wide-  
 31431 character string pointed to by `ws` to the stream pointed to by `stream`. No character corresponding  
 31432 to the terminating null wide-character code shall be written.

31433 CX The last data modification and last file status change timestamps of the file shall be marked for  
 31434 update between the successful execution of `fputws()` and the next successful completion of a call  
 31435 to `fflush()` or `fclose()` on the same stream or a call to `exit()` or `abort()`.

31436 **RETURN VALUE**

31437 Upon successful completion, `fputws()` shall return a non-negative number. Otherwise, it shall  
 31438 CX return `-1`, set an error indicator for the stream, and set `errno` to indicate the error.

31439 **ERRORS**31440 Refer to `fputwc()`.31441 **EXAMPLES**

31442 None.

31443 **APPLICATION USAGE**31444 The `fputws()` function does not append a `<newline>`.

31445 This volume of POSIX.1-2008 requires that successful completion simply return a non-negative  
 31446 integer. There are at least three known different implementation conventions for this  
 31447 requirement:

- 31448 • Return a constant value.
- 31449 • Return the last character written.
- 31450 • Return the number of bytes written. Note that this implementation convention cannot be  
 31451 adhered to for strings longer than `{INT_MAX}` bytes as the value would not be  
 31452 representable in the return type of the function. For backwards-compatibility,  
 31453 implementations can return the number of bytes for strings of up to `{INT_MAX}` bytes, and  
 31454 return `{INT_MAX}` for all longer strings.

31455 **RATIONALE**

31456 None.

31457 **FUTURE DIRECTIONS**

31458 None.

31459 **SEE ALSO**31460 [Section 2.5](#) (on page 495), `fopen()`31461 XBD `<stdio.h>`, `<wchar.h>`

31462 **CHANGE HISTORY**

31463 First released in Issue 4. Derived from the MSE working draft.

31464 **Issue 5**

31465 The Optional Header (OH) marking is removed from `<stdio.h>`.

31466 **Issue 6**

31467 Extensions beyond the ISO C standard are marked.

31468 The `fputws()` prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

31469 **Issue 7**

31470 Changes are made related to support for finegrained timestamps.

31471 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0176 [412] and XSH/TC1-2008/0177  
31472 [14] are applied.

31473 **NAME**

31474 fread — binary input

31475 **SYNOPSIS**

```
31476 #include <stdio.h>
31477 size_t fread(void *restrict ptr, size_t size, size_t nitems,
31478 FILE *restrict stream);
```

31479 **DESCRIPTION**

31480 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 31481 conflict between the requirements described here and the ISO C standard is unintentional. This  
 31482 volume of POSIX.1-2008 defers to the ISO C standard.

31483 The *fread()* function shall read into the array pointed to by *ptr* up to *nitems* elements whose size  
 31484 is specified by *size* in bytes, from the stream pointed to by *stream*. For each object, *size* calls shall  
 31485 be made to the *fgetc()* function and the results stored, in the order read, in an array of **unsigned**  
 31486 **char** exactly overlaying the object. The file position indicator for the stream (if defined) shall be  
 31487 advanced by the number of bytes successfully read. If an error occurs, the resulting value of the  
 31488 file position indicator for the stream is unspecified. If a partial element is read, its value is  
 31489 unspecified.

31490 CX The *fread()* function may mark the last data access timestamp of the file associated with *stream*  
 31491 for update. The last data access timestamp shall be marked for update by the first successful  
 31492 execution of *fgetc()*, *fgets()*, *fread()*, *fscanf()*, *getc()*, *getchar()*, *getdelim()*, *getline()*, *gets()*, or  
 31493 *scanf()* using *stream* that returns data not supplied by a prior call to *ungetc()*.

31494 **RETURN VALUE**

31495 Upon successful completion, *fread()* shall return the number of elements successfully read which  
 31496 is less than *nitems* only if a read error or end-of-file is encountered. If *size* or *nitems* is 0, *fread()*  
 31497 shall return 0 and the contents of the array and the state of the stream remain unchanged.  
 31498 CX Otherwise, if a read error occurs, the error indicator for the stream shall be set, and *errno* shall  
 31499 be set to indicate the error.

31500 **ERRORS**31501 Refer to *fgetc()*.31502 **EXAMPLES**31503 **Reading from a Stream**

31504 The following example transfers a single 100-byte fixed length record from the *fp* stream into the  
 31505 array pointed to by *buf*.

```
31506 #include <stdio.h>
31507 ...
31508 size_t elements_read;
31509 char buf[100];
31510 FILE *fp;
31511 ...
31512 elements_read = fread(buf, sizeof(buf), 1, fp);
31513 ...
```

31514 If a read error occurs, *elements\_read* will be zero but the number of bytes read from the stream  
 31515 could be anything from zero to *sizeof(buf)*–1.

31516 The following example reads multiple single-byte elements from the *fp* stream into the array  
 31517 pointed to by *buf*.

```
31518     #include <stdio.h>
31519     ...
31520     size_t bytes_read;
31521     char buf[100];
31522     FILE *fp;
31523     ...
31524     bytes_read = fread(buf, 1, sizeof(buf), fp);
31525     ...
```

31526 If a read error occurs, *bytes\_read* will contain the number of bytes read from the stream.

#### 31527 APPLICATION USAGE

31528 The *ferror()* or *feof()* functions must be used to distinguish between an error condition and an  
31529 end-of-file condition.

31530 Because of possible differences in element length and byte ordering, files written using *fwrite()*  
31531 are application-dependent, and possibly cannot be read using *fread()* by a different application  
31532 or by the same application on a different processor.

#### 31533 RATIONALE

31534 None.

#### 31535 FUTURE DIRECTIONS

31536 None.

#### 31537 SEE ALSO

31538 [Section 2.5](#) (on page 495), *feof()*, *ferror()*, *fgetc()*, *fopen()*, *fscanf()*, *getc()*, *gets()*

31539 XBD [<stdio.h>](#)

#### 31540 CHANGE HISTORY

31541 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 31542 Issue 6

31543 Extensions beyond the ISO C standard are marked.

31544 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 31545 • The *fread()* prototype is updated.
- 31546 • The DESCRIPTION is updated to describe how the bytes from a call to *fgetc()* are stored.

#### 31547 Issue 7

31548 Changes are made related to support for finegrained timestamps.

31549 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0178 [232] and XSH/TC1-2008/0179  
31550 [14] are applied.

31551 **NAME**

31552 free — free allocated memory

31553 **SYNOPSIS**

31554 #include &lt;stdlib.h&gt;

31555 void free(void \*ptr);

31556 **DESCRIPTION**

31557 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 31558 conflict between the requirements described here and the ISO C standard is unintentional. This  
 31559 volume of POSIX.1-2008 defers to the ISO C standard.

31560 The *free()* function shall cause the space pointed to by *ptr* to be deallocated; that is, made  
 31561 available for further allocation. If *ptr* is a null pointer, no action shall occur. Otherwise, if the  
 31562 argument does not match a pointer earlier returned by a function in POSIX.1-2008 that allocates  
 31563 memory as if by *malloc()*, or if the space has been deallocated by a call to *free()* or *realloc()*, the  
 31564 behavior is undefined.

31565 Any use of a pointer that refers to freed space results in undefined behavior.

31566 **RETURN VALUE**31567 The *free()* function shall not return a value.31568 **ERRORS**

31569 No errors are defined.

31570 **EXAMPLES**

31571 None.

31572 **APPLICATION USAGE**

31573 There is now no requirement for the implementation to support the inclusion of &lt;malloc.h&gt;.

31574 **RATIONALE**

31575 None.

31576 **FUTURE DIRECTIONS**

31577 None.

31578 **SEE ALSO**31579 *calloc()*, *malloc()*, *posix\_memalign()*, *realloc()*

31580 XBD &lt;stdlib.h&gt;

31581 **CHANGE HISTORY**

31582 First released in Issue 1. Derived from Issue 1 of the SVID.

31583 **Issue 6**31584 Reference to the *valloc()* function is removed.31585 **Issue 7**

31586 The DESCRIPTION is updated to clarify that if the pointer returned is not by a function that  
 31587 allocates memory as if by *malloc()*, then the behavior is undefined.

## NAME

freeaddrinfo, getaddrinfo — get address information

## SYNOPSIS

```
#include <sys/socket.h>
#include <netdb.h>

void freeaddrinfo(struct addrinfo *ai);
int getaddrinfo(const char *restrict nodename,
                const char *restrict servname,
                const struct addrinfo *restrict hints,
                struct addrinfo **restrict res);
```

## DESCRIPTION

The *freeaddrinfo()* function shall free one or more **addrinfo** structures returned by *getaddrinfo()*, along with any additional storage associated with those structures. If the *ai\_next* field of the structure is not null, the entire list of structures shall be freed. The *freeaddrinfo()* function shall support the freeing of arbitrary sublists of an **addrinfo** list originally returned by *getaddrinfo()*.

The *getaddrinfo()* function shall translate the name of a service location (for example, a host name) and/or a service name and shall return a set of socket addresses and associated information to be used in creating a socket with which to address the specified service.

**Note:** In many cases it is implemented by the Domain Name System, as documented in RFC 1034, RFC 1035, and RFC 1886.

The *freeaddrinfo()* and *getaddrinfo()* functions shall be thread-safe.

The *nodename* and *servname* arguments are either null pointers or pointers to null-terminated strings. One or both of these two arguments shall be supplied by the application as a non-null pointer.

The format of a valid name depends on the address family or families. If a specific family is not given and the name could be interpreted as valid within multiple supported families, the implementation shall attempt to resolve the name in all supported families and, in absence of errors, one or more results shall be returned.

If the *nodename* argument is not null, it can be a descriptive name or can be an address string. If the specified address family is AF\_INET, AF\_INET6, or AF\_UNSPEC, valid descriptive names include host names. If the specified address family is AF\_INET or AF\_UNSPEC, address strings using Internet standard dot notation as specified in *inet\_addr()* are valid.

If the specified address family is AF\_INET6 or AF\_UNSPEC, standard IPv6 text forms described in *inet\_ntop()* are valid.

If *nodename* is not null, the requested service location is named by *nodename*; otherwise, the requested service location is local to the caller.

If *servname* is null, the call shall return network-level addresses for the specified *nodename*. If *servname* is not null, it is a null-terminated character string identifying the requested service. This can be either a descriptive name or a numeric representation suitable for use with the address family or families. If the specified address family is AF\_INET, AF\_INET6, or AF\_UNSPEC, the service can be specified as a string specifying a decimal port number.

If the *hints* argument is not null, it refers to a structure containing input values that directs the operation by providing options and by limiting the returned information to a specific socket type, address family, and/or protocol, as described below. The application shall ensure that each of the *ai\_addrlen*, *ai\_addr*, *ai\_canonname*, and *ai\_next* members, as well as each of the non-standard additional members, if any, of this *hints* structure is initialized. If any of these members has a

value other than the value that would result from default initialization, the behavior is implementation-defined. A value of `AF_UNSPEC` for *ai\_family* means that the caller shall accept any address family. A value of zero for *ai\_socktype* means that the caller shall accept any socket type. A value of zero for *ai\_protocol* means that the caller shall accept any protocol. If *hints* is a null pointer, the behavior shall be as if it referred to a structure containing the value zero for the *ai\_flags*, *ai\_socktype*, and *ai\_protocol* fields, and `AF_UNSPEC` for the *ai\_family* field.

The *ai\_flags* field to which the *hints* parameter points shall be set to zero or be the bitwise-inclusive OR of one or more of the values `AI_PASSIVE`, `AI_CANONNAME`, `AI_NUMERICHOST`, `AI_NUMERICSERV`, `AI_V4MAPPED`, `AI_ALL`, and `AI_ADDRCONFIG`.

If the `AI_PASSIVE` flag is specified, the returned address information shall be suitable for use in binding a socket for accepting incoming connections for the specified service. In this case, if the *nodename* argument is null, then the IP address portion of the socket address structure shall be set to `INADDR_ANY` for an IPv4 address or `IN6ADDR_ANY_INIT` for an IPv6 address. If the `AI_PASSIVE` flag is not specified, the returned address information shall be suitable for a call to `connect()` (for a connection-mode protocol) or for a call to `connect()`, `sendto()`, or `sendmsg()` (for a connectionless protocol). In this case, if the *nodename* argument is null, then the IP address portion of the socket address structure shall be set to the loopback address. The `AI_PASSIVE` flag shall be ignored if the *nodename* argument is not null.

If the `AI_CANONNAME` flag is specified and the *nodename* argument is not null, the function shall attempt to determine the canonical name corresponding to *nodename* (for example, if *nodename* is an alias or shorthand notation for a complete name).

**Note:** Since different implementations use different conceptual models, the terms “canonical name” and “alias” cannot be precisely defined for the general case. However, Domain Name System implementations are expected to interpret them as they are used in RFC 1034.

A numeric host address string is not a “name”, and thus does not have a “canonical name” form; no address to host name translation is performed. See below for handling of the case where a canonical name cannot be obtained.

If the `AI_NUMERICHOST` flag is specified, then a non-null *nodename* string supplied shall be a numeric host address string. Otherwise, an `[EAI_NONAME]` error is returned. This flag shall prevent any type of name resolution service (for example, the DNS) from being invoked.

If the `AI_NUMERICSERV` flag is specified, then a non-null *servname* string supplied shall be a numeric port string. Otherwise, an `[EAI_NONAME]` error shall be returned. This flag shall prevent any type of name resolution service (for example, NIS+) from being invoked.

By default, with an *ai\_family* of `AF_INET6`, `getaddrinfo()` shall return only IPv6 addresses. If the `AI_V4MAPPED` flag is specified along with an *ai\_family* of `AF_INET6`, then `getaddrinfo()` shall return IPv4-mapped IPv6 addresses on finding no matching IPv6 addresses. The `AI_V4MAPPED` flag shall be ignored unless *ai\_family* equals `AF_INET6`. If the `AI_ALL` flag is used with the `AI_V4MAPPED` flag, then `getaddrinfo()` shall return all matching IPv6 and IPv4 addresses. The `AI_ALL` flag without the `AI_V4MAPPED` flag shall be ignored.

If the `AI_ADDRCONFIG` flag is specified, IPv4 addresses shall be returned only if an IPv4 address is configured on the local system, and IPv6 addresses shall be returned only if an IPv6 address is configured on the local system.

The *ai\_socktype* field to which argument *hints* points specifies the socket type for the service, as defined in `socket()`. If a specific socket type is not given (for example, a value of zero) and the service name could be interpreted as valid with multiple supported socket types, the implementation shall attempt to resolve the service name for all supported socket types and, in the absence of errors, all possible results shall be returned. A non-zero socket type value shall



limit the returned information to values with the specified socket type.

If the *ai\_family* field to which *hints* points has the value AF\_UNSPEC, addresses shall be returned for use with any address family that can be used with the specified *nodename* and/or *servname*. Otherwise, addresses shall be returned for use only with the specified address family. If *ai\_family* is not AF\_UNSPEC and *ai\_protocol* is not zero, then addresses shall be returned for use only with the specified address family and protocol; the value of *ai\_protocol* shall be interpreted as in a call to the *socket()* function with the corresponding values of *ai\_family* and *ai\_protocol*.

## RETURN VALUE

A zero return value for *getaddrinfo()* indicates successful completion; a non-zero return value indicates failure. The possible values for the failures are listed in the ERRORS section.

Upon successful return of *getaddrinfo()*, the location to which *res* points shall refer to a linked list of **addrinfo** structures, each of which shall specify a socket address and information for use in creating a socket with which to use that socket address. The list shall include at least one **addrinfo** structure. The *ai\_next* field of each structure contains a pointer to the next structure on the list, or a null pointer if it is the last structure on the list. Each structure on the list shall include values for use with a call to the *socket()* function, and a socket address for use with the *connect()* function or, if the AI\_PASSIVE flag was specified, for use with the *bind()* function. The fields *ai\_family*, *ai\_socktype*, and *ai\_protocol* shall be usable as the arguments to the *socket()* function to create a socket suitable for use with the returned address. The fields *ai\_addr* and *ai\_addrlen* are usable as the arguments to the *connect()* or *bind()* functions with such a socket, according to the AI\_PASSIVE flag.

If *nodename* is not null, and if requested by the AI\_CANONNAME flag, the *ai\_canonname* field of the first returned **addrinfo** structure shall point to a null-terminated string containing the canonical name corresponding to the input *nodename*; if the canonical name is not available, then *ai\_canonname* shall refer to the *nodename* argument or a string with the same contents. The contents of the *ai\_flags* field of the returned structures are undefined.

All fields in socket address structures returned by *getaddrinfo()* that are not filled in through an explicit argument (for example, *sin6\_flowinfo*) shall be set to zero.

**Note:** This makes it easier to compare socket address structures.

## ERRORS

The *getaddrinfo()* function shall fail and return the corresponding error value if:

[EAI\_AGAIN] The name could not be resolved at this time. Future attempts may succeed.

[EAI\_BADFLAGS]

The *flags* parameter had an invalid value.

[EAI\_FAIL] A non-recoverable error occurred when attempting to resolve the name.

[EAI\_FAMILY] The address family was not recognized.

[EAI\_MEMORY] There was a memory allocation failure when trying to allocate storage for the return value.

[EAI\_NONAME] The name does not resolve for the supplied parameters.

Neither *nodename* nor *servname* were supplied. At least one of these shall be supplied.

[EAI\_SERVICE] The service passed was not recognized for the specified socket type.



31724 [EAI\_SOCKTYPE]  
 31725 The intended socket type was not recognized.  
 31726 [EAI\_SYSTEM] A system error occurred; the error code can be found in *errno*.

### 31727 EXAMPLES

31728 The following (incomplete) program demonstrates the use of *getaddrinfo()* to obtain the socket  
 31729 address structure(s) for the service named in the program's command-line argument. The  
 31730 program then loops through each of the address structures attempting to create and bind a  
 31731 socket to the address, until it performs a successful *bind()*.

```

31732 #include <stdio.h>
31733 #include <stdlib.h>
31734 #include <unistd.h>
31735 #include <string.h>
31736 #include <sys/socket.h>
31737 #include <netdb.h>

31738 int
31739 main(int argc, char *argv[])
31740 {
31741     struct addrinfo *result, *rp;
31742     int sfd, s;

31743     if (argc != 2) {
31744         fprintf(stderr, "Usage: %s port\n", argv[0]);
31745         exit(EXIT_FAILURE);
31746     }

31747     struct addrinfo hints = {0};
31748     hints.ai_family = AF_UNSPEC;
31749     hints.ai_socktype = SOCK_DGRAM;
31750     hints.ai_flags = AI_PASSIVE;
31751     hints.ai_protocol = 0;

31752     s = getaddrinfo(NULL, argv[1], &hints, &result);
31753     if (s != 0) {
31754         fprintf(stderr, "getaddrinfo: %s\n", gai_strerror(s));
31755         exit(EXIT_FAILURE);
31756     }

31757     /* getaddrinfo() returns a list of address structures.
31758     Try each address until a successful bind().
31759     If socket(2) (or bind(2)) fails, close the socket
31760     and try the next address. */

31761     for (rp = result; rp != NULL; rp = rp->ai_next) {
31762         sfd = socket(rp->ai_family, rp->ai_socktype,
31763                     rp->ai_protocol);
31764         if (sfd == -1)
31765             continue;

31766         if (bind(sfd, rp->ai_addr, rp->ai_addrlen) == 0)
31767             break;          /* Success */

31768         close(sfd);
31769     }
  
```

```

31770         if (rp == NULL) {           /* No address succeeded */
31771             fprintf(stderr, "Could not bind\n");
31772             exit(EXIT_FAILURE);
31773         }
31774         freeaddrinfo(result);        /* No longer needed */
31775
31776         /* ... use socket bound to sockfd ... */

```

## APPLICATION USAGE

If the caller handles only TCP and not UDP, for example, then the *ai\_protocol* member of the *hints* structure should be set to IPPROTO\_TCP when *getaddrinfo()* is called.

If the caller handles only IPv4 and not IPv6, then the *ai\_family* member of the *hints* structure should be set to AF\_INET when *getaddrinfo()* is called.

Although it is common practice to initialize the *hints* structure using:

```

struct addrinfo hints;
memset(&hints, 0, sizeof hints);

```

this method is not portable according to this standard, because the structure can contain pointer or floating-point members that are not required to have an all-bits-zero representation after default initialization. Portable methods make use of default initialization; for example:

```

struct addrinfo hints = { 0 };

```

or:

```

static struct addrinfo hints_init;
struct addrinfo hints = hints_init;

```

A future version of this standard may require that a pointer object with an all-bits-zero representation is a null pointer, and that **addrinfo** does not have any floating-point members if a floating-point object with an all-bits-zero representation does not have the value 0.0.

The term “canonical name” is misleading; it is taken from the Domain Name System (RFC 2181). It should be noted that the canonical name is a result of alias processing, and not necessarily a unique attribute of a host, address, or set of addresses. See RFC 2181 for more discussion of this in the Domain Name System context.

## RATIONALE

None.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*connect()*, *endservent()*, *gai\_strerror()*, *getnameinfo()*, *socket()*

XBD [\*\*<netdb.h>\*\*](#), [\*\*<sys/socket.h>\*\*](#)

## CHANGE HISTORY

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

The **restrict** keyword is added to the *getaddrinfo()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/19 is applied, adding three notes to the DESCRIPTION and adding text to the APPLICATION USAGE related to the term “canonical

- 31812 name". A reference to RFC 2181 is also added to the Informative References.
- 31813 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/20 is applied, making changes for
- 31814 alignment with IPv6. These include the following:
- 31815 • Adding AI\_V4MAPPED, AI\_ALL, and AI\_ADDRCONFIG to the allowed values for the
  - 31816 *ai\_flags* field
  - 31817 • Adding a description of AI\_ADDRCONFIG
  - 31818 • Adding a description of the consequences of ignoring the AI\_PASSIVE flag.
- 31819 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/39 is applied, changing "corresponding
- 31820 value" to "corresponding error value" in the ERRORS section.
- 31821 **Issue 7**
- 31822 Austin Group Interpretation 1003.1-2001 #013 is applied.
- 31823 Austin Group Interpretation 1003.1-2001 #146 is applied, updating the DESCRIPTION.
- 31824 An example is added.

31825 **NAME**

31826       freelocale — free resources allocated for a locale object

31827 **SYNOPSIS**

```
31828 CX      #include <locale.h>
31829          void freelocale(locale_t locobj);
```

31830 **DESCRIPTION**

31831       The *freelocale()* function shall cause the resources allocated for a locale object returned by a call  
31832       to the *newlocale()* or *duplocale()* functions to be released.

31833       The behavior is undefined if the *locobj* argument is the special locale object  
31834       LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

31835       Any use of a locale object that has been freed results in undefined behavior.

31836 **RETURN VALUE**

31837       None.

31838 **ERRORS**

31839       None.

31840 **EXAMPLES**31841       **Freeing Up a Locale Object**

31842       The following example shows a code fragment to free a locale object created by *newlocale()*:

```
31843       #include <locale.h>
31844       ...
31845       /* Every locale object allocated with newlocale() should be
31846        * freed using freelocale():
31847        */
31848       locale_t loc;
31849       /* Get the locale. */
31850       loc = newlocale (LC_CTYPE_MASK | LC_TIME_MASK, "locname", NULL);
31851       /* ... Use the locale object ... */
31852       ...
31853       /* Free the locale object resources. */
31854       freelocale (loc);
```

31855 **APPLICATION USAGE**

31856       None.

31857 **RATIONALE**

31858       None.

31859 **FUTURE DIRECTIONS**

31860       None.

31861 **SEE ALSO**31862       *duplocale()*, *newlocale()*, *uselocale()*

31863       XBD &lt;locale.h&gt;

CHANGE HISTORY

- First released in Issue 7.
- POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0180 [283] is applied.

**NAME**

freopen — open a stream

**SYNOPSIS**

#include &lt;stdio.h&gt;

```
FILE *freopen(const char *restrict pathname, const char *restrict mode,
              FILE *restrict stream);
```

**DESCRIPTION**

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *freopen()* function shall first attempt to flush the stream associated with *stream* as if by a call to *fflush(stream)*. Failure to flush the stream successfully shall be ignored. If *pathname* is not a null pointer, *freopen()* shall close any file descriptor associated with *stream*. Failure to close the file descriptor successfully shall be ignored. The error and end-of-file indicators for the stream shall be cleared.

The *freopen()* function shall open the file whose pathname is the string pointed to by *pathname* and associate the stream pointed to by *stream* with it. The *mode* argument shall be used just as in *fopen()*.

The original stream shall be closed regardless of whether the subsequent open succeeds.

If *pathname* is a null pointer, the *freopen()* function shall attempt to change the mode of the stream to that specified by *mode*, as if the name of the file currently associated with the stream had been used. In this case, the file descriptor associated with the stream need not be closed if the call to *freopen()* succeeds. It is implementation-defined which changes of mode are permitted (if any), and under what circumstances.

After a successful call to the *freopen()* function, the orientation of the stream shall be cleared, the encoding rule shall be cleared, and the associated **mbstate\_t** object shall be set to describe an initial conversion state.

If *pathname* is not a null pointer, or if *pathname* is a null pointer and the specified mode change necessitates the file descriptor associated with the stream to be closed and reopened, the file descriptor associated with the reopened stream shall be allocated and opened as if by a call to *open()* with the following flags:

| <i>freopen()</i> Mode                 | <i>open()</i> Flags       |
|---------------------------------------|---------------------------|
| <i>r</i> or <i>rb</i>                 | O_RDONLY                  |
| <i>w</i> or <i>wb</i>                 | O_WRONLY O_CREAT O_TRUNC  |
| <i>a</i> or <i>ab</i>                 | O_WRONLY O_CREAT O_APPEND |
| <i>r+</i> or <i>rb+</i> or <i>r+b</i> | O_RDWR                    |
| <i>w+</i> or <i>wb+</i> or <i>w+b</i> | O_RDWR O_CREAT O_TRUNC    |
| <i>a+</i> or <i>ab+</i> or <i>a+b</i> | O_RDWR O_CREAT O_APPEND   |

**RETURN VALUE**

Upon successful completion, *freopen()* shall return the value of *stream*. Otherwise, a null pointer shall be returned, and *errno* shall be set to indicate the error.

31908 **ERRORS**31909 The *freopen()* function shall fail if:

31910 CX [EACCES] Search permission is denied on a component of the path prefix, or the file  
 31911 exists and the permissions specified by *mode* are denied, or the file does not  
 31912 exist and write permission is denied for the parent directory of the file to be  
 31913 created.

31914 CX [EBADF] The file descriptor underlying the stream is not a valid file descriptor when  
 31915 *pathname* is a null pointer.

31916 CX [EINTR] A signal was caught during *freopen()*.

31917 CX [EISDIR] The named file is a directory and *mode* requires write access.

31918 CX [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*  
 31919 argument.

31920 CX [EMFILE] All file descriptors available to the process are currently open.

31921 CX [ENAMETOOLONG]

31922 The length of a component of a pathname is longer than {NAME\_MAX}.

31923 CX [ENFILE] The maximum allowable number of files is currently open in the system.

31924 CX [ENOENT] The *mode* string begins with 'r' and a component of *pathname* does not name  
 31925 an existing file, or *mode* begins with 'w' or 'a' and a component of the path  
 31926 prefix of *pathname* does not name an existing file, or *pathname* is an empty  
 31927 string.

31928 CX [ENOENT] or [ENOTDIR]

31929 The *pathname* argument contains at least one non-*<slash>* character and ends  
 31930 with one or more trailing *<slash>* characters. If *pathname* without the trailing  
 31931 *<slash>* characters would name an existing file, an [ENOENT] error shall not  
 31932 occur.

31933 CX [ENOSPC] The directory or file system that would contain the new file cannot be  
 31934 expanded, the file does not exist, and it was to be created.

31935 CX [ENOTDIR] A component of the path prefix names an existing file that is neither a  
 31936 directory nor a symbolic link to a directory, or the *pathname* argument contains  
 31937 at least one non-*<slash>* character and ends with one or more trailing *<slash>*  
 31938 characters and the last pathname component names an existing file that is  
 31939 neither a directory nor a symbolic link to a directory.

31940 CX [ENXIO] The named file is a character special or block special file, and the device  
 31941 associated with this special file does not exist.

31942 CX [EOVERFLOW] The named file is a regular file and the size of the file cannot be represented  
 31943 correctly in an object of type *off\_t*.

31944 CX [EROFS] The named file resides on a read-only file system and *mode* requires write  
 31945 access.

31946 The *freopen()* function may fail if:

31947 CX [EBADF] The mode with which the file descriptor underlying the stream was opened  
 31948 does not support the requested mode when *pathname* is a null pointer.

|       |    |                |                                                                                        |
|-------|----|----------------|----------------------------------------------------------------------------------------|
| 31949 | CX | [EINVAL]       | The value of the <i>mode</i> argument is not valid.                                    |
| 31950 | CX | [ELOOP]        | More than {SYMLOOP_MAX} symbolic links were encountered during                         |
| 31951 |    |                | resolution of the <i>path</i> argument.                                                |
| 31952 | CX | [ENAMETOOLONG] |                                                                                        |
| 31953 |    |                | The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a               |
| 31954 |    |                | symbolic link produced an intermediate result with a length that exceeds               |
| 31955 |    |                | {PATH_MAX}.                                                                            |
| 31956 | CX | [ENOMEM]       | Insufficient storage space is available.                                               |
| 31957 | CX | [ENXIO]        | A request was made of a nonexistent device, or the request was outside the             |
| 31958 |    |                | capabilities of the device.                                                            |
| 31959 | CX | [ETXTBSY]      | The file is a pure procedure (shared text) file that is being executed and <i>mode</i> |
| 31960 |    |                | requires write access.                                                                 |

## 31961 EXAMPLES

### 31962 Directing Standard Output to a File

31963 The following example logs all standard output to the */tmp/logfile* file.

```
31964 #include <stdio.h>
31965 ...
31966 FILE *fp;
31967 ...
31968 fp = freopen ("/tmp/logfile", "a+", stdout);
31969 ...
```

## 31970 APPLICATION USAGE

31971 The *freopen()* function is typically used to attach the pre-opened *streams* associated with *stdin*,  
 31972 *stdout*, and *stderr* to other files.

31973 Since implementations are not required to support any stream mode changes when the *pathname*  
 31974 argument is NULL, portable applications cannot rely on the use of *freopen()* to change the stream  
 31975 mode, and use of this feature is discouraged. The feature was originally added to the ISO C  
 31976 standard in order to facilitate changing *stdin* and *stdout* to binary mode. Since a 'b' character in  
 31977 the mode has no effect on POSIX systems, this use of the feature is unnecessary in POSIX  
 31978 applications. However, even though the 'b' is ignored, a successful call to *freopen*(NULL, "wb",  
 31979 *stdout*) does have an effect. In particular, for regular files it truncates the file and sets the file-  
 31980 position indicator for the stream to the start of the file. It is possible that these side-effects are an  
 31981 unintended consequence of the way the feature is specified in the ISO/IEC 9899:1999 standard,  
 31982 but unless or until the ISO C standard is changed, applications which successfully call  
 31983 *freopen*(NULL, "wb", *stdout*) will behave in unexpected ways on conforming systems in situations  
 31984 such as:

```
31985 { appl file1; appl file2; } > file3
```

31986 which will result in **file3** containing only the output from the second invocation of *appl*.

## 31987 RATIONALE

31988 None.



31989 **FUTURE DIRECTIONS**

31990 None.

31991 **SEE ALSO**31992 Section 2.5 (on page 495), *fclose()*, *fdopen()*, *fflush()*, *fmemopen()*, *fopen()*, *mbsinit()*, *open()*,  
31993 *open\_memstream()*

31994 XBD &lt;stdio.h&gt;

31995 **CHANGE HISTORY**

31996 First released in Issue 1. Derived from Issue 1 of the SVID.

31997 **Issue 5**31998 The DESCRIPTION is updated to indicate that the orientation of the stream is cleared and the  
31999 conversion state of the stream is set to an initial conversion state by a successful call to the  
32000 *freopen()* function.

32001 Large File Summit extensions are added.

32002 **Issue 6**

32003 Extensions beyond the ISO C standard are marked.

32004 The following new requirements on POSIX implementations derive from alignment with the  
32005 Single UNIX Specification:

- 32006 • In the DESCRIPTION, text is added to indicate setting of the offset maximum in the open
- 32007 file description. This change is to support large files.
- 32008 • In the ERRORS section, the [Eoverflow] condition is added. This change is to support
- 32009 large files.
- 32010 • The [ELOOP] mandatory error condition is added.
- 32011 • A second [ENAMETOOLONG] is added as an optional error condition.
- 32012 • The [EINVAL], [ENOMEM], [ENXIO], and [ETXTBSY] optional error conditions are added.

32013 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 32014 • The *freopen()* prototype is updated.
- 32015 • The DESCRIPTION is updated.

32016 The wording of the mandatory [ELOOP] error condition is updated, and a second optional  
32017 [ELOOP] error condition is added.

32018 The DESCRIPTION is updated regarding failure to close, changing the “file” to “file descriptor”.

32019 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/40 is applied, adding the following  
32020 sentence to the DESCRIPTION: “In this case, the file descriptor associated with the stream need  
32021 not be closed if the call to *freopen()* succeeds.”.32022 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/41 is applied, adding an mandatory  
32023 [EBADF] error, and an optional [EBADF] error to the ERRORS section.32024 **Issue 7**32025 Austin Group Interpretation 1003.1-2001 #043 is applied, clarifying that the *freopen()* function  
32026 allocates a file descriptor as per *open()*.

32027 Austin Group Interpretation 1003.1-2001 #143 is applied.

32028 Austin Group Interpretation 1003.1-2001 #159 is applied, clarifying requirements for the flags set  
32029 on the open file description.

- 32030 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.
- 32031 SD5-XSH-ERN-150 and SD5-XSH-ERN-219 are applied.
- 32032 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0181 [291,433], XSH/TC1-2008/0182
- 32033 [146,433], XSH/TC1-2008/0183 [324], and XSH/TC1-2008/0184 [14] are applied.

32034 **NAME**

32035 frexp, frexpf, frexpl — extract mantissa and exponent from a double precision number

32036 **SYNOPSIS**

```
32037 #include <math.h>
32038 double frexp(double num, int *exp);
32039 float frexpf(float num, int *exp);
32040 long double frexpl(long double num, int *exp);
```

32041 **DESCRIPTION**

32042 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 32043 conflict between the requirements described here and the ISO C standard is unintentional. This  
 32044 volume of POSIX.1-2008 defers to the ISO C standard.

32045 These functions shall break a floating-point number *num* into a normalized fraction and an  
 32046 integral power of 2. The integer exponent shall be stored in the **int** object pointed to by *exp*.

32047 **RETURN VALUE**

32048 For finite arguments, these functions shall return the value *x*, such that *x* has a magnitude in the  
 32049 interval  $[\frac{1}{2}, 1)$  or 0, and *num* equals *x* times 2 raised to the power *\*exp*.

32050 MX If *num* is NaN, a NaN shall be returned, and the value of *\*exp* is unspecified.

32051 If *num* is  $\pm 0$ ,  $\pm 0$  shall be returned, and the value of *\*exp* shall be 0.

32052 If *num* is  $\pm \text{Inf}$ , *num* shall be returned, and the value of *\*exp* is unspecified.

32053 **ERRORS**

32054 No errors are defined.

32055 **EXAMPLES**

32056 None.

32057 **APPLICATION USAGE**

32058 None.

32059 **RATIONALE**

32060 None.

32061 **FUTURE DIRECTIONS**

32062 None.

32063 **SEE ALSO**

32064 *isnan()*, *ldexp()*, *modf()*

32065 XBD **<math.h>**

32066 **CHANGE HISTORY**

32067 First released in Issue 1. Derived from Issue 1 of the SVID.

32068 **Issue 5**

32069 The DESCRIPTION is updated to indicate how an application should check for an error. This  
 32070 text was previously published in the APPLICATION USAGE section.

32071 **Issue 6**

32072 The *frexp()* and *frexpl()* functions are added for alignment with the ISO/IEC 9899:1999  
32073 standard.

32074 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
32075 revised to align with the ISO/IEC 9899:1999 standard.

32076 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
32077 marked.

32078 **NAME**

32079       fscanf, scanf, sscanf — convert formatted input

32080 **SYNOPSIS**

32081       #include &lt;stdio.h&gt;

32082       int fscanf(FILE \*restrict *stream*, const char \*restrict *format*, ...);32083       int scanf(const char \*restrict *format*, ...);32084       int sscanf(const char \*restrict *s*, const char \*restrict *format*, ...);32085 **DESCRIPTION**

32086 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 32087 conflict between the requirements described here and the ISO C standard is unintentional. This  
 32088 volume of POSIX.1-2008 defers to the ISO C standard.

32089       The *fscanf()* function shall read from the named input *stream*. The *scanf()* function shall read  
 32090 from the standard input stream *stdin*. The *sscanf()* function shall read from the string *s*. Each  
 32091 function reads bytes, interprets them according to a format, and stores the results in its  
 32092 arguments. Each expects, as arguments, a control string *format* described below, and a set of  
 32093 *pointer* arguments indicating where the converted input should be stored. The result is  
 32094 undefined if there are insufficient arguments for the format. If the format is exhausted while  
 32095 arguments remain, the excess arguments shall be evaluated but otherwise ignored.

32096 CX       Conversions can be applied to the *n*th argument after the *format* in the argument list, rather than  
 32097 to the next unused argument. In this case, the conversion specifier character % (see below) is  
 32098 replaced by the sequence "%n\$", where *n* is a decimal integer in the range [1,{NL\_ARGMAX}].  
 32099 This feature provides for the definition of format strings that select arguments in an order  
 32100 appropriate to specific languages. In format strings containing the "%n\$" form of conversion  
 32101 specifications, it is unspecified whether numbered arguments in the argument list can be  
 32102 referenced from the format string more than once.

32103       The *format* can contain either form of a conversion specification—that is, % or "%n\$"—but the  
 32104 two forms cannot be mixed within a single *format* string. The only exception to this is that %% or  
 32105 %\* can be mixed with the "%n\$" form. When numbered argument specifications are used,  
 32106 specifying the *N*th argument requires that all the leading arguments, from the first to the  
 32107 (*N*–1)th, are pointers.

32108       The *fscanf()* function in all its forms shall allow detection of a language-dependent radix  
 32109 character in the input string. The radix character is defined in the current locale (category  
 32110 LC\_NUMERIC). In the POSIX locale, or in a locale where the radix character is not defined, the  
 32111 radix character shall default to a <period> ('.').

32112       The format is a character string, beginning and ending in its initial shift state, if any, composed  
 32113 of zero or more directives. Each directive is composed of one of the following: one or more  
 32114 white-space characters (<space>, <tab>, <newline>, <vertical-tab>, or <form-feed>); an ordinary  
 32115 character (neither '%' nor a white-space character); or a conversion specification. Each  
 32116 CX       conversion specification is introduced by the character '%' or the character sequence "%n\$",  
 32117 after which the following appear in sequence:

- 32118       • An optional assignment-suppressing character '\*'.
- 32119       • An optional non-zero decimal integer that specifies the maximum field width.
- 32120 CX       • An optional assignment-allocation character 'm'.
- 32121       • An option length modifier that specifies the size of the receiving object.

- A *conversion specifier* character that specifies the type of conversion to be applied. The valid conversion specifiers are described below.

The *fscanf()* functions shall execute each directive of the format in turn. If a directive fails, as detailed below, the function shall return. Failures are described as input failures (due to the unavailability of input bytes) or matching failures (due to inappropriate input).

A directive composed of one or more white-space characters shall be executed by reading input until no more valid input can be read, or up to the first byte which is not a white-space character, which remains unread.

A directive that is an ordinary character shall be executed as follows: the next byte shall be read from the input and compared with the byte that comprises the directive; if the comparison shows that they are not equivalent, the directive shall fail, and the differing and subsequent bytes shall remain unread. Similarly, if end-of-file, an encoding error, or a read error prevents a character from being read, the directive shall fail.

A directive that is a conversion specification defines a set of matching input sequences, as described below for each conversion character. A conversion specification shall be executed in the following steps.

Input white-space characters (as specified by *isspace()*) shall be skipped, unless the conversion specification includes a *[*, *c*, *C*, or *n* conversion specifier.

An item shall be read from the input, unless the conversion specification includes an *n* conversion specifier. An input item shall be defined as the longest sequence of input bytes (up to any specified maximum field width, which may be measured in characters or bytes dependent on the conversion specifier) which is an initial subsequence of a matching sequence. The first byte, if any, after the input item shall remain unread. If the length of the input item is 0, the execution of the conversion specification shall fail; this condition is a matching failure, unless end-of-file, an encoding error, or a read error prevented input from the stream, in which case it is an input failure.

Except in the case of a *%* conversion specifier, the input item (or, in the case of a *%n* conversion specification, the count of input bytes) shall be converted to a type appropriate to the conversion character. If the input item is not a matching sequence, the execution of the conversion specification fails; this condition is a matching failure. Unless assignment suppression was indicated by a *'\*'*, the result of the conversion shall be placed in the object pointed to by the first argument following the *format* argument that has not already received a conversion result if the conversion specification is introduced by *%*, or in the *n*th argument if introduced by the character sequence *"%n\$"*. If this object does not have an appropriate type, or if the result of the conversion cannot be represented in the space provided, the behavior is undefined.

The *%c*, *%s*, and *%[* conversion specifiers shall accept an optional assignment-allocation character *'m'*, which shall cause a memory buffer to be allocated to hold the string converted including a terminating null character. In such a case, the argument corresponding to the conversion specifier should be a reference to a pointer variable that will receive a pointer to the allocated buffer. The system shall allocate a buffer as if *malloc()* had been called. The application shall be responsible for freeing the memory after usage. If there is insufficient memory to allocate a buffer, the function shall set *errno* to *[ENOMEM]* and a conversion error shall result. If the function returns EOF, any memory successfully allocated for parameters using assignment-allocation character *'m'* by this call shall be freed before the function returns.

|       |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------|----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 32166 | The length modifiers and their meanings are:                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32167 | hh                                                                                                                   | Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to <b>signed char</b> or <b>unsigned char</b> .                                                                                                                                                                                                                                                                                                                                                                                             |
| 32168 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32169 | h                                                                                                                    | Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to <b>short</b> or <b>unsigned short</b> .                                                                                                                                                                                                                                                                                                                                                                                                  |
| 32170 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32171 | l (ell)                                                                                                              | Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to <b>long</b> or <b>unsigned long</b> ; that a following a, A, e, E, f, F, g, or G conversion specifier applies to an argument with type pointer to <b>double</b> ; or that a following c, s, or [ conversion specifier applies to an argument with type pointer to <b>wchar_t</b> . If the 'm' assignment-allocation character is specified, the conversion applies to an argument with the type pointer to a pointer to <b>wchar_t</b> . |
| 32172 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32173 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32174 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32175 | CX                                                                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32176 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32177 | ll (ell-ell)                                                                                                         | Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to <b>long long</b> or <b>unsigned long long</b> .                                                                                                                                                                                                                                                                                                                                                                                          |
| 32178 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32179 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32180 | j                                                                                                                    | Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to <b>intmax_t</b> or <b>uintmax_t</b> .                                                                                                                                                                                                                                                                                                                                                                                                    |
| 32181 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32182 | z                                                                                                                    | Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to <b>size_t</b> or the corresponding signed integer type.                                                                                                                                                                                                                                                                                                                                                                                  |
| 32183 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32184 | t                                                                                                                    | Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to <b>ptrdiff_t</b> or the corresponding <b>unsigned</b> type.                                                                                                                                                                                                                                                                                                                                                                              |
| 32185 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32186 | L                                                                                                                    | Specifies that a following a, A, e, E, f, F, g, or G conversion specifier applies to an argument with type pointer to <b>long double</b> .                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 32187 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32188 | If a length modifier appears with any conversion specifier other than as specified above, the behavior is undefined. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32189 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32190 | The following conversion specifiers are valid:                                                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32191 | d                                                                                                                    | Matches an optionally signed decimal integer, whose format is the same as expected for the subject sequence of <i>strtol()</i> with the value 10 for the <i>base</i> argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to <b>int</b> .                                                                                                                                                                                                                                                 |
| 32192 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32193 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32194 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32195 | i                                                                                                                    | Matches an optionally signed integer, whose format is the same as expected for the subject sequence of <i>strtol()</i> with 0 for the <i>base</i> argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to <b>int</b> .                                                                                                                                                                                                                                                                    |
| 32196 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32197 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32198 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32199 | o                                                                                                                    | Matches an optionally signed octal integer, whose format is the same as expected for the subject sequence of <i>strtoul()</i> with the value 8 for the <i>base</i> argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to <b>unsigned</b> .                                                                                                                                                                                                                                              |
| 32200 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32201 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32202 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32203 | u                                                                                                                    | Matches an optionally signed decimal integer, whose format is the same as expected for the subject sequence of <i>strtoul()</i> with the value 10 for the <i>base</i> argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to <b>unsigned</b> .                                                                                                                                                                                                                                           |
| 32204 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32205 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32206 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32207 | x                                                                                                                    | Matches an optionally signed hexadecimal integer, whose format is the same as expected for the subject sequence of <i>strtoul()</i> with the value 16 for the <i>base</i> argument. In the absence of a size modifier, the application shall ensure that the corresponding                                                                                                                                                                                                                                                                                  |
| 32208 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 32209 |                                                                                                                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |



argument is a pointer to **unsigned**.

a, e, f, g

Matches an optionally signed floating-point number, infinity, or NaN, whose format is the same as expected for the subject sequence of *strtod*( ). In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to **float**.

If the *fprintf*( ) family of functions generates character string representations for infinity and NaN (a symbolic entity encoded in floating-point format) to support IEEE Std 754-1985, the *fscanf*( ) family of functions shall recognize them as input.

s

Matches a sequence of bytes that are not white-space characters. If the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to the initial byte of an array of **char**, **signed char**, or **unsigned char** large enough to accept the sequence and a terminating null character code, which shall be added automatically. Otherwise, the application shall ensure that the corresponding argument is a pointer to a pointer to a **char**.

If an l (ell) qualifier is present, the input is a sequence of characters that begins in the initial shift state. Each character shall be converted to a wide character as if by a call to the *mbrtowc*( ) function, with the conversion state described by an **mbstate\_t** object initialized to zero before the first character is converted. If the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to an array of **wchar\_t** large enough to accept the sequence and the terminating null wide character, which shall be added automatically. Otherwise, the application shall ensure that the corresponding argument is a pointer to a pointer to a **wchar\_t**.

[

Matches a non-empty sequence of bytes from a set of expected bytes (the *scanset*). The normal skip over white-space characters shall be suppressed in this case. If the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to the initial byte of an array of **char**, **signed char**, or **unsigned char** large enough to accept the sequence and a terminating null byte, which shall be added automatically. Otherwise, the application shall ensure that the corresponding argument is a pointer to a pointer to a **char**.

If an l (ell) qualifier is present, the input is a sequence of characters that begins in the initial shift state. Each character in the sequence shall be converted to a wide character as if by a call to the *mbrtowc*( ) function, with the conversion state described by an **mbstate\_t** object initialized to zero before the first character is converted. If the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to an array of **wchar\_t** large enough to accept the sequence and the terminating null wide character, which shall be added automatically. Otherwise, the application shall ensure that the corresponding argument is a pointer to a pointer to a **wchar\_t**.

The conversion specification includes all subsequent bytes in the *format* string up to and including the matching <right-square-bracket> (']'). The bytes between the square brackets (the *scanlist*) comprise the *scanset*, unless the byte after the <left-square-bracket> is a <circumflex> ('^'), in which case the *scanset* contains all bytes that do not appear in the *scanlist* between the <circumflex> and the <right-square-bracket>. If the conversion specification begins with "[ ]" or "[ ^ ]", the <right-square-bracket> is included in the *scanlist* and the next <right-square-bracket> is the matching <right-square-bracket> that ends the conversion specification; otherwise, the



|       |     |                                                                                                           |
|-------|-----|-----------------------------------------------------------------------------------------------------------|
| 32258 |     | first <right-square-bracket> is the one that ends the conversion specification. If a '-' is               |
| 32259 |     | in the scanlist and is not the first character, nor the second where the first character is a             |
| 32260 |     | '^', nor the last character, the behavior is implementation-defined.                                      |
| 32261 | c   | Matches a sequence of bytes of the number specified by the field width (1 if no field                     |
| 32262 |     | width is present in the conversion specification). No null byte is added. The normal                      |
| 32263 |     | skip over white-space characters shall be suppressed in this case. If the 'm'                             |
| 32264 |     | assignment-allocation character is not specified, the application shall ensure that the                   |
| 32265 |     | corresponding argument is a pointer to the initial byte of an array of <b>char</b> , <b>signed char</b> , |
| 32266 | CX  | or <b>unsigned char</b> large enough to accept the sequence. Otherwise, the application                   |
| 32267 |     | shall ensure that the corresponding argument is a pointer to a pointer to a <b>char</b> .                 |
| 32268 |     | If an l (ell) qualifier is present, the input shall be a sequence of characters that begins in            |
| 32269 |     | the initial shift state. Each character in the sequence is converted to a wide character as               |
| 32270 |     | if by a call to the <i>mbrtowc()</i> function, with the conversion state described by an                  |
| 32271 |     | <b>mbstate_t</b> object initialized to zero before the first character is converted. No null wide         |
| 32272 |     | character is added. If the 'm' assignment-allocation character is not specified, the                      |
| 32273 |     | application shall ensure that the corresponding argument is a pointer to an array of                      |
| 32274 | CX  | <b>wchar_t</b> large enough to accept the resulting sequence of wide characters. Otherwise,               |
| 32275 |     | the application shall ensure that the corresponding argument is a pointer to a pointer to                 |
| 32276 |     | a <b>wchar_t</b> .                                                                                        |
| 32277 | p   | Matches an implementation-defined set of sequences, which shall be the same as the set                    |
| 32278 |     | of sequences that is produced by the %p conversion specification of the corresponding                     |
| 32279 |     | <i>fprintf()</i> functions. The application shall ensure that the corresponding argument is a             |
| 32280 |     | pointer to a pointer to <b>void</b> . The interpretation of the input item is implementation-             |
| 32281 |     | defined. If the input item is a value converted earlier during the same program                           |
| 32282 |     | execution, the pointer that results shall compare equal to that value; otherwise, the                     |
| 32283 |     | behavior of the %p conversion specification is undefined.                                                 |
| 32284 | n   | No input is consumed. The application shall ensure that the corresponding argument is                     |
| 32285 |     | a pointer to the integer into which shall be written the number of bytes read from the                    |
| 32286 |     | input so far by this call to the <i>fscanf()</i> functions. Execution of a %n conversion                  |
| 32287 |     | specification shall not increment the assignment count returned at the completion of                      |
| 32288 |     | execution of the function. No argument shall be converted, but one shall be consumed.                     |
| 32289 |     | If the conversion specification includes an assignment-suppressing character or a field                   |
| 32290 |     | width, the behavior is undefined.                                                                         |
| 32291 | XSI | <b>C</b> Equivalent to <b>lc</b> .                                                                        |
| 32292 | XSI | <b>S</b> Equivalent to <b>ls</b> .                                                                        |
| 32293 | %   | Matches a single '%' character; no conversion or assignment occurs. The complete                          |
| 32294 |     | conversion specification shall be %%.                                                                     |
| 32295 |     | If a conversion specification is invalid, the behavior is undefined.                                      |
| 32296 |     | The conversion specifiers A, E, F, G, and X are also valid and shall be equivalent to a, e, f, g, and     |
| 32297 |     | x, respectively.                                                                                          |
| 32298 |     | If end-of-file is encountered during input, conversion shall be terminated. If end-of-file occurs         |
| 32299 |     | before any bytes matching the current conversion specification (except for %n) have been read             |
| 32300 |     | (other than leading white-space characters, where permitted), execution of the current                    |
| 32301 |     | conversion specification shall terminate with an input failure. Otherwise, unless execution of the        |
| 32302 |     | current conversion specification is terminated with a matching failure, execution of the                  |
| 32303 |     | following conversion specification (if any) shall be terminated with an input failure.                    |

Reaching the end of the string in *sscanf()* shall be equivalent to encountering end-of-file for *fscanf()*.

If conversion terminates on a conflicting input, the offending input is left unread in the input. Any trailing white space (including <newline> characters) shall be left unread unless matched by a conversion specification. The success of literal matches and suppressed assignments is only directly determinable via the %n conversion specification.

CX The *fscanf()* and *scanf()* functions may mark the last data access timestamp of the file associated with *stream* for update. The last data access timestamp shall be marked for update by the first successful execution of *fgetc()*, *fgets()*, *fread()*, *getc()*, *getchar()*, *getdelim()*, *getline()*, *gets()*, *fscanf()*, or *scanf()* using *stream* that returns data not supplied by a prior call to *ungetc()*.

#### RETURN VALUE

Upon successful completion, these functions shall return the number of successfully matched and assigned input items; this number can be zero in the event of an early matching failure. If the input ends before the first conversion (if any) has completed, and without a matching failure having occurred, EOF shall be returned. If an error occurs before the first conversion (if any) has completed, and without a matching failure having occurred, EOF shall be returned and *errno* shall be set to indicate the error. If a read error occurs, the error indicator for the stream shall be set.

#### ERRORS

For the conditions under which the *fscanf()* functions fail and may fail, refer to *fgetc()* or *fgetwc()*.

In addition, the *fscanf()* function shall fail if:

CX [EILSEQ] Input byte sequence does not form a valid character.

[ENOMEM] Insufficient storage space is available.

In addition, the *fscanf()* function may fail if:

CX [EINVAL] There are insufficient arguments.

#### EXAMPLES

The call:

```
int i, n; float x; char name[50];
n = scanf("%d%f%s", &i, &x, name);
```

with the input line:

```
25 54.32E-1 Hamster
```

assigns to *n* the value 3, to *i* the value 25, to *x* the value 5.432, and *name* contains the string "Hamster".

The call:

```
int i; float x; char name[50];
(void) scanf("%2d%f%d %[0123456789]", &i, &x, name);
```

with input:

```
56789 0123 56a72
```

assigns 56 to *i*, 789.0 to *x*, skips 0123, and places the string "56\0" in *name*. The next call to *getchar()* shall return the character 'a'.

**Reading Data into an Array**

The following call uses *fscanf()* to read three floating-point numbers from standard input into the *input* array.

```
float input[3]; fscanf (stdin, "%f %f %f", input, input+1, input+2);
```

**APPLICATION USAGE**

If the application calling *fscanf()* has any objects of type **wint\_t** or **wchar\_t**, it must also include the **<wchar.h>** header to have these objects defined.

For functions that allocate memory as if by *malloc()*, the application should release such memory when it is no longer required by a call to *free()*. For *fscanf()*, this is memory allocated via use of the 'm' assignment-allocation character.

**RATIONALE**

This function is aligned with the ISO/IEC 9899:1999 standard, and in doing so a few “obvious” things were not included. Specifically, the set of characters allowed in a scanset is limited to single-byte characters. In other similar places, multi-byte characters have been permitted, but for alignment with the ISO/IEC 9899:1999 standard, it has not been done here. Applications needing this could use the corresponding wide-character functions to achieve the desired results.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 2.5](#) (on page 495), *fprintf()*, *getc()*, *setlocale()*, *strtod()*, *strtol()*, *strtoul()*, *wcrtomb()*

[XBD Chapter 7](#) (on page 135), **<inttypes.h>**, **<langinfo.h>**, **<stdio.h>**, **<wchar.h>**

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

Aligned with ISO/IEC 9899:1990/Amendment 1:1995 (E). Specifically, the *l* (ell) qualifier is now defined for the *c*, *s*, and *[* conversion specifiers.

The DESCRIPTION is updated to indicate that if infinity and NaN can be generated by the *fprintf()* family of functions, then they are recognized by the *fscanf()* family.

**Issue 6**

The Open Group Corrigenda U021/7 and U028/10 are applied. These correct several occurrences of “characters” in the text which have been replaced with the term “bytes”.

The normative text is updated to avoid use of the term “must” for application requirements.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The prototypes for *fscanf()*, *scanf()*, and *sscanf()* are updated.
- The DESCRIPTION is updated.
- The *hh*, *ll*, *j*, *t*, and *z* length modifiers are added.
- The *a*, *A*, and *F* conversion characters are added.

The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion specification” consistently.

32385 **Issue 7**

32386 Austin Group Interpretation 1003.1-2001 #170 is applied.

32387 SD5-XSH-ERN-9 is applied, correcting *fscanf()* to *scanf()* in the DESCRIPTION.

32388 SD5-XSH-ERN-132 is applied, adding the assignment-allocation character 'm'.

32389 Functionality relating to the %n\$ form of conversion specification is moved from the XSI option  
32390 to the Base.

32391 Changes are made related to support for finegrained timestamps.

32392 The APPLICATION USAGE section is updated to clarify that memory is allocated as if by  
32393 *malloc()*.

32394 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0185 [302], XSH/TC1-2008/0186 [90],  
32395 and XSH/TC1-2008/0187 [14] are applied. XSH/TC1-2008/0186 [90] changes the second  
32396 sentence in the RETURN VALUE section to align with expected wording changes in the next  
32397 revision of the ISO C standard.

32398 **NAME**

32399       fseek, fseeko — reposition a file-position indicator in a stream

32400 **SYNOPSIS**

32401       #include &lt;stdio.h&gt;

32402       int fseek(FILE \*stream, long offset, int whence);

32403 CX      int fseeko(FILE \*stream, off\_t offset, int whence);

32404 **DESCRIPTION**

32405 CX      The functionality described on this reference page is aligned with the ISO C standard. Any  
 32406 conflict between the requirements described here and the ISO C standard is unintentional. This  
 32407 volume of POSIX.1-2008 defers to the ISO C standard.

32408      The *fseek()* function shall set the file-position indicator for the stream pointed to by *stream*. If a  
 32409 read or write error occurs, the error indicator for the stream shall be set and *fseek()* fails.

32410      The new position, measured in bytes from the beginning of the file, shall be obtained by adding  
 32411 *offset* to the position specified by *whence*. The specified point is the beginning of the file for  
 32412 SEEK\_SET, the current value of the file-position indicator for SEEK\_CUR, or end-of-file for  
 32413 SEEK\_END.

32414      If the stream is to be used with wide-character input/output functions, the application shall  
 32415 ensure that *offset* is either 0 or a value returned by an earlier call to *ftell()* on the same stream and  
 32416 *whence* is SEEK\_SET.

32417      A successful call to *fseek()* shall clear the end-of-file indicator for the stream and undo any effects  
 32418 of *ungetc()* and *ungetwc()* on the same stream. After an *fseek()* call, the next operation on an  
 32419 update stream may be either input or output.

32420 CX      If the most recent operation, other than *ftell()*, on a given stream is *fflush()*, the file offset in the  
 32421 underlying open file description shall be adjusted to reflect the location specified by *fseek()*.

32422      The *fseek()* function shall allow the file-position indicator to be set beyond the end of existing  
 32423 data in the file. If data is later written at this point, subsequent reads of data in the gap shall  
 32424 return bytes with the value 0 until data is actually written into the gap.

32425      The behavior of *fseek()* on devices which are incapable of seeking is implementation-defined.  
 32426 The value of the file offset associated with such a device is undefined.

32427      If the stream is writable and buffered data had not been written to the underlying file, *fseek()*  
 32428 shall cause the unwritten data to be written to the file and shall mark the last data modification  
 32429 and last file status change timestamps of the file for update.

32430      In a locale with state-dependent encoding, whether *fseek()* restores the stream's shift state is  
 32431 implementation-defined.

32432      The *fseeko()* function shall be equivalent to the *fseek()* function except that the *offset* argument is  
 32433 of type **off\_t**.

32434 **RETURN VALUE**32435 CX      The *fseek()* and *fseeko()* functions shall return 0 if they succeed.32436 CX      Otherwise, they shall return -1 and set *errno* to indicate the error.32437 **ERRORS**

32438 CX      The *fseek()* and *fseeko()* functions shall fail if, either the *stream* is unbuffered or the *stream's*  
 32439 buffer needed to be flushed, and the call to *fseek()* or *fseeko()* causes an underlying *lseek()* or  
 32440 *write()* to be invoked, and:

|       |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |
|-------|-----|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 32441 | CX  | [EAGAIN]    | The O_NONBLOCK flag is set for the file descriptor and the thread would be delayed in the write operation.                                                                                                                                                                                                                                                                                 |
| 32442 |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |
| 32443 | CX  | [EBADF]     | The file descriptor underlying the stream file is not open for writing or the stream's buffer needed to be flushed and the file is not open.                                                                                                                                                                                                                                               |
| 32444 |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |
| 32445 | CX  | [EFBIG]     | An attempt was made to write a file that exceeds the maximum file size.                                                                                                                                                                                                                                                                                                                    |
| 32446 | XSI | [EFBIG]     | An attempt was made to write a file that exceeds the file size limit of the process.                                                                                                                                                                                                                                                                                                       |
| 32447 |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |
| 32448 | CX  | [EFBIG]     | The file is a regular file and an attempt was made to write at or beyond the offset maximum associated with the corresponding stream.                                                                                                                                                                                                                                                      |
| 32449 |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |
| 32450 | CX  | [EINTR]     | The write operation was terminated due to the receipt of a signal, and no data was transferred.                                                                                                                                                                                                                                                                                            |
| 32451 |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |
| 32452 | CX  | [EINVAL]    | The <i>whence</i> argument is invalid. The resulting file-position indicator would be set to a negative value.                                                                                                                                                                                                                                                                             |
| 32453 |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |
| 32454 | CX  | [EIO]       | A physical I/O error has occurred, or the process is a member of a background process group attempting to perform a <i>write()</i> to its controlling terminal, TOSTOP is set, the calling thread is not blocking SIGTTOU, the process is not ignoring SIGTTOU, and the process group of the process is orphaned. This error may also be returned under implementation-defined conditions. |
| 32455 |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |
| 32456 |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |
| 32457 |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |
| 32458 |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |
| 32459 | CX  | [ENOSPC]    | There was no free space remaining on the device containing the file.                                                                                                                                                                                                                                                                                                                       |
| 32460 | CX  | [EOVERFLOW] | For <i>fseek()</i> , the resulting file offset would be a value which cannot be represented correctly in an object of type <b>long</b> .                                                                                                                                                                                                                                                   |
| 32461 |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |
| 32462 | CX  | [EOVERFLOW] | For <i>fseeko()</i> , the resulting file offset would be a value which cannot be represented correctly in an object of type <b>off_t</b> .                                                                                                                                                                                                                                                 |
| 32463 |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |
| 32464 | CX  | [EPIPE]     | An attempt was made to write to a pipe or FIFO that is not open for reading by any process; a SIGPIPE signal shall also be sent to the thread.                                                                                                                                                                                                                                             |
| 32465 |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |
| 32466 | CX  | [ESPIPE]    | The file descriptor underlying <i>stream</i> is associated with a pipe, FIFO, or socket.                                                                                                                                                                                                                                                                                                   |
| 32467 | CX  |             | The <i>fseek()</i> and <i>fseeko()</i> functions may fail if:                                                                                                                                                                                                                                                                                                                              |
| 32468 | CX  | [ENXIO]     | A request was made of a nonexistent device, or the request was outside the capabilities of the device.                                                                                                                                                                                                                                                                                     |
| 32469 |     |             |                                                                                                                                                                                                                                                                                                                                                                                            |

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

Section 2.5 (on page 495), *fopen()*, *fsetpos()*, *ftell()*, *getrlimit()*, *lseek()*, *rewind()*, *ulimit()*, *ungetc()*, *write()*

XBD &lt;stdio.h&gt;

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

Normative text previously in the APPLICATION USAGE section is moved to the DESCRIPTION.

Large File Summit extensions are added.

**Issue 6**

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The *fseeko()* function is added.
- The [EFBIG], [EOVERFLOW], and [ENXIO] mandatory error conditions are added.

The following change is incorporated for alignment with the FIPS requirements:

- The [EINTR] error is no longer an indication that the implementation does not report partial transfers.

The normative text is updated to avoid use of the term “must” for application requirements.

The DESCRIPTION is updated to explicitly state that *fseek()* sets the file-position indicator, and then on error the error indicator is set and *fseek()* fails. This is for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/42 is applied, updating the [EAGAIN] error in the ERRORS section from “the process would be delayed” to “the thread would be delayed”.

**Issue 7**

Changes are made related to support for finegrained timestamps.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0188 [79], XSH/TC1-2008/0189 [122], XSH/TC1-2008/0190 [225], and XSH/TC1-2008/0191 [14] are applied.



32508 **NAME**

32509 fsetpos — set current file position

32510 **SYNOPSIS**

32511 #include &lt;stdio.h&gt;

32512 int fsetpos(FILE \*stream, const fpos\_t \*pos);

32513 **DESCRIPTION**

32514 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 32515 conflict between the requirements described here and the ISO C standard is unintentional. This  
 32516 volume of POSIX.1-2008 defers to the ISO C standard.

32517 The *fsetpos()* function shall set the file position and state indicators for the stream pointed to by  
 32518 *stream* according to the value of the object pointed to by *pos*, which the application shall ensure is  
 32519 a value obtained from an earlier call to *fgetpos()* on the same stream. If a read or write error  
 32520 occurs, the error indicator for the stream shall be set and *fsetpos()* fails.

32521 A successful call to the *fsetpos()* function shall clear the end-of-file indicator for the stream and  
 32522 undo any effects of *ungetc()* on the same stream. After an *fsetpos()* call, the next operation on an  
 32523 update stream may be either input or output.

32524 CX The behavior of *fsetpos()* on devices which are incapable of seeking is implementation-defined.  
 32525 The value of the file offset associated with such a device is undefined.

32526 The *fsetpos()* function shall not change the setting of *errno* if successful.

32527 **RETURN VALUE**

32528 The *fsetpos()* function shall return 0 if it succeeds; otherwise, it shall return a non-zero value and  
 32529 set *errno* to indicate the error.

32530 **ERRORS**

32531 CX The *fsetpos()* function shall fail if, either the *stream* is unbuffered or the *stream*'s buffer needed to  
 32532 be flushed, and the call to *fsetpos()* causes an underlying *lseek()* or *write()* to be invoked, and:

32533 CX [EAGAIN] The O\_NONBLOCK flag is set for the file descriptor and the thread would be  
 32534 delayed in the write operation.

32535 CX [EBADF] The file descriptor underlying the stream file is not open for writing or the  
 32536 stream's buffer needed to be flushed and the file is not open.

32537 CX [EFBIG] An attempt was made to write a file that exceeds the maximum file size.

32538 XSI [EFBIG] An attempt was made to write a file that exceeds the file size limit of the  
 32539 process.

32540 CX [EFBIG] The file is a regular file and an attempt was made to write at or beyond the  
 32541 offset maximum associated with the corresponding stream.

32542 CX [EINTR] The write operation was terminated due to the receipt of a signal, and no data  
 32543 was transferred.

32544 CX [EIO] A physical I/O error has occurred, or the process is a member of a background  
 32545 process group attempting to perform a *write()* to its controlling terminal,  
 32546 TOSTOP is set, the calling thread is not blocking SIGTTOU, the process is not  
 32547 ignoring SIGTTOU, and the process group of the process is orphaned. This  
 32548 error may also be returned under implementation-defined conditions.

32549 CX [ENOSPC] There was no free space remaining on the device containing the file.



32550 CX [EPIPE] An attempt was made to write to a pipe or FIFO that is not open for reading  
 32551 by any process; a SIGPIPE signal shall also be sent to the thread.

32552 CX [ESPIPE] The file descriptor underlying *stream* is associated with a pipe, FIFO, or socket.

32553 The *fsetpos()* function may fail if:

32554 CX [ENXIO] A request was made of a nonexistent device, or the request was outside the  
 32555 capabilities of the device.

#### 32556 EXAMPLES

32557 None.

#### 32558 APPLICATION USAGE

32559 None.

#### 32560 RATIONALE

32561 None.

#### 32562 FUTURE DIRECTIONS

32563 None.

#### 32564 SEE ALSO

32565 [Section 2.5](#) (on page 495), *fopen()*, *ftell()*, *lseek()*, *rewind()*, *ungetc()*, *write()*

32566 XBD [<stdio.h>](#)

#### 32567 CHANGE HISTORY

32568 First released in Issue 4. Derived from the ISO C standard.

#### 32569 Issue 6

32570 Extensions beyond the ISO C standard are marked.

32571 The normative text is updated to avoid use of the term “must” for application requirements.

32572 The DESCRIPTION is updated to clarify that the error indicator is set for the stream on a read or  
 32573 write error. This is for alignment with the ISO/IEC 9899:1999 standard.

32574 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/21 is applied, deleting an erroneous  
 32575 [EINVAL] error case from the ERRORS section.

32576 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/43 is applied, updating the [EAGAIN]  
 32577 error in the ERRORS section from “the process would be delayed” to “the thread would be  
 32578 delayed”.

#### 32579 Issue 7

32580 SD5-XSH-ERN-220 is applied.

32581 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0192 [105], XSH/TC1-2008/0193 [79],  
 32582 XSH/TC1-2008/0194 [225], XSH/TC1-2008/0195 [450], XSH/TC1-2008/0196 [450], and  
 32583 XSH/TC1-2008/0197 [14] are applied.

32584 **NAME**

32585       fstat — get file status

32586 **SYNOPSIS**

32587       #include &lt;sys/stat.h&gt;

32588       int fstat(int *fildes*, struct stat \**buf*);32589 **DESCRIPTION**32590       The *fstat()* function shall obtain information about an open file associated with the file  
32591       descriptor *fildes*, and shall write it to the area pointed to by *buf*.32592 SHM     If *fildes* references a shared memory object, the implementation shall update in the **stat** structure  
32593       pointed to by the *buf* argument the *st\_uid*, *st\_gid*, *st\_size*, and *st\_mode* fields, and only the  
32594       S\_IRUSR, S\_IWUSR, S\_IRGRP, S\_IWGRP, S\_IROTH, and S\_IWOTH file permission bits need be  
32595       valid. The implementation may update other fields and flags.32596 TYM     If *fildes* references a typed memory object, the implementation shall update in the **stat** structure  
32597       pointed to by the *buf* argument the *st\_uid*, *st\_gid*, *st\_size*, and *st\_mode* fields, and only the  
32598       S\_IRUSR, S\_IWUSR, S\_IRGRP, S\_IWGRP, S\_IROTH, and S\_IWOTH file permission bits need be  
32599       valid. The implementation may update other fields and flags.32600       The *buf* argument is a pointer to a **stat** structure, as defined in <sys/stat.h>, into which  
32601       information is placed concerning the file.32602       For all other file types defined in this volume of POSIX.1-2008, the structure members *st\_mode*,  
32603       *st\_ino*, *st\_dev*, *st\_uid*, *st\_gid*, *st\_atim*, *st\_ctim*, and *st\_mtim* shall have meaningful values and the  
32604       value of the *st\_nlink* member shall be set to the number of links to the file.32605       An implementation that provides additional or alternative file access control mechanisms may,  
32606       under implementation-defined conditions, cause *fstat()* to fail.32607       The *fstat()* function shall update any time-related fields (as described in XBD [Section 4.9](#), on  
32608       page 109), before writing into the **stat** structure.32609 **RETURN VALUE**32610       Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to  
32611       indicate the error.32612 **ERRORS**32613       The *fstat()* function shall fail if:32614       [EBADF]       The *fildes* argument is not a valid file descriptor.

32615       [EIO]         An I/O error occurred while reading from the file system.

32616       [EOVERFLOW]   The file size in bytes or the number of blocks allocated to the file or the file  
32617       serial number cannot be represented correctly in the structure pointed to by  
32618       *buf*.32619       The *fstat()* function may fail if:32620       [EOVERFLOW]   One of the values is too large to store into the structure pointed to by the *buf*  
32621       argument.

32622 **EXAMPLES**32623 **Obtaining File Status Information**

32624 The following example shows how to obtain file status information for a file named  
32625 **/home/cnd/mod1**. The structure variable *buffer* is defined for the **stat** structure. The  
32626 **/home/cnd/mod1** file is opened with read/write privileges and is passed to the open file  
32627 descriptor *fildev*.

```
32628 #include <sys/types.h>
32629 #include <sys/stat.h>
32630 #include <fcntl.h>

32631 struct stat buffer;
32632 int      status;
32633 ...
32634 fildev = open("/home/cnd/mod1", O_RDWR);
32635 status = fstat(fildev, &buffer);
```

32636 **APPLICATION USAGE**

32637 None.

32638 **RATIONALE**

32639 None.

32640 **FUTURE DIRECTIONS**

32641 None.

32642 **SEE ALSO**

32643 [\*fstatat\(\)\*](#)

32644 XBD [Section 4.9](#) (on page 109), [\*<sys/stat.h>\*](#), [\*<sys/types.h>\*](#)

32645 **CHANGE HISTORY**

32646 First released in Issue 1. Derived from Issue 1 of the SVID.

32647 **Issue 5**

32648 The DESCRIPTION is updated for alignment with the POSIX Realtime Extension.

32649 Large File Summit extensions are added.

32650 **Issue 6**

32651 In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

32652 The following new requirements on POSIX implementations derive from alignment with the  
32653 Single UNIX Specification:

- 32654 • The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was  
32655 required for conforming implementations of previous POSIX specifications, it was not  
32656 required for UNIX applications.
- 32657 • The [EIO] mandatory error condition is added.
- 32658 • The [EOVERFLOW] mandatory error condition is added. This change is to support large  
32659 files.
- 32660 • The [EOVERFLOW] optional error condition is added.

32661 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that  
32662 shared memory object semantics apply to typed memory objects.

32663 **Issue 7**

32664 XSH-SD5-ERN-161 is applied, updating the DESCRIPTION to clarify to which file types *st\_nlink*  
32665 applies.

32666 Changes are made related to support for finegrained timestamps.

**NAME**

fstatat, lstat, stat — get file status

**SYNOPSIS**

```
OH    #include <fcntl.h>
      #include <sys/stat.h>

      int fstatat(int fd, const char *restrict path,
                  struct stat *restrict buf, int flag);
      int lstat(const char *restrict path, struct stat *restrict buf);
      int stat(const char *restrict path, struct stat *restrict buf);
```

**DESCRIPTION**

The *stat()* function shall obtain information about the named file and write it to the area pointed to by the *buf* argument. The *path* argument points to a pathname naming a file. Read, write, or execute permission of the named file is not required. An implementation that provides additional or alternate file access control mechanisms may, under implementation-defined conditions, cause *stat()* to fail. In particular, the system may deny the existence of the file specified by *path*.

If the named file is a symbolic link, the *stat()* function shall continue pathname resolution using the contents of the symbolic link, and shall return information pertaining to the resulting file if the file exists.

The *buf* argument is a pointer to a **stat** structure, as defined in the **<sys/stat.h>** header, into which information is placed concerning the file.

The *stat()* function shall update any time-related fields (as described in XBD [Section 4.9](#), on page 109), before writing into the **stat** structure.

**SHM** If the named file is a shared memory object, the implementation shall update in the **stat** structure pointed to by the *buf* argument the *st\_uid*, *st\_gid*, *st\_size*, and *st\_mode* fields, and only the *S\_IRUSR*, *S\_IWUSR*, *S\_IRGRP*, *S\_IWGRP*, *S\_IROTH*, and *S\_IWOTH* file permission bits need be valid. The implementation may update other fields and flags.

**TYM** If the named file is a typed memory object, the implementation shall update in the **stat** structure pointed to by the *buf* argument the *st\_uid*, *st\_gid*, *st\_size*, and *st\_mode* fields, and only the *S\_IRUSR*, *S\_IWUSR*, *S\_IRGRP*, *S\_IWGRP*, *S\_IROTH*, and *S\_IWOTH* file permission bits need be valid. The implementation may update other fields and flags.

For all other file types defined in this volume of POSIX.1-2008, the structure members *st\_mode*, *st\_ino*, *st\_dev*, *st\_uid*, *st\_gid*, *st\_atim*, *st\_ctim*, and *st\_mtim* shall have meaningful values and the value of the member *st\_nlink* shall be set to the number of links to the file.

The *lstat()* function shall be equivalent to *stat()*, except when *path* refers to a symbolic link. In that case *lstat()* shall return information about the link, while *stat()* shall return information about the file the link references.

For symbolic links, the *st\_mode* member shall contain meaningful information when used with the file type macros. The file mode bits in *st\_mode* are unspecified. The structure members *st\_ino*, *st\_dev*, *st\_uid*, *st\_gid*, *st\_atim*, *st\_ctim*, and *st\_mtim* shall have meaningful values and the value of the *st\_nlink* member shall be set to the number of (hard) links to the symbolic link. The value of the *st\_size* member shall be set to the length of the pathname contained in the symbolic link not including any terminating null byte.

The *fstatat()* function shall be equivalent to the *stat()* or *lstat()* function, depending on the value of *flag* (see below), except in the case where *path* specifies a relative path. In this case the status shall be retrieved from a file relative to the directory associated with the file descriptor *fd* instead

32713 of the current working directory. If the access mode of the open file description associated with  
 32714 the file descriptor is not O\_SEARCH, the function shall check whether directory searches are  
 32715 permitted using the current permissions of the directory underlying the file descriptor. If the  
 32716 access mode is O\_SEARCH, the function shall not perform the check.

32717 Values for *flag* are constructed by a bitwise-inclusive OR of flags from the following list, defined  
 32718 in **<fcntl.h>**:

32719 AT\_SYMLINK\_NOFOLLOW

32720 If *path* names a symbolic link, the status of the symbolic link is returned.

32721 If *fstatat()* is passed the special value AT\_FDCWD in the *fd* parameter, the current working  
 32722 directory shall be used and the behavior shall be identical to a call to *stat()* or *lstat()* respectively,  
 32723 depending on whether or not the AT\_SYMLINK\_NOFOLLOW bit is set in *flag*.

## 32724 RETURN VALUE

32725 Upon successful completion, these functions shall return 0. Otherwise, these functions shall  
 32726 return -1 and set *errno* to indicate the error.

## 32727 ERRORS

32728 These functions shall fail if:

- |       |                |                                                                                                       |
|-------|----------------|-------------------------------------------------------------------------------------------------------|
| 32729 | [EACCES]       | Search permission is denied for a component of the path prefix.                                       |
| 32730 | [EIO]          | An error occurred while reading from the file system.                                                 |
| 32731 | [ELOOP]        | A loop exists in symbolic links encountered during resolution of the <i>path</i>                      |
| 32732 |                | argument.                                                                                             |
| 32733 | [ENAMETOOLONG] |                                                                                                       |
| 32734 |                | The length of a component of a pathname is longer than {NAME_MAX}.                                    |
| 32735 | [ENOENT]       | A component of <i>path</i> does not name an existing file or <i>path</i> is an empty string.          |
| 32736 | [ENOTDIR]      | A component of the path prefix names an existing file that is neither a                               |
| 32737 |                | directory nor a symbolic link to a directory, or the <i>path</i> argument contains at                 |
| 32738 |                | least one non- <i>&lt;slash&gt;</i> character and ends with one or more trailing <i>&lt;slash&gt;</i> |
| 32739 |                | characters and the last pathname component names an existing file that is                             |
| 32740 |                | neither a directory nor a symbolic link to a directory.                                               |
| 32741 | [EOVERFLOW]    | The file size in bytes or the number of blocks allocated to the file or the file                      |
| 32742 |                | serial number cannot be represented correctly in the structure pointed to by                          |
| 32743 |                | <i>buf</i> .                                                                                          |

32744 The *fstatat()* function shall fail if:

- |       |           |                                                                                                |
|-------|-----------|------------------------------------------------------------------------------------------------|
| 32745 | [EACCES]  | The access mode of the open file description associated with <i>fd</i> is not                  |
| 32746 |           | O_SEARCH and the permissions of the directory underlying <i>fd</i> do not permit               |
| 32747 |           | directory searches.                                                                            |
| 32748 | [EBADF]   | The <i>path</i> argument does not specify an absolute path and the <i>fd</i> argument is       |
| 32749 |           | neither AT_FDCWD nor a valid file descriptor open for reading or searching.                    |
| 32750 | [ENOTDIR] | The <i>path</i> argument is not an absolute path and <i>fd</i> is a file descriptor associated |
| 32751 |           | with a non-directory file.                                                                     |

32752 These functions may fail if:

- |       |         |                                                                |
|-------|---------|----------------------------------------------------------------|
| 32753 | [ELOOP] | More than {SYMLOOP_MAX} symbolic links were encountered during |
| 32754 |         | resolution of the <i>path</i> argument.                        |

32755 [ENAMETOOLONG]  
 32756       The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
 32757       symbolic link produced an intermediate result with a length that exceeds  
 32758       {PATH\_MAX}.

32759 [EOVERFLOW]   A value to be stored would overflow one of the members of the **stat** structure.

32760   The *fstatat()* function may fail if:

32761 [EINVAL]       The value of the *flag* argument is not valid.

## 32762 EXAMPLES

### 32763 Obtaining File Status Information

32764   The following example shows how to obtain file status information for a file named  
 32765   /home/cnd/mod1. The structure variable *buffer* is defined for the **stat** structure.

```
32766 #include <sys/types.h>
32767 #include <sys/stat.h>
32768 #include <fcntl.h>

32769 struct stat buffer;
32770 int      status;
32771 ...
32772 status = stat("/home/cnd/mod1", &buffer);
```

### 32773 Getting Directory Information

32774   The following example fragment gets status information for each entry in a directory. The call to  
 32775   the *stat()* function stores file information in the **stat** structure pointed to by *statbuf*. The lines  
 32776   that follow the *stat()* call format the fields in the **stat** structure for presentation to the user of the  
 32777   program.

```
32778 #include <sys/types.h>
32779 #include <sys/stat.h>
32780 #include <dirent.h>
32781 #include <pwd.h>
32782 #include <grp.h>
32783 #include <time.h>
32784 #include <locale.h>
32785 #include <langinfo.h>
32786 #include <stdio.h>
32787 #include <stdint.h>

32788 struct dirent *dp;
32789 struct stat   statbuf;
32790 struct passwd *pwd;
32791 struct group  *grp;
32792 struct tm     *tm;
32793 char          datestring[256];
32794 ...
32795 /* Loop through directory entries. */
32796 while ((dp = readdir(dir)) != NULL) {

32797     /* Get entry's information. */
32798     if (stat(dp->d_name, &statbuf) == -1)
```

```

32799         continue;

32800         /* Print out type, permissions, and number of links. */
32801         printf("%10.10s", spem (statbuf.st_mode));
32802         printf("%4d", statbuf.st_nlink);

32803         /* Print out owner's name if it is found using getpwuid(). */
32804         if ((pwd = getpwuid(statbuf.st_uid)) != NULL)
32805             printf(" %-8.8s", pwd->pw_name);
32806         else
32807             printf(" %-8d", statbuf.st_uid);

32808         /* Print out group name if it is found using getgrgid(). */
32809         if ((grp = getgrgid(statbuf.st_gid)) != NULL)
32810             printf(" %-8.8s", grp->gr_name);
32811         else
32812             printf(" %-8d", statbuf.st_gid);

32813         /* Print size of file. */
32814         printf(" %9jd", (intmax_t)statbuf.st_size);

32815         tm = localtime(&statbuf.st_mtime);

32816         /* Get localized date string. */
32817         strftime(datestring, sizeof(datestring), nl_langinfo(D_T_FMT), tm);

32818         printf(" %s %s\n", datestring, dp->d_name);
32819     }

```

### Obtaining Symbolic Link Status Information

The following example shows how to obtain status information for a symbolic link named **/modules/pass1**. The structure variable *buffer* is defined for the **stat** structure. If the *path* argument specified the pathname for the file pointed to by the symbolic link (**/home/cnd/mod1**), the results of calling the function would be the same as those returned by a call to the *stat()* function.

```

32826 #include <sys/stat.h>

32827 struct stat buffer;
32828 int status;
32829 ...
32830 status = lstat("/modules/pass1", &buffer);

```

### APPLICATION USAGE

None.

### RATIONALE

The intent of the paragraph describing “additional or alternate file access control mechanisms” is to allow a secure implementation where a process with a label that does not dominate the file’s label cannot perform a *stat()* function. This is not related to read permission; a process with a label that dominates the file’s label does not need read permission. An implementation that supports write-up operations could fail *fstat()* function calls even though it has a valid file descriptor open for writing.

The purpose of the *fstatat()* function is to obtain the status of files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *stat()*, resulting in unspecified behavior. By opening a



32843 file descriptor for the target directory and using the *fstatat()* function it can be guaranteed that  
 32844 the file for which status is returned is located relative to the desired directory.

#### 32845 FUTURE DIRECTIONS

32846 None.

#### 32847 SEE ALSO

32848 *access()*, *chmod()*, *fdopendir()*, *fstat()*, *mknod()*, *readlink()*, *symlink()*

32849 XBD Section 4.9 (on page 109), *<fcntl.h>*, *<sys/stat.h>*, *<sys/types.h>*

#### 32850 CHANGE HISTORY

32851 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 32852 Issue 5

32853 Large File Summit extensions are added.

#### 32854 Issue 6

32855 In the SYNOPSIS, the optional include of the *<sys/types.h>* header is removed.

32856 The following new requirements on POSIX implementations derive from alignment with the  
 32857 Single UNIX Specification:

- 32858 • The requirement to include *<sys/types.h>* has been removed. Although *<sys/types.h>* was  
 32859 required for conforming implementations of previous POSIX specifications, it was not  
 32860 required for UNIX applications.
- 32861 • The [EIO] mandatory error condition is added.
- 32862 • The [ELOOP] mandatory error condition is added.
- 32863 • The [EOVERFLOW] mandatory error condition is added. This change is to support large  
 32864 files.
- 32865 • The [ENAMETOOLONG] and the second [EOVERFLOW] optional error conditions are  
 32866 added.

32867 The following changes were made to align with the IEEE P1003.1a draft standard:

- 32868 • Details are added regarding the treatment of symbolic links.
- 32869 • The [ELOOP] optional error condition is added.

32870 The normative text is updated to avoid use of the term “must” for application requirements.

32871 The **restrict** keyword is added to the *stat()* prototype for alignment with the ISO/IEC 9899:1999  
 32872 standard.

#### 32873 Issue 7

32874 Austin Group Interpretation 1003.1-2001 #143 is applied.

32875 XSH-SD5-ERN-161 is applied, updating the DESCRIPTION to clarify to which file types *st\_nlink*  
 32876 applies.

32877 The *fstatat()* function is added from The Open Group Technical Standard, 2006, Extended API  
 32878 Set Part 2.

32879 Changes are made related to support for finegrained timestamps.

32880 The *lstat()* function is now required to return meaningful data for symbolic links in all **stat**  
 32881 structure fields, except for the permission bits of *st\_mode*.

32882 Changes are made to allow a directory to be opened for searching.

32883 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a  
32884 pathname exists but is not a directory or a symbolic link to a directory.

32885 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0198 [461], XSH/TC1-2008/0199 [324],  
32886 XSH/TC1-2008/0200 [278], XSH/TC1-2008/0201 [278], and XSH/TC1-2008/0202 [291] are  
32887 applied.

32888 **NAME**

32889 fstatvfs, statvfs — get file system information

32890 **SYNOPSIS**

32891 #include &lt;sys/statvfs.h&gt;

32892 int fstatvfs(int *fildev*, struct statvfs \**buf*);32893 int statvfs(const char \*restrict *path*, struct statvfs \*restrict *buf*);32894 **DESCRIPTION**32895 The *fstatvfs()* function shall obtain information about the file system containing the file  
32896 referenced by *fildev*.32897 The *statvfs()* function shall obtain information about the file system containing the file named by  
32898 *path*.32899 For both functions, the *buf* argument is a pointer to a **statvfs** structure that shall be filled. Read,  
32900 write, or execute permission of the named file is not required.32901 The following flags can be returned in the *f\_flag* member:

32902 ST\_RDONLY Read-only file system.

32903 ST\_NOSUID Setuid/setgid bits ignored by *exec*.32904 It is unspecified whether all members of the **statvfs** structure have meaningful values on all file  
32905 systems.32906 **RETURN VALUE**32907 Upon successful completion, *statvfs()* shall return 0. Otherwise, it shall return -1 and set *errno* to  
32908 indicate the error.32909 **ERRORS**32910 The *fstatvfs()* and *statvfs()* functions shall fail if:

32911 [EIO] An I/O error occurred while reading the file system.

32912 [EINTR] A signal was caught during execution of the function.

32913 [EOVERFLOW] One of the values to be returned cannot be represented correctly in the  
32914 structure pointed to by *buf*.32915 The *fstatvfs()* function shall fail if:32916 [EBADF] The *fildev* argument is not an open file descriptor.32917 The *statvfs()* function shall fail if:

32918 [EACCES] Search permission is denied on a component of the path prefix.

32919 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*  
32920 argument.

32921 [ENAMETOOLONG]

32922 The length of a component of a pathname is longer than {NAME\_MAX}.

32923 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.32924 [ENOTDIR] A component of the path prefix names an existing file that is neither a  
32925 directory nor a symbolic link to a directory, or the *path* argument contains at  
32926 least one non-*<slash>* character and ends with one or more trailing *<slash>*  
32927 characters and the last pathname component names an existing file that is  
32928 neither a directory nor a symbolic link to a directory.

32929 The *statvfs()* function may fail if:

32930 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
32931 resolution of the *path* argument.

32932 [ENAMETOOLONG]  
32933 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
32934 symbolic link produced an intermediate result with a length that exceeds  
32935 {PATH\_MAX}.

## 32936 EXAMPLES

### 32937 Obtaining File System Information Using *fstatvfs()*

32938 The following example shows how to obtain file system information for the file system upon  
32939 which the file named **/home/cnd/mod1** resides, using the *fstatvfs()* function. The  
32940 **/home/cnd/mod1** file is opened with read/write privileges and the open file descriptor is passed  
32941 to the *fstatvfs()* function.

```
32942 #include <sys/statvfs.h>
32943 #include <fcntl.h>

32944 struct statvfs buffer;
32945 int          status;
32946 ...
32947 fildes = open("/home/cnd/mod1", O_RDWR);
32948 status = fstatvfs(fildes, &buffer);
```

### 32949 Obtaining File System Information Using *statvfs()*

32950 The following example shows how to obtain file system information for the file system upon  
32951 which the file named **/home/cnd/mod1** resides, using the *statvfs()* function.

```
32952 #include <sys/statvfs.h>

32953 struct statvfs buffer;
32954 int          status;
32955 ...
32956 status = statvfs("/home/cnd/mod1", &buffer);
```

## 32957 APPLICATION USAGE

32958 None.

## 32959 RATIONALE

32960 None.

## 32961 FUTURE DIRECTIONS

32962 None.

## 32963 SEE ALSO

32964 *chmod()*, *chown()*, *creat()*, *dup()*, *exec*, *fcntl()*, *link()*, *mknod()*, *open()*, *pipe()*, *read()*, *time()*,  
32965 *unlink()*, *utime()*, *write()*

32966 XBD *<sys/statvfs.h>*

**CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

Large File Summit extensions are added.

**Issue 6**

The normative text is updated to avoid use of the term “must” for application requirements.

The **restrict** keyword is added to the *statvfs()* prototype for alignment with the ISO/IEC 9899:1999 standard.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #143 is applied.

SD5-XSH-ERN-68 is applied, correcting the EXAMPLES section.

The *fstatvfs()* and *statvfs()* functions are moved from the XSI option to the Base.

The [ENOTDIR] error condition is clarified to cover the condition where the last component of a pathname exists but is not a directory or a symbolic link to a directory.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0203 [324] is applied.

32985 **NAME**

32986 fsync — synchronize changes to a file

32987 **SYNOPSIS**

```
32988 #include <unistd.h>  
32989 int fsync(int fildes);
```

32990 **DESCRIPTION**

32991 The *fsync()* function shall request that all data for the open file descriptor named by *fildes* is to be  
32992 transferred to the storage device associated with the file described by *fildes*. The nature of the  
32993 transfer is implementation-defined. The *fsync()* function shall not return until the system has  
32994 completed that action or until an error is detected.

32995 SIO If `_POSIX_SYNCHRONIZED_IO` is defined, the *fsync()* function shall force all currently queued  
32996 I/O operations associated with the file indicated by file descriptor *fildes* to the synchronized I/O  
32997 completion state. All I/O operations shall be completed as defined for synchronized I/O file  
32998 integrity completion.

32999 **RETURN VALUE**

33000 Upon successful completion, *fsync()* shall return 0. Otherwise, -1 shall be returned and *errno* set  
33001 to indicate the error. If the *fsync()* function fails, outstanding I/O operations are not guaranteed  
33002 to have been completed.

33003 **ERRORS**33004 The *fsync()* function shall fail if:

- |       |          |                                                                                          |
|-------|----------|------------------------------------------------------------------------------------------|
| 33005 | [EBADF]  | The <i>fildes</i> argument is not a valid descriptor.                                    |
| 33006 | [EINTR]  | The <i>fsync()</i> function was interrupted by a signal.                                 |
| 33007 | [EINVAL] | The <i>fildes</i> argument does not refer to a file on which this operation is possible. |
| 33008 | [EIO]    | An I/O error occurred while reading from or writing to the file system.                  |

33009 In the event that any of the queued I/O operations fail, *fsync()* shall return the error conditions  
33010 defined for *read()* and *write()*.

33011 **EXAMPLES**

33012 None.

33013 **APPLICATION USAGE**

33014 The *fsync()* function should be used by programs which require modifications to a file to be  
33015 completed before continuing; for example, a program which contains a simple transaction  
33016 facility might use it to ensure that all modifications to a file or files caused by a transaction are  
33017 recorded.

33018 **RATIONALE**

33019 The *fsync()* function is intended to force a physical write of data from the buffer cache, and to  
33020 assure that after a system crash or other failure that all data up to the time of the *fsync()* call is  
33021 recorded on the disk. Since the concepts of “buffer cache”, “system crash”, “physical write”, and  
33022 “non-volatile storage” are not defined here, the wording has to be more abstract.

33023 If `_POSIX_SYNCHRONIZED_IO` is not defined, the wording relies heavily on the conformance  
33024 document to tell the user what can be expected from the system. It is explicitly intended that a  
33025 null implementation is permitted. This could be valid in the case where the system cannot assure  
33026 non-volatile storage under any circumstances or when the system is highly fault-tolerant and the  
33027 functionality is not required. In the middle ground between these extremes, *fsync()* might or  
33028 might not actually cause data to be written where it is safe from a power failure. The

conformance document should identify at least that one configuration exists (and how to obtain that configuration) where this can be assured for at least some files that the user can select to use for critical data. It is not intended that an exhaustive list is required, but rather sufficient information is provided so that if critical data needs to be saved, the user can determine how the system is to be configured to allow the data to be written to non-volatile storage.

It is reasonable to assert that the key aspects of *fsync()* are unreasonable to test in a test suite. That does not make the function any less valuable, just more difficult to test. A formal conformance test should probably force a system crash (power shutdown) during the test for this condition, but it needs to be done in such a way that automated testing does not require this to be done except when a formal record of the results is being made. It would also not be unreasonable to omit testing for *fsync()*, allowing it to be treated as a quality-of-implementation issue.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*sync()*

XBD <unistd.h>

#### CHANGE HISTORY

First released in Issue 3.

#### Issue 5

Aligned with *fsync()* in the POSIX Realtime Extension. Specifically, the DESCRIPTION and RETURN VALUE sections are much expanded, and the ERRORS section is updated to indicate that *fsync()* can return the error conditions defined for *read()* and *write()*.

#### Issue 6

This function is marked as part of the File Synchronization option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EINVAL] and [EIO] mandatory error conditions are added.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/44 is applied, applying an editorial rewording of the DESCRIPTION. No change in meaning is intended.

33059 **NAME**

33060 ftell, ftello — return a file offset in a stream

33061 **SYNOPSIS**

33062 #include &lt;stdio.h&gt;

33063 long ftell(FILE \*stream);

33064 CX off\_t ftello(FILE \*stream);

33065 **DESCRIPTION**

33066 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 33067 conflict between the requirements described here and the ISO C standard is unintentional. This  
 33068 volume of POSIX.1-2008 defers to the ISO C standard.

33069 The *ftell()* function shall obtain the current value of the file-position indicator for the stream  
 33070 pointed to by *stream*.

33071 The *ftell()* function shall not change the setting of *errno* if successful.

33072 CX The *ftello()* function shall be equivalent to *ftell()*, except that the return value is of type **off\_t** and  
 33073 the *ftello()* function may change the setting of *errno* if successful.

33074 **RETURN VALUE**

33075 CX Upon successful completion, *ftell()* and *ftello()* shall return the current value of the file-position  
 33076 indicator for the stream measured in bytes from the beginning of the file.

33077 Otherwise, *ftell()* and *ftello()* shall return  $-1$ , and set *errno* to indicate the error.

33078 **ERRORS**

33079 CX The *ftell()* and *ftello()* functions shall fail if:

33080 CX [EBADF] The file descriptor underlying *stream* is not an open file descriptor.

33081 CX [EOVERFLOW] For *ftell()*, the current file offset cannot be represented correctly in an object of  
 33082 type **long**.

33083 CX [EOVERFLOW] For *ftello()*, the current file offset cannot be represented correctly in an object  
 33084 of type **off\_t**.

33085 CX [ESPIPE] The file descriptor underlying *stream* is associated with a pipe, FIFO, or socket.

33086 **EXAMPLES**

33087 None.

33088 **APPLICATION USAGE**

33089 None.

33090 **RATIONALE**

33091 None.

33092 **FUTURE DIRECTIONS**

33093 None.

33094 **SEE ALSO**33095 Section 2.5 (on page 495), *fgetpos()*, *fopen()*, *fseek()*, *lseek()*

33096 XBD &lt;stdio.h&gt;



33097 **CHANGE HISTORY**

33098 First released in Issue 1. Derived from Issue 1 of the SVID.

33099 **Issue 5**

33100 Large File Summit extensions are added.

33101 **Issue 6**

33102 Extensions beyond the ISO C standard are marked.

33103 The following new requirements on POSIX implementations derive from alignment with the  
33104 Single UNIX Specification:

- 33105 • The *ftello()* function is added.
- 33106 • The [EOVERFLOW] error conditions are added.

33107 An additional [ESPIPE] error condition is added for sockets.

33108 **Issue 7**33109 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0204 [105], XSH/TC1-2008/0205 [421],  
33110 XSH/TC1-2008/0206 [122], XSH/TC1-2008/0207 [122], and XSH/TC1-2008/0208 [14] are  
33111 applied.

**NAME**

ftok — generate an IPC key

**SYNOPSIS**

```
#include <sys/ipc.h>

key_t ftok(const char *path, int id);
```

**DESCRIPTION**

The *ftok()* function shall return a key based on *path* and *id* that is usable in subsequent calls to *msgget()*, *semget()*, and *shmget()*. The application shall ensure that the *path* argument is the pathname of an existing file that the process is able to *stat()*, with the exception that if *stat()* would fail with [EOVERFLOW] due to file size, *ftok()* shall still succeed.

The *ftok()* function shall return the same key value for all paths that name the same file, when called with the same *id* value, and should return different key values when called with different *id* values or with paths that name different files existing on the same file system at the same time. It is unspecified whether *ftok()* shall return the same key value when called again after the file named by *path* is removed and recreated with the same name.

Only the low-order 8-bits of *id* are significant. The behavior of *ftok()* is unspecified if these bits are 0.

**RETURN VALUE**

Upon successful completion, *ftok()* shall return a key. Otherwise, *ftok()* shall return (**key\_t**)−1 and set *errno* to indicate the error.

**ERRORS**

The *ftok()* function shall fail if:

- [EACCES] Search permission is denied for a component of the path prefix.
- [EIO] An error occurred while reading from the file system.
- [ELOOP] A loop exists in symbolic links encountered during resolution of the *path* argument.
- [ENAMETOOLONG] The length of a component of a pathname is longer than {NAME\_MAX}.
- [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.
- [ENOTDIR] A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the *path* argument contains at least one non-*<slash>* character and ends with one or more trailing *<slash>* characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.

The *ftok()* function may fail if:

- [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during resolution of the *path* argument.
- [ENAMETOOLONG] The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH\_MAX}.

**EXAMPLES****Getting an IPC Key**

The following example gets a key based on the pathname `/tmp` and the ID value `a`. It also assigns the value of the resulting key to the `semkey` variable so that it will be available to a later call to `semget()`, `msgget()`, or `shmget()`.

```
#include <sys/ipc.h>
...
key_t semkey;

if ((semkey = ftok("/tmp", 'a')) == (key_t) -1) {
    perror("IPC error: ftok"); exit(1);
}
```

**APPLICATION USAGE**

For maximum portability, *id* should be a single-byte character.

Applications should not assume that the resulting key value is unique.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

Future versions of this standard may add new interfaces to provide unique keys.

**SEE ALSO**

[\*msgget\(\)\*](#), [\*semget\(\)\*](#), [\*shmget\(\)\*](#)

XBD [\*<sys/ipc.h>\*](#)

**CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

**Issue 6**

The normative text is updated to avoid use of the term “must” for application requirements.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #143 is applied.

The [ENOTDIR] error condition is clarified to cover the condition where the last component of a pathname exists but is not a directory or a symbolic link to a directory.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0209 [343], XSH/TC1-2008/0210 [366], XSH/TC1-2008/0211 [343], XSH/TC1-2008/0212 [324], XSH/TC1-2008/0213 [366], XSH/TC1-2008/0214 [366], XSH/TC1-2008/0215 [366], and XSH/TC1-2008/0216 [366] are applied.

33190 **NAME**

33191 ftruncate — truncate a file to a specified length

33192 **SYNOPSIS**

33193 #include &lt;unistd.h&gt;

33194 int ftruncate(int *fildes*, off\_t *length*);33195 **DESCRIPTION**33196 If *fildes* is not a valid file descriptor open for writing, the *ftruncate()* function shall fail.

33197 If *fildes* refers to a regular file, the *ftruncate()* function shall cause the size of the file to be truncated to *length*. If the size of the file previously exceeded *length*, the extra data shall no longer be available to reads on the file. If the file previously was smaller than this size, *ftruncate()* shall increase the size of the file. If the file size is increased, the extended area shall appear as if it were zero-filled. The value of the seek pointer shall not be modified by a call to *ftruncate()*.

33203 Upon successful completion, if *fildes* refers to a regular file, *ftruncate()* shall mark for update the last data modification and last file status change timestamps of the file and the S\_ISUID and S\_ISGID bits of the file mode may be cleared. If the *ftruncate()* function is unsuccessful, the file is unaffected.

33207 XSI If the request would cause the file size to exceed the soft file size limit for the process, the request shall fail and the implementation shall generate the SIGXFSZ signal for the thread.

33209 If *fildes* refers to a directory, *ftruncate()* shall fail.33210 If *fildes* refers to any other file type, except a shared memory object, the result is unspecified.

33211 SHM If *fildes* refers to a shared memory object, *ftruncate()* shall set the size of the shared memory object to *length*.

33213 SHM If the effect of *ftruncate()* is to decrease the size of a memory mapped file or a shared memory object and whole pages beyond the new end were previously mapped, then the whole pages beyond the new end shall be discarded.

33216 References to discarded pages shall result in the generation of a SIGBUS signal.

33217 If the effect of *ftruncate()* is to increase the size of a memory object, it is unspecified whether the contents of any mapped pages between the old end-of-file and the new are flushed to the underlying object.

33220 **RETURN VALUE**

33221 Upon successful completion, *ftruncate()* shall return 0; otherwise, -1 shall be returned and *errno* set to indicate the error.

33223 **ERRORS**33224 The *ftruncate()* function shall fail if:

33225 [EINTR] A signal was caught during execution.

33226 [EINVAL] The *length* argument was less than 0.

33227 [EFBIG] or [EINVAL]

33228 The *length* argument was greater than the maximum file size.

33229 [EFBIG] The file is a regular file and *length* is greater than the offset maximum established in the open file description associated with *fildes*.

33230

33231 [EIO] An I/O error occurred while reading from or writing to a file system.

33232 [EBADF] or [EINVAL]

33233 The *fildest* argument is not a file descriptor open for writing.

#### 33234 EXAMPLES

33235 None.

#### 33236 APPLICATION USAGE

33237 None.

#### 33238 RATIONALE

33239 None.

#### 33240 FUTURE DIRECTIONS

33241 None.

#### 33242 SEE ALSO

33243 *open()*, *truncate()*

33244 XBD <unistd.h>

#### 33245 CHANGE HISTORY

33246 First released in Issue 4, Version 2.

#### 33247 Issue 5

33248 Moved from X/OPEN UNIX extension to BASE and aligned with *ftruncate()* in the POSIX  
33249 Realtime Extension. Specifically, the DESCRIPTION is extensively reworded and [EROFS] is  
33250 added to the list of mandatory errors that can be returned by *ftruncate()*.

33251 Large File Summit extensions are added.

#### 33252 Issue 6

33253 The *truncate()* function is split out into a separate reference page.

33254 The following new requirements on POSIX implementations derive from alignment with the  
33255 Single UNIX Specification:

- 33256 • The DESCRIPTION is changed to indicate that if the file size is changed, and if the file is a  
33257 regular file, the S\_ISUID and S\_ISGID bits in the file mode may be cleared.

33258 The following changes were made to align with the IEEE P1003.1a draft standard:

- 33259 • The DESCRIPTION text is updated.

33260 XSI-conformant systems are required to increase the size of the file if the file was previously  
33261 smaller than the size requested.

#### 33262 Issue 7

33263 Austin Group Interpretation 1003.1-2001 #056 is applied, revising the ERRORS section (although  
33264 the [EINVAL] “may fail” error was subsequently removed during review of the XSI option).

33265 Functionality relating to the Memory Protection and Memory Mapped Files options is moved to  
33266 the Base.

33267 The DESCRIPTION is updated so that a call to *ftruncate()* when the file is smaller than the size  
33268 requested will increase the size of the file. Previously, non-XSI-conforming implementations  
33269 were allowed to increase the size of the file or fail.

33270 Changes are made related to support for finegrained timestamps.

33271 **NAME**  
33272       ftrylockfile — stdio locking functions

33273 **SYNOPSIS**  
33274 CX       #include <stdio.h>  
33275       int ftrylockfile(FILE \*file);

33276 **DESCRIPTION**  
33277       Refer to *flockfile()*.

**NAME**

ftw — traverse (walk) a file tree

**SYNOPSIS**

```
#include <ftw.h>

int ftw(const char *path, int (*fn)(const char *,
    const struct stat *ptr, int flag), int ndirs);
```

**DESCRIPTION**

The *ftw()* function shall recursively descend the directory hierarchy rooted in *path*. For each object in the hierarchy, *ftw()* shall call the function pointed to by *fn*, passing it a pointer to a null-terminated character string containing the name of the object, a pointer to a **stat** structure containing information about the object, filled in as if *stat()* or *lstat()* had been called to retrieve the information. Possible values of the integer, defined in the **<ftw.h>** header, are:

FTW\_D For a directory.

FTW\_DNR For a directory that cannot be read.

FTW\_F For a non-directory file.

FTW\_SL For a symbolic link (but see also FTW\_NS below).

FTW\_NS For an object other than a symbolic link on which *stat()* could not successfully be executed. If the object is a symbolic link and *stat()* failed, it is unspecified whether *ftw()* passes FTW\_SL or FTW\_NS to the user-supplied function.

If the integer is FTW\_DNR, descendants of that directory shall not be processed. If the integer is FTW\_NS, the **stat** structure contains undefined values. An example of an object that would cause FTW\_NS to be passed to the function pointed to by *fn* would be a file in a directory with read but without execute (search) permission.

The *ftw()* function shall visit a directory before visiting any of its descendants.

The *ftw()* function shall use at most one file descriptor for each level in the tree.

The argument *ndirs* should be in the range [1,{OPEN\_MAX}].

The tree traversal shall continue until either the tree is exhausted, an invocation of *fn* returns a non-zero value, or some error, other than [EACCES], is detected within *ftw()*.

The *ndirs* argument shall specify the maximum number of directory streams or file descriptors or both available for use by *ftw()* while traversing the tree. When *ftw()* returns it shall close any directory streams and file descriptors it uses not counting any opened by the application-supplied *fn* function.

The results are unspecified if the application-supplied *fn* function does not preserve the current working directory.

The *ftw()* function need not be thread-safe.

**RETURN VALUE**

If the tree is exhausted, *ftw()* shall return 0. If the function pointed to by *fn* returns a non-zero value, *ftw()* shall stop its tree traversal and return whatever value was returned by the function pointed to by *fn*. If *ftw()* detects an error, it shall return -1 and set *errno* to indicate the error.

If *ftw()* encounters an error other than [EACCES] (see FTW\_DNR and FTW\_NS above), it shall return -1 and set *errno* to indicate the error. The external variable *errno* may contain any error value that is possible when a directory is opened or when one of the *stat* functions is executed on

33320 a directory or file.

## 33321 ERRORS

33322 The *ftw()* function shall fail if:

33323 [EACCES] Search permission is denied for any component of *path* or read permission is  
33324 denied for *path*.

33325 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*  
33326 argument.

33327 [ENAMETOOLONG]

33328 The length of a component of a pathname is longer than {NAME\_MAX}.

33329 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.

33330 [ENOTDIR] A component of *path* names an existing file that is neither a directory nor a  
33331 symbolic link to a directory.

33332 [EOVERFLOW] A field in the **stat** structure cannot be represented correctly in the current  
33333 programming environment for one or more files found in the file hierarchy.

33334 The *ftw()* function may fail if:

33335 [EINVAL] The value of the *ndirs* argument is invalid.

33336 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
33337 resolution of the *path* argument.

33338 [ENAMETOOLONG]

33339 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
33340 symbolic link produced an intermediate result with a length that exceeds  
33341 {PATH\_MAX}.

33342 In addition, if the function pointed to by *fn* encounters system errors, *errno* may be set  
33343 accordingly.

## 33344 EXAMPLES

### 33345 Walking a Directory Structure

33346 The following example walks the current directory structure, calling the *fn* function for every  
33347 directory entry, using at most 10 file descriptors:

```
33348 #include <ftw.h>
33349 ...
33350 if (ftw(".", fn, 10) != 0) {
33351     perror("ftw"); exit(2);
33352 }
```

## 33353 APPLICATION USAGE

33354 The *ftw()* function may allocate dynamic storage during its operation. If *ftw()* is forcibly  
33355 terminated, such as by *longjmp()* or *siglongjmp()* being executed by the function pointed to by *fn*  
33356 or an interrupt routine, *ftw()* does not have a chance to free that storage, so it remains  
33357 permanently allocated. A safe way to handle interrupts is to store the fact that an interrupt has  
33358 occurred, and arrange to have the function pointed to by *fn* return a non-zero value at its next  
33359 invocation.

33360 Applications should use the *nftw()* function instead of the obsolescent *ftw()* function.



33361 **RATIONALE**

33362 None.

33363 **FUTURE DIRECTIONS**33364 The *ftw()* function may be removed in a future version.33365 **SEE ALSO**33366 *fdopendir()*, *fstatat()*, *longjmp()*, *nftw()*, *siglongjmp()*

33367 XBD &lt;ftw.h&gt;, &lt;sys/stat.h&gt;

33368 **CHANGE HISTORY**

33369 First released in Issue 1. Derived from Issue 1 of the SVID.

33370 **Issue 5**

33371 UX codings in the DESCRIPTION, RETURN VALUE, and ERRORS sections are changed to EX.

33372 **Issue 6**

33373 The ERRORS section is updated as follows:

- 33374 • The wording of the mandatory [ELOOP] error condition is updated.
- 33375 • A second optional [ELOOP] error condition is added.
- 33376 • The [Eoverflow] mandatory error condition is added.

33377 A note is added to the DESCRIPTION indicating that this function need not be reentrant, and  
 33378 that the results are unspecified if the application-supplied *fn* function does not preserve the  
 33379 current working directory.

33380 **Issue 7**

33381 Austin Group Interpretations 1003.1-2001 #143 and #156 are applied.

33382 SD5-XBD-ERN-61 is applied.

33383 The *ftw()* function is marked obsolescent.

33384 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0217 [403], XSH/TC1-2008/0218 [324],  
 33385 and XSH/TC1-2008/0219 [361] are applied.

33386 **NAME**

33387 funlockfile — stdio locking functions

33388 **SYNOPSIS**

```
33389 CX    #include <stdio.h>
33390        void funlockfile(FILE *file);
```

33391 **DESCRIPTION**33392 Refer to *flockfile()*.

## 33393 NAME

33394 futimens, utimensat, utimes — set file access and modification times

## 33395 SYNOPSIS

33396 #include &lt;sys/stat.h&gt;

33397 int futimens(int *fd*, const struct timespec *times*[2]);

33398 OH #include &lt;fcntl.h&gt; +

33399 +

33400 int utimensat(int *fd*, const char \**path*, const struct timespec *times*[2],  
33401 int *flag*);

33402 XSI #include &lt;sys/time.h&gt;

33403 int utimes(const char \**path*, const struct timeval *times*[2]);

## 33404 DESCRIPTION

33405 The *futimens()* and *utimensat()* functions shall set the access and modification times of a file to  
 33406 the values of the *times* argument. The *futimens()* function changes the times of the file associated  
 33407 with the file descriptor *fd*. The *utimensat()* function changes the times of the file pointed to by  
 33408 the *path* argument, relative to the directory associated with the file descriptor *fd*. Both functions  
 33409 allow time specifications accurate to the nanosecond.

33410 For *futimens()* and *utimensat()*, the *times* argument is an array of two **timespec** structures. The  
 33411 first array member represents the date and time of last access, and the second member  
 33412 represents the date and time of last modification. The times in the **timespec** structure are  
 33413 measured in seconds and nanoseconds since the Epoch. The file's relevant timestamp shall be set  
 33414 to the greatest value supported by the file system that is not greater than the specified time.

33415 If the *tv\_nsec* field of a **timespec** structure has the special value **UTIME\_NOW**, the file's relevant  
 33416 timestamp shall be set to the greatest value supported by the file system that is not greater than  
 33417 the current time. If the *tv\_nsec* field has the special value **UTIME\_OMIT**, the file's relevant  
 33418 timestamp shall not be changed. In either case, the *tv\_sec* field shall be ignored.

33419 If the *times* argument is a null pointer, both the access and modification timestamps shall be set  
 33420 to the greatest value supported by the file system that is not greater than the current time. If  
 33421 *utimensat()* is passed a relative path in the *path* argument, the file to be used shall be relative to  
 33422 the directory associated with the file descriptor *fd* instead of the current working directory. If the  
 33423 access mode of the open file description associated with the file descriptor is not **O\_SEARCH**,  
 33424 the function shall check whether directory searches are permitted using the current permissions  
 33425 of the directory underlying the file descriptor. If the access mode is **O\_SEARCH**, the function  
 33426 shall not perform the check.

33427 If *utimensat()* is passed the special value **AT\_FDCWD** in the *fd* parameter, the current working  
 33428 directory shall be used.

33429 Only a process with the effective user ID equal to the user ID of the file, or with write access to  
 33430 the file, or with appropriate privileges may use *futimens()* or *utimensat()* with a null pointer as  
 33431 the *times* argument or with both *tv\_nsec* fields set to the special value **UTIME\_NOW**. Only a  
 33432 process with the effective user ID equal to the user ID of the file or with appropriate privileges  
 33433 may use *futimens()* or *utimensat()* with a non-null *times* argument that does not have both  
 33434 *tv\_nsec* fields set to **UTIME\_NOW** and does not have both *tv\_nsec* fields set to **UTIME\_OMIT**. If  
 33435 both *tv\_nsec* fields are set to **UTIME\_OMIT**, no ownership or permissions check shall be  
 33436 performed for the file, but other error conditions may still be detected (including [EACCES]  
 33437 errors related to the path prefix).

Values for the *flag* argument of *utimensat()* are constructed by a bitwise-inclusive OR of flags from the following list, defined in `<fcntl.h>`:

AT\_SYMLINK\_NOFOLLOW

If *path* names a symbolic link, then the access and modification times of the symbolic link are changed.

Upon successful completion, *futimens()* and *utimensat()* shall mark the last file status change timestamp for update, with the exception that if both *tv\_nsec* fields are set to `UTIME_OMIT`, the file status change timestamp need not be marked for update.

The *utimes()* function shall be equivalent to the *utimensat()* function with the special value `AT_FDCWD` as the *fd* argument and the *flag* argument set to zero, except that the *times* argument is a **timeval** structure rather than a **timespec** structure, and accuracy is only to the microsecond, not nanosecond, and rounding towards the nearest second may occur.

## RETURN VALUE

Upon successful completion, these functions shall return 0. Otherwise, these functions shall return `-1` and set *errno* to indicate the error. If `-1` is returned, the file times shall not be affected.

## ERRORS

These functions shall fail if:

[EACCES] The *times* argument is a null pointer, or both *tv\_nsec* values are `UTIME_NOW`, and the effective user ID of the process does not match the owner of the file and write access is denied.

[EINVAL] Either of the *times* argument structures specified a *tv\_nsec* value that was neither `UTIME_NOW` nor `UTIME_OMIT`, and was a value less than zero or greater than or equal to 1 000 million.

[EINVAL] A new file timestamp would be a value whose *tv\_sec* component is not a value supported by the file system.

[EPERM] The *times* argument is not a null pointer, does not have both *tv\_nsec* fields set to `UTIME_NOW`, does not have both *tv\_nsec* fields set to `UTIME_OMIT`, the calling process' effective user ID does not match the owner of the file, and the calling process does not have appropriate privileges.

[EROFS] The file system containing the file is read-only.

The *futimens()* function shall fail if:

[EBADF] The *fd* argument is not a valid file descriptor.

The *utimensat()* function shall fail if:

[EACCES] The access mode of the open file description associated with *fd* is not `O_SEARCH` and the permissions of the directory underlying *fd* do not permit directory searches.

[EBADF] The *path* argument does not specify an absolute path and the *fd* argument is neither `AT_FDCWD` nor a valid file descriptor open for reading or searching.

[ENOTDIR] The *path* argument is not an absolute path and *fd* is a file descriptor associated with a non-directory file.

The *utimensat()* and *utimes()* functions shall fail if:

|       |                |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 33479 | [EACCES]       | Search permission is denied by a component of the path prefix.                                                                                                                                                                                                                                                                                                                                                    |
| 33480 | [ELOOP]        | A loop exists in symbolic links encountered during resolution of the <i>path</i> argument.                                                                                                                                                                                                                                                                                                                        |
| 33481 |                |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33482 | [ENAMETOOLONG] |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33483 |                | The length of a component of a pathname is longer than {NAME_MAX}.                                                                                                                                                                                                                                                                                                                                                |
| 33484 | [ENOENT]       | A component of <i>path</i> does not name an existing file or <i>path</i> is an empty string.                                                                                                                                                                                                                                                                                                                      |
| 33485 | [ENOTDIR]      | A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>path</i> argument contains at least one non- <code>&lt;slash&gt;</code> character and ends with one or more trailing <code>&lt;slash&gt;</code> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory. |
| 33486 |                |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33487 |                |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33488 |                |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33489 |                |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33490 |                | The <i>utimensat</i> () and <i>utimes</i> () functions may fail if:                                                                                                                                                                                                                                                                                                                                               |
| 33491 | [ELOOP]        | More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the <i>path</i> argument.                                                                                                                                                                                                                                                                                                            |
| 33492 |                |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33493 | [ENAMETOOLONG] |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33494 |                | The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.                                                                                                                                                                                                                                                     |
| 33495 |                |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33496 |                |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 33497 |                | The <i>utimensat</i> () function may fail if:                                                                                                                                                                                                                                                                                                                                                                     |
| 33498 | [EINVAL]       | The value of the <i>flag</i> argument is not valid.                                                                                                                                                                                                                                                                                                                                                               |

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

The purpose of the *utimensat*() function is to set the access and modification time of files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *utimes*(), resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *utimensat*() function it can be guaranteed that the changed file is located relative to the desired directory.

The standard developers considered including a special case for the permissions required by *utimensat*() when one *tv\_nsec* field is *UTIME\_NOW* and the other is *UTIME\_OMIT*. One possibility would be to include this case in with the cases where *times* is a null pointer or both fields are *UTIME\_NOW*, where the call is allowed if the process has write permission for the file. However, associating write permission with an update to just the last data access timestamp (which is normally updated by *read*()) did not seem appropriate. The other possibility would be to specify that this one case is allowed if the process has read permission, but this was felt to be too great a departure from the *utime*() and *utimes*() functions on which *utimensat*() is based. If an application needs to set the last data access timestamp to the current time for a file on which it has read permission but is not the owner, it can do so by opening the file, reading one or more bytes (or reading a directory entry, if the file is a directory), and then closing it.

33520 **FUTURE DIRECTIONS**

33521 None.

33522 **SEE ALSO**33523 *read()*, *utime()*

33524 XBD &lt;fcntl.h&gt;, &lt;sys/stat.h&gt;, &lt;sys/time.h&gt;

33525 **CHANGE HISTORY**

33526 First released in Issue 4, Version 2.

33527 **Issue 5**

33528 Moved from X/OPEN UNIX extension to BASE.

33529 **Issue 6**

33530 This function is marked LEGACY.

33531 The normative text is updated to avoid use of the term “must” for application requirements.

33532 The wording of the mandatory [ELOOP] error condition is updated, and a second optional  
33533 [ELOOP] error condition is added.33534 **Issue 7**

33535 Austin Group Interpretation 1003.1-2001 #143 is applied.

33536 The LEGACY marking is removed.

33537 The *utimensat()* function (renamed from *futimensat()*) is added from The Open Group Technical  
33538 Standard, 2006, Extended API Set Part 2, and changed to allow modifying a symbolic link by  
33539 adding a *flag* argument.33540 The *futimens()* function is added.

33541 Changes are made related to support for finegrained timestamps.

33542 Changes are made to allow a directory to be opened for searching.

33543 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a  
33544 pathname exists but is not a directory or a symbolic link to a directory.33545 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0220 [63,428], XSH/TC1-2008/0221  
33546 [278], XSH/TC1-2008/0222 [324], XSH/TC1-2008/0223 [306], and XSH/TC1-2008/0224 [278] are  
33547 applied.

33548 **NAME**

33549       fwide — set stream orientation

33550 **SYNOPSIS**

33551       #include &lt;stdio.h&gt;

33552       #include &lt;wchar.h&gt;

33553       int fwide(FILE \**stream*, int *mode*);33554 **DESCRIPTION**

33555 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 33556 conflict between the requirements described here and the ISO C standard is unintentional. This  
 33557 volume of POSIX.1-2008 defers to the ISO C standard.

33558       The *fwide()* function shall determine the orientation of the stream pointed to by *stream*. If *mode* is  
 33559 greater than zero, the function first attempts to make the stream wide-oriented. If *mode* is less  
 33560 than zero, the function first attempts to make the stream byte-oriented. Otherwise, *mode* is zero  
 33561 and the function does not alter the orientation of the stream.

33562       If the orientation of the stream has already been determined, *fwide()* shall not change it.

33563 CX       The *fwide()* function shall not change the setting of *errno* if successful.

33564       Since no return value is reserved to indicate an error, an application wishing to check for error  
 33565 situations should set *errno* to 0, then call *fwide()*, then check *errno*, and if it is non-zero, assume  
 33566 an error has occurred.

33567 **RETURN VALUE**

33568       The *fwide()* function shall return a value greater than zero if, after the call, the stream has wide-  
 33569 orientation, a value less than zero if the stream has byte-orientation, or zero if the stream has no  
 33570 orientation.

33571 **ERRORS**

33572       The *fwide()* function may fail if:

33573 CX       [EBADF]       The *stream* argument is not a valid stream.

33574 **EXAMPLES**

33575       None.

33576 **APPLICATION USAGE**

33577       A call to *fwide()* with *mode* set to zero can be used to determine the current orientation of a  
 33578 stream.

33579 **RATIONALE**

33580       None.

33581 **FUTURE DIRECTIONS**

33582       None.

33583 **SEE ALSO**

33584       XBD <stdio.h>, <wchar.h>

33585 **CHANGE HISTORY**

33586       First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
 33587 (E).

33588 **Issue 6**

33589 Extensions beyond the ISO C standard are marked.

33590 **Issue 7**

33591 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0225 [272] is applied.



## 33592 NAME

33593 fwprintf, swprintf, wprintf — print formatted wide-character output

## 33594 SYNOPSIS

```

33595 #include <stdio.h>
33596 #include <wchar.h>
33597 int fwprintf(FILE *restrict stream, const wchar_t *restrict format, ...);
33598 int swprintf(wchar_t *restrict ws, size_t n,
33599             const wchar_t *restrict format, ...);
33600 int wprintf(const wchar_t *restrict format, ...);

```

## 33601 DESCRIPTION

33602 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 33603 conflict between the requirements described here and the ISO C standard is unintentional. This  
 33604 volume of POSIX.1-2008 defers to the ISO C standard.

33605 The *fwprintf()* function shall place output on the named output *stream*. The *wprintf()* function  
 33606 shall place output on the standard output stream *stdout*. The *swprintf()* function shall place  
 33607 output followed by the null wide character in consecutive wide characters starting at *\*ws*; no  
 33608 more than *n* wide characters shall be written, including a terminating null wide character, which  
 33609 is always added (unless *n* is zero).

33610 Each of these functions shall convert, format, and print its arguments under control of the *format*  
 33611 wide-character string. The *format* is composed of zero or more directives: *ordinary wide-characters*,  
 33612 which are simply copied to the output stream, and *conversion specifications*, each of which results  
 33613 in the fetching of zero or more arguments. The results are undefined if there are insufficient  
 33614 arguments for the *format*. If the *format* is exhausted while arguments remain, the excess  
 33615 arguments are evaluated but are otherwise ignored.

33616 CX Conversions can be applied to the *n*th argument after the *format* in the argument list, rather than  
 33617 to the next unused argument. In this case, the conversion specifier wide character % (see below)  
 33618 is replaced by the sequence "%n\$", where *n* is a decimal integer in the range  
 33619 [1,{NL\_ARGMAX}], giving the position of the argument in the argument list. This feature  
 33620 provides for the definition of *format* wide-character strings that select arguments in an order  
 33621 appropriate to specific languages (see the EXAMPLES section).

33622 The *format* can contain either numbered argument specifications (that is, "%n\$" and "\*m\$"), or  
 33623 unnumbered argument conversion specifications (that is, % and \*), but not both. The only  
 33624 exception to this is that %% can be mixed with the "%n\$" form. The results of mixing numbered  
 33625 and unnumbered argument specifications in a *format* wide-character string are undefined. When  
 33626 numbered argument specifications are used, specifying the *N*th argument requires that all the  
 33627 leading arguments, from the first to the (*N*–1)th, are specified in the *format* wide-character string.

33628 In *format* wide-character strings containing the "%n\$" form of conversion specification,  
 33629 numbered arguments in the argument list can be referenced from the *format* wide-character  
 33630 string as many times as required.

33631 In *format* wide-character strings containing the % form of conversion specification, each  
 33632 argument in the argument list shall be used exactly once. It is unspecified whether an encoding  
 33633 error occurs if the format string contains **wchar\_t** values that do not correspond to members of  
 33634 the character set of the current locale and the specified semantics do not require that value to be  
 33635 processed by *wcrtomb()*.

33636 CX All forms of the *fwprintf()* function allow for the insertion of a locale-dependent radix character  
 33637 in the output string, output as a wide-character value. The radix character is defined in the  
 33638 current locale (category *LC\_NUMERIC*). In the POSIX locale, or in a locale where the radix

character is not defined, the radix character shall default to a <period> ( ' . ' ).

Each conversion specification is introduced by the ' % ' wide character or by the wide-character sequence "%n\$", after which the following appear in sequence:

- Zero or more *flags* (in any order), which modify the meaning of the conversion specification.
- An optional minimum *field width*. If the converted value has fewer wide characters than the field width, it shall be padded with <space> characters by default on the left; it shall be padded on the right, if the left-adjustment flag ( ' - ' ), described below, is given to the field width. The field width takes the form of an <asterisk> ( ' \* ' ), described below, or a decimal integer.
- An optional *precision* that gives the minimum number of digits to appear for the d, i, o, u, x, and X conversion specifiers; the number of digits to appear after the radix character for the a, A, e, E, f, and F conversion specifiers; the maximum number of significant digits for the g and G conversion specifiers; or the maximum number of wide characters to be printed from a string in the s conversion specifiers. The precision takes the form of a <period> ( ' . ' ) followed either by an <asterisk> ( ' \* ' ), described below, or an optional decimal digit string, where a null digit string is treated as 0. If a precision appears with any other conversion wide character, the behavior is undefined.
- An optional length modifier that specifies the size of the argument.
- A *conversion specifier* wide character that indicates the type of conversion to be applied.

A field width, or precision, or both, may be indicated by an <asterisk> ( ' \* ' ). In this case an argument of type **int** supplies the field width or precision. Applications shall ensure that arguments specifying field width, or precision, or both appear in that order before the argument, if any, to be converted. A negative field width is taken as a ' - ' flag followed by a positive field width. A negative precision is taken as if the precision were omitted. In *format* wide-character strings containing the "%n\$" form of a conversion specification, a field width or precision may be indicated by the sequence "\*m\$", where *m* is a decimal integer in the range [1,{NL\_ARGMAX}] giving the position in the argument list (after the *format* argument) of an integer argument containing the field width or precision, for example:

```
wprintf(L"%1$d:%2$.*3$d:%4$.*3$d\n", hour, min, precision, sec);
```

The flag wide characters and their meanings are:

- ' (The <apostrophe>.) The integer portion of the result of a decimal conversion (%i, %d, %u, %f, %F, %g, or %G) shall be formatted with thousands' grouping wide characters. For other conversions, the behavior is undefined. The numeric grouping wide character is used.
- The result of the conversion shall be left-justified within the field. The conversion shall be right-justified if this flag is not specified.
- + The result of a signed conversion shall always begin with a sign ( ' + ' or ' - ' ). The conversion shall begin with a sign only when a negative value is converted if this flag is not specified.
- <space> If the first wide character of a signed conversion is not a sign, or if a signed conversion results in no wide characters, a <space> shall be prefixed to the result. This means that if the <space> and ' + ' flags both appear, the <space> flag shall be ignored.

|       |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
|-------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| 33682 | #            | Specifies that the value is to be converted to an alternative form. For <code>o</code> conversion, it shall increase the precision, if and only if necessary, to force the first digit of the result to be zero (if the value and precision are both 0, a single 0 is printed). For <code>x</code> or <code>X</code> conversion specifiers, a non-zero result shall have 0x (or 0X) prefixed to it. For <code>a</code> , <code>A</code> , <code>e</code> , <code>E</code> , <code>f</code> , <code>F</code> , <code>g</code> , and <code>G</code> conversion specifiers, the result shall always contain a radix character, even if no digits follow it. Without this flag, a radix character appears in the result of these conversions only if a digit follows it. For <code>g</code> and <code>G</code> conversion specifiers, trailing zeros shall <i>not</i> be removed from the result as they normally are. For other conversion specifiers, the behavior is undefined.  |  |
| 33683 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33684 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33685 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33686 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33687 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33688 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33689 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33690 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33691 | 0            | For <code>d</code> , <code>i</code> , <code>o</code> , <code>u</code> , <code>x</code> , <code>X</code> , <code>a</code> , <code>A</code> , <code>e</code> , <code>E</code> , <code>f</code> , <code>F</code> , <code>g</code> , and <code>G</code> conversion specifiers, leading zeros (following any indication of sign or base) are used to pad to the field width rather than performing space padding, except when converting an infinity or NaN. If the <code>'0'</code> and <code>'-'</code> flags both appear, the <code>'0'</code> flag shall be ignored. For <code>d</code> , <code>i</code> , <code>o</code> , <code>u</code> , <code>x</code> , and <code>X</code> conversion specifiers, if a precision is specified, the <code>'0'</code> flag shall be ignored. If the <code>'0'</code> and <code>&lt;apostrophe&gt;</code> flags both appear, the grouping wide characters are inserted before zero padding. For other conversions, the behavior is undefined. |  |
| 33692 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33693 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33694 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33695 | CX           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33696 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33697 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33698 |              | The length modifiers and their meanings are:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |  |
| 33699 | hh           | Specifies that a following <code>d</code> , <code>i</code> , <code>o</code> , <code>u</code> , <code>x</code> , or <code>X</code> conversion specifier applies to a <b>signed char</b> or <b>unsigned char</b> argument (the argument will have been promoted according to the integer promotions, but its value shall be converted to <b>signed char</b> or <b>unsigned char</b> before printing); or that a following <code>n</code> conversion specifier applies to a pointer to a <b>signed char</b> argument.                                                                                                                                                                                                                                                                                                                                                                                                                                                              |  |
| 33700 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33701 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33702 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33703 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33704 | h            | Specifies that a following <code>d</code> , <code>i</code> , <code>o</code> , <code>u</code> , <code>x</code> , or <code>X</code> conversion specifier applies to a <b>short</b> or <b>unsigned short</b> argument (the argument will have been promoted according to the integer promotions, but its value shall be converted to <b>short</b> or <b>unsigned short</b> before printing); or that a following <code>n</code> conversion specifier applies to a pointer to a <b>short</b> argument.                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |  |
| 33705 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33706 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33707 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33708 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33709 | l (ell)      | Specifies that a following <code>d</code> , <code>i</code> , <code>o</code> , <code>u</code> , <code>x</code> , or <code>X</code> conversion specifier applies to a <b>long</b> or <b>unsigned long</b> argument; that a following <code>n</code> conversion specifier applies to a pointer to a <b>long</b> argument; that a following <code>c</code> conversion specifier applies to a <b>wint_t</b> argument; that a following <code>s</code> conversion specifier applies to a pointer to a <b>wchar_t</b> argument; or has no effect on a following <code>a</code> , <code>A</code> , <code>e</code> , <code>E</code> , <code>f</code> , <code>F</code> , <code>g</code> , or <code>G</code> conversion specifier.                                                                                                                                                                                                                                                         |  |
| 33710 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33711 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33712 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33713 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33714 | ll (ell-ell) | Specifies that a following <code>d</code> , <code>i</code> , <code>o</code> , <code>u</code> , <code>x</code> , or <code>X</code> conversion specifier applies to a <b>long long</b> or <b>unsigned long long</b> argument; or that a following <code>n</code> conversion specifier applies to a pointer to a <b>long long</b> argument.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |  |
| 33715 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33716 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33717 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33718 | j            | Specifies that a following <code>d</code> , <code>i</code> , <code>o</code> , <code>u</code> , <code>x</code> , or <code>X</code> conversion specifier applies to an <b>intmax_t</b> or <b>uintmax_t</b> argument; or that a following <code>n</code> conversion specifier applies to a pointer to an <b>intmax_t</b> argument.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33719 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33720 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33721 | z            | Specifies that a following <code>d</code> , <code>i</code> , <code>o</code> , <code>u</code> , <code>x</code> , or <code>X</code> conversion specifier applies to a <b>size_t</b> or the corresponding signed integer type argument; or that a following <code>n</code> conversion specifier applies to a pointer to a signed integer type corresponding to a <b>size_t</b> argument.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 33722 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33723 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33724 | t            | Specifies that a following <code>d</code> , <code>i</code> , <code>o</code> , <code>u</code> , <code>x</code> , or <code>X</code> conversion specifier applies to a <b>ptrdiff_t</b> or the corresponding <b>unsigned</b> type argument; or that a following <code>n</code> conversion specifier applies to a pointer to a <b>ptrdiff_t</b> argument.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 33725 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 33726 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |

|       |      |                                                                                                    |
|-------|------|----------------------------------------------------------------------------------------------------|
| 33727 | L    | Specifies that a following a, A, e, E, f, F, g, or G conversion specifier applies to a <b>long</b> |
| 33728 |      | <b>double</b> argument.                                                                            |
| 33729 |      | If a length modifier appears with any conversion specifier other than as specified above, the      |
| 33730 |      | behavior is undefined.                                                                             |
| 33731 |      | The conversion specifiers and their meanings are:                                                  |
| 33732 | d, i | The <b>int</b> argument shall be converted to a signed decimal in the style "[−]dddd". The         |
| 33733 |      | precision specifies the minimum number of digits to appear; if the value being                     |
| 33734 |      | converted can be represented in fewer digits, it shall be expanded with leading zeros.             |
| 33735 |      | The default precision shall be 1. The result of converting zero with an explicit precision         |
| 33736 |      | of zero shall be no wide characters.                                                               |
| 33737 | o    | The <b>unsigned</b> argument shall be converted to unsigned octal format in the style              |
| 33738 |      | "dddd". The precision specifies the minimum number of digits to appear; if the value               |
| 33739 |      | being converted can be represented in fewer digits, it shall be expanded with leading              |
| 33740 |      | zeros. The default precision shall be 1. The result of converting zero with an explicit            |
| 33741 |      | precision of zero shall be no wide characters.                                                     |
| 33742 | u    | The <b>unsigned</b> argument shall be converted to unsigned decimal format in the style            |
| 33743 |      | "dddd". The precision specifies the minimum number of digits to appear; if the value               |
| 33744 |      | being converted can be represented in fewer digits, it shall be expanded with leading              |
| 33745 |      | zeros. The default precision shall be 1. The result of converting zero with an explicit            |
| 33746 |      | precision of zero shall be no wide characters.                                                     |
| 33747 | x    | The <b>unsigned</b> argument shall be converted to unsigned hexadecimal format in the style        |
| 33748 |      | "dddd"; the letters "abcdef" are used. The precision specifies the minimum number                  |
| 33749 |      | of digits to appear; if the value being converted can be represented in fewer digits, it           |
| 33750 |      | shall be expanded with leading zeros. The default precision shall be 1. The result of              |
| 33751 |      | converting zero with an explicit precision of zero shall be no wide characters.                    |
| 33752 | X    | Equivalent to the x conversion specifier, except that letters "ABCDEF" are used instead            |
| 33753 |      | of "abcdef".                                                                                       |
| 33754 | f, F | The <b>double</b> argument shall be converted to decimal notation in the style                     |
| 33755 |      | "[−]ddd.ddd", where the number of digits after the radix character shall be equal to               |
| 33756 |      | the precision specification. If the precision is missing, it shall be taken as 6; if the           |
| 33757 |      | precision is explicitly zero and no '#' flag is present, no radix character shall appear. If       |
| 33758 |      | a radix character appears, at least one digit shall appear before it. The value shall be           |
| 33759 |      | rounded in an implementation-defined manner to the appropriate number of digits.                   |
| 33760 |      | A <b>double</b> argument representing an infinity shall be converted in one of the styles          |
| 33761 |      | "[−]inf" or "[−]infinity"; which style is implementation-defined. A <b>double</b>                  |
| 33762 |      | argument representing a NaN shall be converted in one of the styles "[−]nan" or                    |
| 33763 |      | "[−]nan( <i>n-char-sequence</i> )"; which style, and the meaning of any <i>n-char-sequence</i> ,   |
| 33764 |      | is implementation-defined. The F conversion specifier produces "INF", "INFINITY",                  |
| 33765 |      | or "NAN" instead of "inf", "infinity", or "nan", respectively.                                     |
| 33766 | e, E | The <b>double</b> argument shall be converted in the style "[−]d.ddde±dd", where there             |
| 33767 |      | shall be one digit before the radix character (which is non-zero if the argument is non-           |
| 33768 |      | zero) and the number of digits after it shall be equal to the precision; if the precision is       |
| 33769 |      | missing, it shall be taken as 6; if the precision is zero and no '#' flag is present, no           |
| 33770 |      | radix character shall appear. The value shall be rounded in an implementation-defined              |
| 33771 |      | manner to the appropriate number of digits. The E conversion wide character shall                  |
| 33772 |      | produce a number with 'E' instead of 'e' introducing the exponent. The exponent                    |
| 33773 |      | shall always contain at least two digits. If the value is zero, the exponent shall be zero.        |

33774 A **double** argument representing an infinity or NaN shall be converted in the style of  
 33775 an **f** or **F** conversion specifier.

33776 **g, G** The **double** argument representing a floating-point number shall be converted in the  
 33777 style **f** or **e** (or in the style **F** or **E** in the case of a **G** conversion specifier), depending on  
 33778 the value converted and the precision. Let **P** equal the precision if non-zero, 6 if the  
 33779 precision is omitted, or 1 if the precision is zero. Then, if a conversion with style **E**  
 33780 would have an exponent of **X**:

33781 — If  $P > X \geq -4$ , the conversion shall be with style **f** (or **F**) and precision  $P - (X + 1)$ .  
 33782 — Otherwise, the conversion shall be with style **e** (or **E**) and precision  $P - 1$ .

33783 Finally, unless the '**#**' flag is used, any trailing zeros shall be removed from the  
 33784 fractional portion of the result and the decimal-point character shall be removed if there  
 33785 is no fractional portion remaining.

33786 A **double** argument representing an infinity or NaN shall be converted in the style of  
 33787 an **f** or **F** conversion specifier.

33788 **a, A** A **double** argument representing a floating-point number shall be converted in the  
 33789 style "**[−]0xh.hhhhp±d**", where there shall be one hexadecimal digit (which is non-  
 33790 zero if the argument is a normalized floating-point number and is otherwise  
 33791 unspecified) before the decimal-point wide character and the number of hexadecimal  
 33792 digits after it shall be equal to the precision; if the precision is missing and **FLT\_RADIX**  
 33793 is a power of 2, then the precision shall be sufficient for an exact representation of the  
 33794 value; if the precision is missing and **FLT\_RADIX** is not a power of 2, then the precision  
 33795 shall be sufficient to distinguish values of type **double**, except that trailing zeros may  
 33796 be omitted; if the precision is zero and the '**#**' flag is not specified, no decimal-point  
 33797 wide character shall appear. The letters "**abcdef**" are used for a conversion and the  
 33798 letters "**ABCDEF**" for **A** conversion. The **A** conversion specifier produces a number with  
 33799 '**X**' and '**P**' instead of '**x**' and '**p**'. The exponent shall always contain at least one  
 33800 digit, and only as many more digits as necessary to represent the decimal exponent of  
 33801 2. If the value is zero, the exponent shall be zero.

33802 A **double** argument representing an infinity or NaN shall be converted in the style of  
 33803 an **f** or **F** conversion specifier.

33804 **c** If no **l** (ell) qualifier is present, the **int** argument shall be converted to a wide character  
 33805 as if by calling the **btowc()** function and the resulting wide character shall be written.  
 33806 Otherwise, the **wint\_t** argument shall be converted to **wchar\_t**, and written.

33807 **s** If no **l** (ell) qualifier is present, the application shall ensure that the argument is a  
 33808 pointer to a character array containing a character sequence beginning in the initial  
 33809 shift state. Characters from the array shall be converted as if by repeated calls to the  
 33810 **mbrtowc()** function, with the conversion state described by an **mbstate\_t** object  
 33811 initialized to zero before the first character is converted, and written up to (but not  
 33812 including) the terminating null wide character. If the precision is specified, no more  
 33813 than that many wide characters shall be written. If the precision is not specified, or is  
 33814 greater than the size of the array, the application shall ensure that the array contains a  
 33815 null wide character.

33816 If an **l** (ell) qualifier is present, the application shall ensure that the argument is a  
 33817 pointer to an array of type **wchar\_t**. Wide characters from the array shall be written up  
 33818 to (but not including) a terminating null wide character. If no precision is specified, or  
 33819 is greater than the size of the array, the application shall ensure that the array contains a  
 33820 null wide character. If a precision is specified, no more than that many wide characters



|       |     |                                                                                                                         |
|-------|-----|-------------------------------------------------------------------------------------------------------------------------|
| 33821 |     | shall be written.                                                                                                       |
| 33822 | p   | The application shall ensure that the argument is a pointer to <b>void</b> . The value of the                           |
| 33823 |     | pointer shall be converted to a sequence of printable wide characters in an                                             |
| 33824 |     | implementation-defined manner.                                                                                          |
| 33825 | n   | The application shall ensure that the argument is a pointer to an integer into which is                                 |
| 33826 |     | written the number of wide characters written to the output so far by this call to one of                               |
| 33827 |     | the <i>fwprintf()</i> functions. No argument shall be converted, but one shall be consumed. If                          |
| 33828 |     | the conversion specification includes any flags, a field width, or a precision, the                                     |
| 33829 |     | behavior is undefined.                                                                                                  |
| 33830 | XSI | <b>C</b> Equivalent to <b>lc</b> .                                                                                      |
| 33831 | XSI | <b>S</b> Equivalent to <b>ls</b> .                                                                                      |
| 33832 | %   | Output a ' % ' wide character; no argument shall be converted. The entire conversion                                    |
| 33833 |     | specification shall be %%.                                                                                              |
| 33834 |     | If a conversion specification does not match one of the above forms, the behavior is undefined.                         |
| 33835 |     | In no case does a nonexistent or small field width cause truncation of a field; if the result of a                      |
| 33836 |     | conversion is wider than the field width, the field shall be expanded to contain the conversion                         |
| 33837 |     | result. Characters generated by <i>fwprintf()</i> and <i>wprintf()</i> shall be printed as if <i>fputwc()</i> had been  |
| 33838 |     | called.                                                                                                                 |
| 33839 |     | For a and A conversions, if FLT_RADIX is not a power of 2 and the result is not exactly                                 |
| 33840 |     | representable in the given precision, the result should be one of the two adjacent numbers in                           |
| 33841 |     | hexadecimal floating style with the given precision, with the extra stipulation that the error                          |
| 33842 |     | should have a correct sign for the current rounding direction.                                                          |
| 33843 |     | For e, E, f, F, g, and G conversion specifiers, if the number of significant decimal digits is at                       |
| 33844 |     | most DECIMAL_DIG, then the result should be correctly rounded. If the number of significant                             |
| 33845 |     | decimal digits is more than DECIMAL_DIG but the source value is exactly representable with                              |
| 33846 |     | DECIMAL_DIG digits, then the result should be an exact representation with trailing zeros.                              |
| 33847 |     | Otherwise, the source value is bounded by two adjacent decimal strings $L < U$ , both having                            |
| 33848 |     | DECIMAL_DIG significant digits; the value of the resultant decimal string $D$ should satisfy $L \leq$                   |
| 33849 |     | $D \leq U$ , with the extra stipulation that the error should have a correct sign for the current                       |
| 33850 |     | rounding direction.                                                                                                     |
| 33851 | CX  | The last data modification and last file status change timestamps of the file shall be marked for                       |
| 33852 |     | update between the call to a successful execution of <i>fwprintf()</i> or <i>wprintf()</i> and the next                 |
| 33853 |     | successful completion of a call to <i>fflush()</i> or <i>fclose()</i> on the same stream, or a call to <i>exit()</i> or |
| 33854 |     | <i>abort()</i> .                                                                                                        |
| 33855 |     | <b>RETURN VALUE</b>                                                                                                     |
| 33856 |     | Upon successful completion, these functions shall return the number of wide characters                                  |
| 33857 |     | transmitted, excluding the terminating null wide character in the case of <i>swprintf()</i> , or a negative             |
| 33858 | CX  | value if an output error was encountered, and set <i>errno</i> to indicate the error.                                   |
| 33859 |     | If <i>n</i> or more wide characters were requested to be written, <i>swprintf()</i> shall return a negative             |
| 33860 | CX  | value, and set <i>errno</i> to indicate the error.                                                                      |
| 33861 |     | <b>ERRORS</b>                                                                                                           |
| 33862 |     | For the conditions under which <i>fwprintf()</i> and <i>wprintf()</i> fail and may fail, refer to <i>fputwc()</i> .     |
| 33863 |     | In addition, all forms of <i>fwprintf()</i> shall fail if:                                                              |

33864 CX [EILSEQ] A wide-character code that does not correspond to a valid character has been  
33865 detected.

33866 In addition, *fwprintf()* and *wprintf()* may fail if: -

33867 CX [ENOMEM] Insufficient storage space is available.

33868 The *swprintf()* shall fail if:

33869 CX [EOVERFLOW] The value of *n* is greater than {INT\_MAX} or the number of bytes needed to  
33870 hold the output excluding the terminating null is greater than {INT\_MAX}.

### 33871 EXAMPLES

33872 To print the language-independent date and time format, the following statement could be used:

33873 `wprintf(format, weekday, month, day, hour, min);`

33874 For American usage, *format* could be a pointer to the wide-character string:

33875 `L"%s, %s %d, %d:%.2d\n"`

33876 producing the message:

33877 `Sunday, July 3, 10:02`

33878 whereas for German usage, *format* could be a pointer to the wide-character string:

33879 `L"%1$s, %3$d. %2$s, %4$d:%5$.2d\n"`

33880 producing the message:

33881 `Sonntag, 3. Juli, 10:02`

### 33882 APPLICATION USAGE

33883 None.

### 33884 RATIONALE

33885 If an implementation detects that there are insufficient arguments for the format, it is  
33886 recommended that the function should fail and report an [EINVAL] error.

### 33887 FUTURE DIRECTIONS

33888 None.

### 33889 SEE ALSO

33890 [Section 2.5](#) (on page 495), [btowc\(\)](#), [fputwc\(\)](#), [fwscanf\(\)](#), [mbrtowc\(\)](#), [setlocale\(\)](#)

33891 [XBD Chapter 7](#) (on page 135), [<inttypes.h>](#), [<stdio.h>](#), [<wchar.h>](#)

### 33892 CHANGE HISTORY

33893 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
33894 (E).

### 33895 Issue 6

33896 The Open Group Corrigendum U040/1 is applied to the RETURN VALUE section, describing  
33897 the case if *n* or more wide characters are requested to be written using *swprintf()*.

33898 The normative text is updated to avoid use of the term “must” for application requirements.

33899 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 33900 • The prototypes for *fwprintf()*, *swprintf()*, and *wprintf()* are updated.

- 33901 • The DESCRIPTION is updated.
- 33902 • The hh, ll, j, t, and z length modifiers are added.
- 33903 • The a, A, and F conversion characters are added.
- 33904 • XSI shading is removed from the description of character string representations of infinity
- 33905 and NaN floating-point values.
- 33906 The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion
- 33907 specification” consistently.
- 33908 ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.
- 33909 **Issue 7**
- 33910 Austin Group Interpretation 1003.1-2001 #161 is applied, updating the DESCRIPTION of the 0
- 33911 flag.
- 33912 Austin Group Interpretation 1003.1-2001 #170 is applied.
- 33913 ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #68 (SD5-XSH-ERN-70) is applied,
- 33914 revising the description of g and G.
- 33915 Functionality relating to the "%n\$" form of conversion specification and the <apostrophe> flag
- 33916 is moved from the XSI option to the Base.
- 33917 The [Eoverflow] error is added for *swprintf*().
- 33918 Changes are made related to support for finegrained timestamps.
- 33919 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0226 [302] and XSH/TC1-2008/0227
- 33920 [14] are applied.



33921 **NAME**

33922       fwrite — binary output

33923 **SYNOPSIS**

33924       #include &lt;stdio.h&gt;

```
33925       size_t fwrite(const void *restrict ptr, size_t size, size_t nitems,
33926                   FILE *restrict stream);
```

33927 **DESCRIPTION**

33928 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 33929       conflict between the requirements described here and the ISO C standard is unintentional. This  
 33930       volume of POSIX.1-2008 defers to the ISO C standard.

33931       The *fwrite()* function shall write, from the array pointed to by *ptr*, up to *nitems* elements whose  
 33932       size is specified by *size*, to the stream pointed to by *stream*. For each object, *size* calls shall be  
 33933       made to the *fputc()* function, taking the values (in order) from an array of **unsigned char** exactly  
 33934       overlying the object. The file-position indicator for the stream (if defined) shall be advanced by  
 33935       the number of bytes successfully written. If an error occurs, the resulting value of the file-  
 33936       position indicator for the stream is unspecified.

33937 CX       The last data modification and last file status change timestamps of the file shall be marked for  
 33938       update between the successful execution of *fwrite()* and the next successful completion of a call  
 33939       to *fflush()* or *fclose()* on the same stream, or a call to *exit()* or *abort()*.

33940 **RETURN VALUE**

33941       The *fwrite()* function shall return the number of elements successfully written, which may be  
 33942       less than *nitems* if a write error is encountered. If *size* or *nitems* is 0, *fwrite()* shall return 0 and the  
 33943       state of the stream remains unchanged. Otherwise, if a write error occurs, the error indicator for  
 33944       CX       the stream shall be set, and *errno* shall be set to indicate the error.

33945 **ERRORS**33946       Refer to *fputc()*.33947 **EXAMPLES**

33948       None.

33949 **APPLICATION USAGE**

33950       Because of possible differences in element length and byte ordering, files written using *fwrite()*  
 33951       are application-dependent, and possibly cannot be read using *fread()* by a different application  
 33952       or by the same application on a different processor.

33953 **RATIONALE**

33954       None.

33955 **FUTURE DIRECTIONS**

33956       None.

33957 **SEE ALSO**33958       Section 2.5 (on page 495), *ferror()*, *fopen()*, *fprintf()*, *putc()*, *puts()*, *write()*

33959       XBD &lt;stdio.h&gt;

33960 **CHANGE HISTORY**

33961       First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 6**

Extensions beyond the ISO C standard are marked.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *fwrite()* prototype is updated.
- The DESCRIPTION is updated to clarify how the data is written out using *fputc()*.

**Issue 7**

Changes are made related to support for finegrained timestamps.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0228 [14] is applied.

## 33970 NAME

33971 fwscanf, swscanf, wscanf — convert formatted wide-character input

## 33972 SYNOPSIS

```

33973 #include <stdio.h>
33974 #include <wchar.h>

33975 int fwscanf(FILE *restrict stream, const wchar_t *restrict format, ...);
33976 int swscanf(const wchar_t *restrict ws,
33977             const wchar_t *restrict format, ...);
33978 int wscanf(const wchar_t *restrict format, ...);

```

## 33979 DESCRIPTION

33980 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 33981 conflict between the requirements described here and the ISO C standard is unintentional. This  
 33982 volume of POSIX.1-2008 defers to the ISO C standard.

33983 The *fwscanf()* function shall read from the named input *stream*. The *wscanf()* function shall read  
 33984 from the standard input stream *stdin*. The *swscanf()* function shall read from the wide-character  
 33985 string *ws*. Each function reads wide characters, interprets them according to a format, and stores  
 33986 the results in its arguments. Each expects, as arguments, a control wide-character string *format*  
 33987 described below, and a set of *pointer* arguments indicating where the converted input should be  
 33988 stored. The result is undefined if there are insufficient arguments for the format. If the *format* is  
 33989 exhausted while arguments remain, the excess arguments are evaluated but are otherwise  
 33990 ignored.

33991 CX Conversions can be applied to the *n*th argument after the *format* in the argument list, rather than  
 33992 to the next unused argument. In this case, the conversion specifier wide character % (see below)  
 33993 is replaced by the sequence "%n\$", where *n* is a decimal integer in the range  
 33994 [1,{NL\_ARGMAX}]. This feature provides for the definition of *format* wide-character strings that  
 33995 select arguments in an order appropriate to specific languages. In *format* wide-character strings  
 33996 containing the "%n\$" form of conversion specifications, it is unspecified whether numbered  
 33997 arguments in the argument list can be referenced from the *format* wide-character string more  
 33998 than once.

33999 The *format* can contain either form of a conversion specification—that is, % or "%n\$" — but the  
 34000 two forms cannot normally be mixed within a single *format* wide-character string. The only  
 34001 exception to this is that %% or %\* can be mixed with the "%n\$" form. When numbered argument  
 34002 specifications are used, specifying the *N*th argument requires that all the leading arguments,  
 34003 from the first to the (*N*–1)th, are pointers.

34004 The *fwscanf()* function in all its forms allows for detection of a language-dependent radix  
 34005 character in the input string, encoded as a wide-character value. The radix character is defined  
 34006 in the current locale (category *LC\_NUMERIC*). In the POSIX locale, or in a locale where the  
 34007 radix character is not defined, the radix character shall default to a <period> ('.').

34008 The *format* is a wide-character string composed of zero or more directives. Each directive is  
 34009 composed of one of the following: one or more white-space wide characters (<space>, <tab>,  
 34010 <newline>, <vertical-tab>, or <form-feed>); an ordinary wide character (neither '%' nor a  
 34011 white-space character); or a conversion specification. It is unspecified whether an encoding  
 34012 error occurs if the format string contains **wchar\_t** values that do not correspond to members of  
 34013 the character set of the current locale and the specified semantics do not require that value to be  
 34014 processed by *wcrtomb()*.

34015 CX Each conversion specification is introduced by the '%' or by the character sequence "%n\$",  
 34016 after which the following appear in sequence:

|       |    |                                                                                                              |
|-------|----|--------------------------------------------------------------------------------------------------------------|
| 34017 |    | • An optional assignment-suppressing character ' * '.                                                        |
| 34018 |    | • An optional non-zero decimal integer that specifies the maximum field width.                               |
| 34019 | CX | • An optional assignment-allocation character ' m '.                                                         |
| 34020 |    | • An optional length modifier that specifies the size of the receiving object.                               |
| 34021 |    | • A conversion specifier wide character that specifies the type of conversion to be applied.                 |
| 34022 |    | The valid conversion specifiers are described below.                                                         |
| 34023 |    | The <i>fwscanf()</i> functions shall execute each directive of the format in turn. If a directive fails, as  |
| 34024 |    | detailed below, the function shall return. Failures are described as input failures (due to the              |
| 34025 |    | unavailability of input bytes) or matching failures (due to inappropriate input).                            |
| 34026 |    | A directive composed of one or more white-space wide characters is executed by reading input                 |
| 34027 |    | until no more valid input can be read, or up to the first wide character which is not a white-               |
| 34028 |    | space wide character, which remains unread.                                                                  |
| 34029 |    | A directive that is an ordinary wide character shall be executed as follows. The next wide                   |
| 34030 |    | character is read from the input and compared with the wide character that comprises the                     |
| 34031 |    | directive; if the comparison shows that they are not equivalent, the directive shall fail, and the           |
| 34032 |    | differing and subsequent wide characters remain unread. Similarly, if end-of-file, an encoding               |
| 34033 |    | error, or a read error prevents a wide character from being read, the directive shall fail.                  |
| 34034 |    | A directive that is a conversion specification defines a set of matching input sequences, as                 |
| 34035 |    | described below for each conversion wide character. A conversion specification is executed in                |
| 34036 |    | the following steps.                                                                                         |
| 34037 |    | Input white-space wide characters (as specified by <i>iswspace()</i> ) shall be skipped, unless the          |
| 34038 |    | conversion specification includes a [ , c, or n conversion specifier.                                        |
| 34039 |    | An item shall be read from the input, unless the conversion specification includes an n                      |
| 34040 |    | conversion specifier wide character. An input item is defined as the longest sequence of input               |
| 34041 |    | wide characters, not exceeding any specified field width, which is an initial subsequence of a               |
| 34042 |    | matching sequence. The first wide character, if any, after the input item shall remain unread. If            |
| 34043 |    | the length of the input item is zero, the execution of the conversion specification shall fail; this         |
| 34044 |    | condition is a matching failure, unless end-of-file, an encoding error, or a read error prevented            |
| 34045 |    | input from the stream, in which case it is an input failure.                                                 |
| 34046 |    | Except in the case of a % conversion specifier, the input item (or, in the case of a %n conversion           |
| 34047 |    | specification, the count of input wide characters) shall be converted to a type appropriate to the           |
| 34048 |    | conversion wide character. If the input item is not a matching sequence, the execution of the                |
| 34049 |    | conversion specification shall fail; this condition is a matching failure. Unless assignment                 |
| 34050 |    | suppression was indicated by a ' * ', the result of the conversion shall be placed in the object             |
| 34051 |    | pointed to by the first argument following the <i>format</i> argument that has not already received a        |
| 34052 | CX | conversion result if the conversion specification is introduced by %, or in the <i>n</i> th argument if      |
| 34053 |    | introduced by the wide-character sequence "%n\$". If this object does not have an appropriate                |
| 34054 |    | type, or if the result of the conversion cannot be represented in the space provided, the behavior           |
| 34055 |    | is undefined.                                                                                                |
| 34056 | CX | The %c, %s, and %[ conversion specifiers shall accept an optional assignment-allocation                      |
| 34057 |    | character ' m ', which shall cause a memory buffer to be allocated to hold the wide-character                |
| 34058 |    | string converted including a terminating null wide character. In such a case, the argument                   |
| 34059 |    | corresponding to the conversion specifier should be a reference to a pointer value that will                 |
| 34060 |    | receive a pointer to the allocated buffer. The system shall allocate a buffer as if <i>malloc()</i> had been |
| 34061 |    | called. The application shall be responsible for freeing the memory after usage. If there is                 |
| 34062 |    | insufficient memory to allocate a buffer, the function shall set <i>errno</i> to [ENOMEM] and a              |

conversion error shall result. If the function returns EOF, any memory successfully allocated for parameters using assignment-allocation character 'm' by this call shall be freed before the function returns.

The length modifiers and their meanings are:

hh Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **signed char** or **unsigned char**.

h Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **short** or **unsigned short**.

l (ell) Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **long** or **unsigned long**; that a following a, A, e, E, f, F, g, or G conversion specifier applies to an argument with type pointer to **double**; or that a following c, s, or l conversion specifier applies to an argument with type pointer to **wchar\_t**. If the 'm' assignment-allocation character is specified, the conversion applies to an argument with the type pointer to a pointer to **wchar\_t**.

ll (ell-ell) Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **long long** or **unsigned long long**.

j Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **intmax\_t** or **uintmax\_t**.

z Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **size\_t** or the corresponding signed integer type.

t Specifies that a following d, i, o, u, x, X, or n conversion specifier applies to an argument with type pointer to **ptrdiff\_t** or the corresponding **unsigned** type.

L Specifies that a following a, A, e, E, f, F, g, or G conversion specifier applies to an argument with type pointer to **long double**.

If a length modifier appears with any conversion specifier other than as specified above, the behavior is undefined.

The following conversion specifier wide characters are valid:

d Matches an optionally signed decimal integer, whose format is the same as expected for the subject sequence of *wcstol()* with the value 10 for the *base* argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to **int**.

i Matches an optionally signed integer, whose format is the same as expected for the subject sequence of *wcstol()* with 0 for the *base* argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to **int**.

o Matches an optionally signed octal integer, whose format is the same as expected for the subject sequence of *wcstoul()* with the value 8 for the *base* argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to **unsigned**.

u Matches an optionally signed decimal integer, whose format is the same as expected for the subject sequence of *wcstoul()* with the value 10 for the *base* argument. In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to **unsigned**.

|       |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 34107 | x          | Matches an optionally signed hexadecimal integer, whose format is the same as expected for the subject sequence of <i>wcstoul()</i> with the value 16 for the <i>base</i> argument.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 34108 |            | In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to <b>unsigned</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 34109 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34110 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34111 | a, e, f, g | Matches an optionally signed floating-point number, infinity, or NaN whose format is the same as expected for the subject sequence of <i>wcstod()</i> . In the absence of a size modifier, the application shall ensure that the corresponding argument is a pointer to <b>float</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 34112 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34113 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34114 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34115 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34116 |            | If the <i>fwprintf()</i> family of functions generates character string representations for infinity and NaN (a symbolic entity encoded in floating-point format) to support IEEE Std 754-1985, the <i>fwscanf()</i> family of functions shall recognize them as input.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 34117 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34118 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34119 | s          | Matches a sequence of non-white-space wide characters. If no 1 (ell) qualifier is present, characters from the input field shall be converted as if by repeated calls to the <i>wcrtomb()</i> function, with the conversion state described by an <b>mbstate_t</b> object initialized to zero before the first wide character is converted. If the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to a character array large enough to accept the sequence and the terminating null character, which shall be added automatically. Otherwise, the application shall ensure that the corresponding argument is a pointer to a pointer to a <b>wchar_t</b> .                                                             |
| 34120 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34121 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34122 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34123 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34124 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34125 | CX         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34126 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34127 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34128 |            | If the 1 (ell) qualifier is present and the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to an array of <b>wchar_t</b> large enough to accept the sequence and the terminating null wide character, which shall be added automatically. If the 1 (ell) qualifier is present and the 'm' assignment-allocation character is present, the application shall ensure that the corresponding argument is a pointer to a pointer to a <b>wchar_t</b> .                                                                                                                                                                                                                                                                     |
| 34129 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34130 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34131 | CX         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34132 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34133 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34134 | [          | Matches a non-empty sequence of wide characters from a set of expected wide characters (the <i>scanset</i> ). If no 1 (ell) qualifier is present, wide characters from the input field shall be converted as if by repeated calls to the <i>wcrtomb()</i> function, with the conversion state described by an <b>mbstate_t</b> object initialized to zero before the first wide character is converted. If the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to a character array large enough to accept the sequence and the terminating null character, which shall be added automatically. Otherwise, the application shall ensure that the corresponding argument is a pointer to a pointer to a <b>wchar_t</b> . |
| 34135 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34136 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34137 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34138 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34139 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34140 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34141 | CX         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34142 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34143 |            | If an 1 (ell) qualifier is present and the 'm' assignment-allocation character is not specified, the application shall ensure that the corresponding argument is a pointer to an array of <b>wchar_t</b> large enough to accept the sequence and the terminating null wide character. If an 1 (ell) qualifier is present and the 'm' assignment-allocation character is specified, the application shall ensure that the corresponding argument is a pointer to a pointer to a <b>wchar_t</b> .                                                                                                                                                                                                                                                                                                         |
| 34144 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34145 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34146 | CX         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34147 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34148 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34149 |            | The conversion specification includes all subsequent wide characters in the <i>format</i> string up to and including the matching <right-square-bracket> (' <i>]</i> '). The wide characters between the square brackets (the <i>scanlist</i> ) comprise the scanset, unless the wide character after the <left-square-bracket> is a <circumflex> (' <i>^</i> '), in which case the scanset contains all wide characters that do not appear in the scanlist between the <circumflex> and the <right-square-bracket>. If the conversion specification begins                                                                                                                                                                                                                                             |
| 34150 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34151 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34152 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34153 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 34154 |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |



|       |     |   |                                                                                                      |
|-------|-----|---|------------------------------------------------------------------------------------------------------|
| 34155 |     |   | with "[ ]" or "[ ^ ]", the <right-square-bracket> is included in the scanlist and the                |
| 34156 |     |   | next <right-square-bracket> is the matching <right-square-bracket> that ends the                     |
| 34157 |     |   | conversion specification; otherwise, the first <right-square-bracket> is the one that ends           |
| 34158 |     |   | the conversion specification. If a '-' is in the scanlist and is not the first wide character,       |
| 34159 |     |   | nor the second where the first wide character is a '^', nor the last wide character, the             |
| 34160 |     |   | behavior is implementation-defined.                                                                  |
| 34161 | c   |   | Matches a sequence of wide characters of exactly the number specified by the field                   |
| 34162 |     |   | width (1 if no field width is present in the conversion specification).                              |
| 34163 |     |   | If no l (ell) length modifier is present, characters from the input field shall be converted         |
| 34164 |     |   | as if by repeated calls to the <i>wcrtomb()</i> function, with the conversion state described by     |
| 34165 |     |   | an <b>mbstate_t</b> object initialized to zero before the first wide character is converted. No      |
| 34166 |     |   | null character is added. If the 'm' assignment-allocation character is not specified, the            |
| 34167 |     |   | application shall ensure that the corresponding argument is a pointer to the initial                 |
| 34168 | CX  |   | element of a character array large enough to accept the sequence. Otherwise, the                     |
| 34169 |     |   | application shall ensure that the corresponding argument is a pointer to a pointer to a              |
| 34170 |     |   | <b>char</b> .                                                                                        |
| 34171 |     |   | No null wide character is added. If an l (ell) length modifier is present and the 'm'                |
| 34172 |     |   | assignment-allocation character is not specified, the application shall ensure that the              |
| 34173 |     |   | corresponding argument shall be a pointer to the initial element of an array of <b>wchar_t</b>       |
| 34174 | CX  |   | large enough to accept the sequence. If an l (ell) qualifier is present and the 'm'                  |
| 34175 |     |   | assignment-allocation character is specified, the application shall ensure that the                  |
| 34176 |     |   | corresponding argument is a pointer to a pointer to a <b>wchar_t</b> .                               |
| 34177 | p   |   | Matches an implementation-defined set of sequences, which shall be the same as the set               |
| 34178 |     |   | of sequences that is produced by the %p conversion specification of the corresponding                |
| 34179 |     |   | <i>fwprintf()</i> functions. The application shall ensure that the corresponding argument is a       |
| 34180 |     |   | pointer to a pointer to <b>void</b> . The interpretation of the input item is implementation-        |
| 34181 |     |   | defined. If the input item is a value converted earlier during the same program                      |
| 34182 |     |   | execution, the pointer that results shall compare equal to that value; otherwise, the                |
| 34183 |     |   | behavior of the %p conversion is undefined.                                                          |
| 34184 | n   |   | No input is consumed. The application shall ensure that the corresponding argument is                |
| 34185 |     |   | a pointer to the integer into which is to be written the number of wide characters read              |
| 34186 |     |   | from the input so far by this call to the <i>fwscanf()</i> functions. Execution of a %n              |
| 34187 |     |   | conversion specification shall not increment the assignment count returned at the                    |
| 34188 |     |   | completion of execution of the function. No argument shall be converted, but one shall               |
| 34189 |     |   | be consumed. If the conversion specification includes an assignment-suppressing wide                 |
| 34190 |     |   | character or a field width, the behavior is undefined.                                               |
| 34191 | XSI | C | Equivalent to lc.                                                                                    |
| 34192 | XSI | S | Equivalent to ls.                                                                                    |
| 34193 | %   |   | Matches a single '%' wide character; no conversion or assignment shall occur. The                    |
| 34194 |     |   | complete conversion specification shall be %%.                                                       |
| 34195 |     |   | If a conversion specification is invalid, the behavior is undefined.                                 |
| 34196 |     |   | The conversion specifiers A, E, F, G, and X are also valid and shall be equivalent to, respectively, |
| 34197 |     |   | a, e, f, g, and x.                                                                                   |
| 34198 |     |   | If end-of-file is encountered during input, conversion is terminated. If end-of-file occurs before   |
| 34199 |     |   | any wide characters matching the current conversion specification (except for %n) have been          |
| 34200 |     |   | read (other than leading white-space, where permitted), execution of the current conversion          |

specification shall terminate with an input failure. Otherwise, unless execution of the current conversion specification is terminated with a matching failure, execution of the following conversion specification (if any) shall be terminated with an input failure.

Reaching the end of the string in *swscanf()* shall be equivalent to encountering end-of-file for *fwscanf()*.

If conversion terminates on a conflicting input, the offending input shall be left unread in the input. Any trailing white space (including <newline>) shall be left unread unless matched by a conversion specification. The success of literal matches and suppressed assignments is only directly determinable via the %n conversion specification.

The *fwscanf()* and *wscanf()* functions may mark the last data access timestamp of the file associated with *stream* for update. The last data access timestamp shall be marked for update by the first successful execution of *fgetwc()*, *fgetws()*, *fwscanf()*, *getwc()*, *getwchar()*, *vfwscanf()*, *vwscanf()*, or *wscanf()* using *stream* that returns data not supplied by a prior call to *ungetwc()*.

## RETURN VALUE

Upon successful completion, these functions shall return the number of successfully matched and assigned input items; this number can be zero in the event of an early matching failure. If the input ends before the first conversion (if any) has completed, and without a matching failure having occurred, EOF shall be returned. If an error occurs before the first conversion (if any) has completed, and without a matching failure having occurred, EOF shall be returned and *errno* shall be set to indicate the error. If a read error occurs, the error indicator for the stream shall be set.

## ERRORS

For the conditions under which the *fwscanf()* functions shall fail and may fail, refer to *fgetwc()*.

In addition, the *fwscanf()* function shall fail if:

[EILSEQ] Input byte sequence does not form a valid character.

[ENOMEM] Insufficient storage space is available.

In addition, the *fwscanf()* function may fail if:

[EINVAL] There are insufficient arguments.

## EXAMPLES

The call:

```
int i, n; float x; char name[50];
n = wscanf(L"%d%f%s", &i, &x, name);
```

with the input line:

```
25 54.32E-1 Hamster
```

assigns to *n* the value 3, to *i* the value 25, to *x* the value 5.432, and *name* contains the string "Hamster".

The call:

```
int i; float x; char name[50];
(void) wscanf(L"%2d%f%d %[0123456789]", &i, &x, name);
```

with input:

```
56789 0123 56a72
```

assigns 56 to *i*, 789.0 to *x*, skips 0123, and places the string "56\0" in *name*. The next call to



34243 `getchar()` shall return the character 'a'.

#### 34244 APPLICATION USAGE

34245 In format strings containing the '%' form of conversion specifications, each argument in the  
34246 argument list is used exactly once.

34247 For functions that allocate memory as if by `malloc()`, the application should release such memory  
34248 when it is no longer required by a call to `free()`. For `fwscanf()`, this is memory allocated via use of  
34249 the 'm' assignment-allocation character.

#### 34250 RATIONALE

34251 None.

#### 34252 FUTURE DIRECTIONS

34253 None.

#### 34254 SEE ALSO

34255 [Section 2.5](#) (on page 495), [getwc\(\)](#), [fwprintf\(\)](#), [setlocale\(\)](#), [wcstod\(\)](#), [wcstol\(\)](#), [wcstoul\(\)](#), [wcrctomb\(\)](#)

34256 [XBD Chapter 7](#) (on page 135), [<inttypes.h>](#), [<stdio.h>](#), [<wchar.h>](#)

#### 34257 CHANGE HISTORY

34258 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
34259 (E).

#### 34260 Issue 6

34261 The normative text is updated to avoid use of the term “must” for application requirements.

34262 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 34263 • The prototypes for `fwscanf()` and `swscanf()` are updated.
- 34264 • The DESCRIPTION is updated.
- 34265 • The hh, ll, j, t, and z length modifiers are added.
- 34266 • The a, A, and F conversion characters are added.

34267 The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion  
34268 specification” consistently.

#### 34269 Issue 7

34270 Austin Group Interpretation 1003.1-2001 #170 is applied.

34271 SD5-XSH-ERN-132 is applied, adding the assignment-allocation character 'm'.

34272 Functionality relating to the "%n\$" form of conversion specification is moved from the XSI  
34273 option to the Base.

34274 Changes are made related to support for finegrained timestamps.

34275 The APPLICATION USAGE section is updated to clarify that memory is allocated as if by  
34276 `malloc()`.

34277 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0229 [302] and XSH/TC1-2008/0230  
34278 [14] are applied.

## NAME

gai\_strerror — address and name information error description

## SYNOPSIS

```
#include <netdb.h>
```

```
const char *gai_strerror(int ecode);
```

## DESCRIPTION

The *gai\_strerror()* function shall return a text string describing an error value for the *getaddrinfo()* and *getnameinfo()* functions listed in the **<netdb.h>** header.

When the *ecode* argument is one of the following values listed in the **<netdb.h>** header:

|                |                |
|----------------|----------------|
| [EAI_AGAIN]    | [EAI_NONAME]   |
| [EAI_BADFLAGS] | [EAI_OVERFLOW] |
| [EAI_FAIL]     | [EAI_SERVICE]  |
| [EAI_FAMILY]   | [EAI_SOCKTYPE] |
| [EAI_MEMORY]   | [EAI_SYSTEM]   |

the function return value shall point to a string describing the error. If the argument is not one of those values, the function shall return a pointer to a string whose contents indicate an unknown error.

## RETURN VALUE

Upon successful completion, *gai\_strerror()* shall return a pointer to an implementation-defined string.

## ERRORS

No errors are defined.

## EXAMPLES

None.

## APPLICATION USAGE

None.

## RATIONALE

None.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*freeaddrinfo()*

XBD **<netdb.h>**

## CHANGE HISTORY

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

The Open Group Base Resolution bwg2001-009 is applied, which changes the return type from **char \*** to **const char \***. This is for coordination with the IPnG Working Group.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/22 is applied, adding the [EAI\_OVERFLOW] error code.

34318 **NAME**

34319       getaddrinfo — get address information

34320 **SYNOPSIS**

34321       #include &lt;sys/socket.h&gt;

34322       #include &lt;netdb.h&gt;

34323       int getaddrinfo(const char \*restrict *nodename*,34324                      const char \*restrict *servname*,34325                      const struct addrinfo \*restrict *hints*,34326                      struct addrinfo \*\*restrict *res*);34327 **DESCRIPTION**34328       Refer to *freeaddrinfo()*.

34329 **NAME**

34330       getc — get a byte from a stream

34331 **SYNOPSIS**

34332       #include &lt;stdio.h&gt;

34333       int getc(FILE \*stream);

34334 **DESCRIPTION**

34335 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
34336 conflict between the requirements described here and the ISO C standard is unintentional. This  
34337 volume of POSIX.1-2008 defers to the ISO C standard.

34338       The *getc()* function shall be equivalent to *fgetc()*, except that if it is implemented as a macro it  
34339 may evaluate *stream* more than once, so the argument should never be an expression with side-  
34340 effects.

34341 **RETURN VALUE**34342       Refer to *fgetc()*.34343 **ERRORS**34344       Refer to *fgetc()*.34345 **EXAMPLES**

34346       None.

34347 **APPLICATION USAGE**

34348       If the integer value returned by *getc()* is stored into a variable of type **char** and then compared  
34349 against the integer constant EOF, the comparison may never succeed, because sign-extension of  
34350 a variable of type **char** on widening to integer is implementation-defined.

34351       Since it may be implemented as a macro, *getc()* may treat incorrectly a *stream* argument with  
34352 side-effects. In particular, *getc(\*f++)* does not necessarily work as expected. Therefore, use of this  
34353 function should be preceded by "#undef getc" in such situations; *fgetc()* could also be used.

34354 **RATIONALE**

34355       None.

34356 **FUTURE DIRECTIONS**

34357       None.

34358 **SEE ALSO**34359       Section 2.5 (on page 495), *fgetc()*

34360       XBD &lt;stdio.h&gt;

34361 **CHANGE HISTORY**

34362       First released in Issue 1. Derived from Issue 1 of the SVID.

34363 **Issue 7**

34364       POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0231 [14] is applied.

34365 **NAME**

34366        `getc_unlocked`, `getchar_unlocked`, `putc_unlocked`, `putchar_unlocked` — stdio with explicit client  
 34367        locking

34368 **SYNOPSIS**

```
34369 CX      #include <stdio.h>
34370
34371      int getc_unlocked(FILE *stream);
34372      int getchar_unlocked(void);
34373      int putc_unlocked(int c, FILE *stream);
34374      int putchar_unlocked(int c);
```

34374 **DESCRIPTION**

34375        Versions of the functions `getc()`, `getchar()`, `putc()`, and `putchar()` respectively named  
 34376        `getc_unlocked()`, `getchar_unlocked()`, `putc_unlocked()`, and `putchar_unlocked()` shall be provided  
 34377        which are functionally equivalent to the original versions, with the exception that they are not  
 34378        required to be implemented in a fully thread-safe manner. They shall be thread-safe when used  
 34379        within a scope protected by `flockfile()` (or `ftrylockfile()`) and `funlockfile()`. These functions can  
 34380        safely be used in a multi-threaded program if and only if they are called while the invoking  
 34381        thread owns the (**FILE \***) object, as is the case after a successful call to the `flockfile()` or  
 34382        `ftrylockfile()` functions.

34383        If `getc_unlocked()` or `putc_unlocked()` are implemented as macros they may evaluate *stream* more  
 34384        than once, so the *stream* argument should never be an expression with side-effects.

34385 **RETURN VALUE**

34386        See `getc()`, `getchar()`, `putc()`, and `putchar()`.

34387 **ERRORS**

34388        See `getc()`, `getchar()`, `putc()`, and `putchar()`.

34389 **EXAMPLES**

34390        None.

34391 **APPLICATION USAGE**

34392        Since they may be implemented as macros, `getc_unlocked()` and `putc_unlocked()` may treat  
 34393        incorrectly a *stream* argument with side-effects. In particular, `getc_unlocked(*f++)` and  
 34394        `putc_unlocked(c,*f++)` do not necessarily work as expected. Therefore, use of these functions in  
 34395        such situations should be preceded by the following statement as appropriate:

```
34396      #undef getc_unlocked
34397      #undef putc_unlocked
```

34398 **RATIONALE**

34399        Some I/O functions are typically implemented as macros for performance reasons (for example,  
 34400        `putc()` and `getc()`). For safety, they need to be synchronized, but it is often too expensive to  
 34401        synchronize on every character. Nevertheless, it was felt that the safety concerns were more  
 34402        important; consequently, the `getc()`, `getchar()`, `putc()`, and `putchar()` functions are required to be  
 34403        thread-safe. However, unlocked versions are also provided with names that clearly indicate the  
 34404        unsafe nature of their operation but can be used to exploit their higher performance. These  
 34405        unlocked versions can be safely used only within explicitly locked program regions, using  
 34406        exported locking primitives. In particular, a sequence such as:

```
34407      flockfile(fileptr);
34408      putc_unlocked('l', fileptr);
34409      putc_unlocked('\n', fileptr);
34410      fprintf(fileptr, "Line 2\n");
```

34411        `funlockfile(fileptr);`  
 34412        is permissible, and results in the text sequence:  
 34413        1  
 34414        Line 2  
 34415        being printed without being interspersed with output from other threads.

34416        It would be wrong to have the standard names such as `getc()`, `putc()`, and so on, map to the  
 34417        “faster, but unsafe” rather than the “slower, but safe” versions. In either case, you would still  
 34418        want to inspect all uses of `getc()`, `putc()`, and so on, by hand when converting existing code.  
 34419        Choosing the safe bindings as the default, at least, results in correct code and maintains the  
 34420        “atomicity at the function” invariant. To do otherwise would introduce gratuitous  
 34421        synchronization errors into converted code. Other routines that modify the *stdio* (**FILE** \*)  
 34422        structures or buffers are also safely synchronized.

34423        Note that there is no need for functions of the form `getc_locked()`, `putc_locked()`, and so on, since  
 34424        this is the functionality of `getc()`, `putc()`, *et al.* It would be inappropriate to use a feature test  
 34425        macro to switch a macro definition of `getc()` between `getc_locked()` and `getc_unlocked()`, since the  
 34426        ISO C standard requires an actual function to exist, a function whose behavior could not be  
 34427        changed by the feature test macro. Also, providing both the `xxx_locked()` and `xxx_unlocked()`  
 34428        forms leads to the confusion of whether the suffix describes the behavior of the function or the  
 34429        circumstances under which it should be used.

34430        Three additional routines, `flockfile()`, `ftrylockfile()`, and `funlockfile()` (which may be macros), are  
 34431        provided to allow the user to delineate a sequence of I/O statements that are executed  
 34432        synchronously.

34433        The `ungetc()` function is infrequently called relative to the other functions/macros so no  
 34434        unlocked variation is needed.

34435        **FUTURE DIRECTIONS**  
 34436        None.

34437        **SEE ALSO**  
 34438        [Section 2.5](#) (on page 495), [flockfile\(\)](#), [getc\(\)](#), [getchar\(\)](#), [putc\(\)](#), [putchar\(\)](#)  
 34439        [XBD <stdio.h>](#)

34440        **CHANGE HISTORY**  
 34441        First released in Issue 5. Included for alignment with the POSIX Threads Extension.

34442        **Issue 6**  
 34443        These functions are marked as part of the Thread-Safe Functions option.  
 34444        The Open Group Corrigendum U030/2 is applied, adding APPLICATION USAGE describing  
 34445        how applications should be written to avoid the case when the functions are implemented as  
 34446        macros.

34447        **Issue 7**  
 34448        The `getc_unlocked()`, `getchar_unlocked()`, `putc_unlocked()`, and `putchar_unlocked()` functions are  
 34449        moved from the Thread-Safe Functions option to the Base.  
 34450        POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0232 [395], XSH/TC1-2008/0233 [395],  
 34451        XSH/TC1-2008/0234 [395], and XSH/TC1-2008/0235 [14] are applied.

34452 **NAME**34453        `getchar` — get a byte from a *stdin* stream34454 **SYNOPSIS**34455        `#include <stdio.h>`34456        `int getchar(void);`34457 **DESCRIPTION**

34458 **CX**        The functionality described on this reference page is aligned with the ISO C standard. Any  
 34459 conflict between the requirements described here and the ISO C standard is unintentional. This  
 34460 volume of POSIX.1-2008 defers to the ISO C standard.

34461        The `getchar()` function shall be equivalent to `getc(stdin)`.34462 **RETURN VALUE**34463        Refer to `fgetc()`.34464 **ERRORS**34465        Refer to `fgetc()`.34466 **EXAMPLES**

34467        None.

34468 **APPLICATION USAGE**

34469        If the integer value returned by `getchar()` is stored into a variable of type **char** and then  
 34470 compared against the integer constant EOF, the comparison may never succeed, because sign-  
 34471 extension of a variable of type **char** on widening to integer is implementation-defined.

34472 **RATIONALE**

34473        None.

34474 **FUTURE DIRECTIONS**

34475        None.

34476 **SEE ALSO**34477        Section 2.5 (on page 495), `getc()`34478        XBD `<stdio.h>`34479 **CHANGE HISTORY**

34480        First released in Issue 1. Derived from Issue 1 of the SVID.

34481 **Issue 7**

34482        POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0236 [14] is applied.

# getchar\_unlocked()

System Interfaces

## 34483 NAME

34484 getchar\_unlocked — stdio with explicit client locking

## 34485 SYNOPSIS

```
34486 CX #include <stdio.h>  
34487 int getchar_unlocked(void);
```

## 34488 DESCRIPTION

34489 Refer to *getc\_unlocked()*.



**NAME**

getcwd — get the pathname of the current working directory

**SYNOPSIS**

```
#include <unistd.h>
```

```
char *getcwd(char *buf, size_t size);
```

**DESCRIPTION**

The *getcwd()* function shall place an absolute pathname of the current working directory in the array pointed to by *buf*, and return *buf*. The pathname shall contain no components that are dot or dot-dot, or are symbolic links.

If there are multiple pathnames that *getcwd()* could place in the array pointed to by *buf*, one beginning with a single <slash> character and one or more beginning with two <slash> characters, then *getcwd()* shall place the pathname beginning with a single <slash> character in the array. The pathname shall not contain any unnecessary <slash> characters after the leading one or two <slash> characters.

The *size* argument is the size in bytes of the character array pointed to by the *buf* argument. If *buf* is a null pointer, the behavior of *getcwd()* is unspecified.

**RETURN VALUE**

Upon successful completion, *getcwd()* shall return the *buf* argument. Otherwise, *getcwd()* shall return a null pointer and set *errno* to indicate the error. The contents of the array pointed to by *buf* are then undefined.

**ERRORS**

The *getcwd()* function shall fail if:

[EINVAL] The *size* argument is 0.

[ERANGE] The *size* argument is greater than 0, but is smaller than the length of the string +1.

The *getcwd()* function may fail if:

[EACCES] Search permission was denied for the current directory, or read or search permission was denied for a directory above the current directory in the file hierarchy.

[ENOMEM] Insufficient storage space is available.

**EXAMPLES**

The following example uses {PATH\_MAX} as the initial buffer size (unless it is indeterminate or very large), and calls *getcwd()* with progressively larger buffers until it does not give an [ERANGE] error.

```
#include <stdlib.h>
#include <errno.h>
#include <unistd.h>
```

```
...
```

```
long path_max;
size_t size;
char *buf;
char *ptr;
```

```
path_max = pathconf(".", _PC_PATH_MAX);
if (path_max == -1)
```

```

34534         size = 1024;
34535     else if (path_max > 10240)
34536         size = 10240;
34537     else
34538         size = path_max;
34539     for (buf = ptr = NULL; ptr == NULL; size *= 2)
34540     {
34541         if ((buf = realloc(buf, size)) == NULL)
34542         {
34543             ... handle error ...
34544         }
34545         ptr = getcwd(buf, size);
34546         if (ptr == NULL && errno != ERANGE)
34547         {
34548             ... handle error ...
34549         }
34550     }
34551     ...
34552     free (buf);

```

#### APPLICATION USAGE

If the pathname obtained from *getcwd()* is longer than {PATH\_MAX} bytes, it could produce an [ENAMETOOLONG] error if passed to *chdir()*. Therefore, in order to return to that directory it may be necessary to break the pathname into sections shorter than {PATH\_MAX} bytes and call *chdir()* on each section in turn (the first section being an absolute pathname and subsequent sections being relative pathnames). A simpler way to handle saving and restoring the working directory when it may be deeper than {PATH\_MAX} bytes in the file hierarchy is to use a file descriptor and *fchdir()*, rather than *getcwd()* and *chdir()*. However, the two methods do have some differences. The *fchdir()* approach causes the program to restore a working directory even if it has been renamed in the meantime, whereas the *chdir()* approach restores to a directory with the same name as the original, even if the directories were renamed in the meantime. Since the *fchdir()* approach does not access parent directories, it can succeed when *getcwd()* would fail due to permissions problems. In applications conforming to earlier versions of this standard, it was not possible to use the *fchdir()* approach when the working directory is searchable but not readable, as the only way to open a directory was with O\_RDONLY, whereas the *getcwd()* approach can succeed in this case.

#### RATIONALE

Having *getcwd()* take no arguments and instead use the *malloc()* function to produce space for the returned argument was considered. The advantage is that *getcwd()* knows how big the working directory pathname is and can allocate an appropriate amount of space. But the programmer would have to use the *free()* function to free the resulting object, or each use of *getcwd()* would further reduce the available memory. Finally, *getcwd()* is taken from the SVID where it has the two arguments used in this volume of POSIX.1-2008.

The older function *getwd()* was rejected for use in this context because it had only a buffer argument and no *size* argument, and thus had no way to prevent overwriting the buffer, except to depend on the programmer to provide a large enough buffer.

On some implementations, if *buf* is a null pointer, *getcwd()* may obtain *size* bytes of memory using *malloc()*. In this case, the pointer returned by *getcwd()* may be used as the argument in a subsequent call to *free()*. Invoking *getcwd()* with *buf* as a null pointer is not recommended in conforming applications.

Earlier implementations of `getcwd()` sometimes generated pathnames like `"../../../../subdirname"` internally, using them to explore the path of ancestor directories back to the root. If one of these internal pathnames exceeded `{PATH_MAX}` in length, the implementation could fail with `errno` set to `[ENAMETOOLONG]`. This is no longer allowed.

If a program is operating in a directory where some (grand)parent directory does not permit reading, `getcwd()` may fail, as in most implementations it must read the directory to determine the name of the file. This can occur if search, but not read, permission is granted in an intermediate directory, or if the program is placed in that directory by some more privileged process (for example, login). Including the `[EACCES]` error condition makes the reporting of the error consistent and warns the application developer that `getcwd()` can fail for reasons beyond the control of the application developer or user. Some implementations can avoid this occurrence (for example, by implementing `getcwd()` using `pwd`, where `pwd` is a set-user-root process), thus the error was made optional. Since this volume of POSIX.1-2008 permits the addition of other errors, this would be a common addition and yet one that applications could not be expected to deal with without this addition.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

`malloc()`

XBD `<unistd.h>`

#### CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

#### Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The `[ENOMEM]` optional error condition is added.

#### Issue 7

Austin Group Interpretation 1003.1-2001 #140 is applied, changing the text for consistency with the `pwd` utility, adding text to address the case where the current directory is deeper in the file hierarchy than `{PATH_MAX}` bytes, and adding the requirements relating to pathnames beginning with two `<slash>` characters.

**NAME**

getdate — convert user format date and time

**SYNOPSIS**

```
XSI    #include <time.h>
      struct tm *getdate(const char *string);
```

**DESCRIPTION**

The *getdate()* function shall convert a string representation of a date or time into a broken-down time.

The external variable or macro *getdate\_err*, which has type **int**, is used by *getdate()* to return error values. It is unspecified whether *getdate\_err* is a macro or an identifier declared with external linkage, and whether or not it is a modifiable lvalue. If a macro definition is suppressed in order to access an actual object, or a program defines an identifier with the name *getdate\_err*, the behavior is undefined.

Templates are used to parse and interpret the input string. The templates are contained in a text file identified by the environment variable *DATMSK*. The *DATMSK* variable should be set to indicate the full pathname of the file that contains the templates. The first line in the template that matches the input specification is used for interpretation and conversion into the internal time format.

The following conversion specifications shall be supported:

|    |                                                                                                                                      |
|----|--------------------------------------------------------------------------------------------------------------------------------------|
| %% | Equivalent to %.                                                                                                                     |
| %a | Abbreviated weekday name.                                                                                                            |
| %A | Full weekday name.                                                                                                                   |
| %b | Abbreviated month name.                                                                                                              |
| %B | Full month name.                                                                                                                     |
| %c | Locale's appropriate date and time representation.                                                                                   |
| %C | Century number [00,99]; leading zeros are permitted but not required.                                                                |
| %d | Day of month [01,31]; the leading 0 is optional.                                                                                     |
| %D | Date as %m/%d/%y.                                                                                                                    |
| %e | Equivalent to %d.                                                                                                                    |
| %h | Abbreviated month name.                                                                                                              |
| %H | Hour [00,23].                                                                                                                        |
| %I | Hour [01,12].                                                                                                                        |
| %m | Month number [01,12].                                                                                                                |
| %M | Minute [00,59].                                                                                                                      |
| %n | Equivalent to <newline>.                                                                                                             |
| %p | Locale's equivalent of either AM or PM.                                                                                              |
| %r | The locale's appropriate representation of time in AM and PM notation. In the POSIX locale, this shall be equivalent to %I:%M:%S %p. |

|       |    |                                                                                                                                                                                                                                                                                                                                            |
|-------|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 34652 | %R | Time as %H:%M.                                                                                                                                                                                                                                                                                                                             |
| 34653 | %S | Seconds [00,60]. The range goes to 60 (rather than stopping at 59) to allow positive leap seconds to be expressed. Since leap seconds cannot be predicted by any algorithm, leap second data must come from some external source.                                                                                                          |
| 34654 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34655 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34656 | %t | Equivalent to <tab>.                                                                                                                                                                                                                                                                                                                       |
| 34657 | %T | Time as %H:%M:%S.                                                                                                                                                                                                                                                                                                                          |
| 34658 | %w | Weekday number (Sunday = [0,6]).                                                                                                                                                                                                                                                                                                           |
| 34659 | %x | Locale's appropriate date representation.                                                                                                                                                                                                                                                                                                  |
| 34660 | %X | Locale's appropriate time representation.                                                                                                                                                                                                                                                                                                  |
| 34661 | %y | Year within century. When a century is not otherwise specified, values in the range [69,99] shall refer to years 1969 to 1999 inclusive, and values in the range [00,68] shall refer to years 2000 to 2068 inclusive.                                                                                                                      |
| 34662 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34663 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34664 |    | <b>Note:</b> It is expected that in a future version of this standard the default century inferred from a 2-digit year will change. (This would apply to all commands accepting a 2-digit year as input.)                                                                                                                                  |
| 34665 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34666 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34667 | %Y | Year as "ccyy" (for example, 2001).                                                                                                                                                                                                                                                                                                        |
| 34668 | %Z | Timezone name or no characters if no timezone exists. If the timezone supplied by %Z is not the timezone that <i>getdate()</i> expects, an invalid input specification error shall result. The <i>getdate()</i> function calculates an expected timezone based on information supplied to the function (such as the hour, day, and month). |
| 34669 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34670 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34671 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34672 |    | The match between the template and input specification performed by <i>getdate()</i> shall be case-insensitive.                                                                                                                                                                                                                            |
| 34673 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34674 |    | The month and weekday names can consist of any combination of upper and lowercase letters. The process can request that the input date or time specification be in a specific language by setting the <i>LC_TIME</i> category (see <i>setlocale()</i> ).                                                                                   |
| 34675 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34676 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34677 |    | Leading zeros are not necessary for the descriptors that allow leading zeros. However, at most two digits are allowed for those descriptors, including leading zeros. Extra white space in either the template file or in <i>string</i> shall be ignored.                                                                                  |
| 34678 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34679 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34680 |    | The results are undefined if the conversion specifications %c, %x, and %X include unsupported conversion specifications.                                                                                                                                                                                                                   |
| 34681 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34682 |    | The following rules apply for converting the input specification into the internal format:                                                                                                                                                                                                                                                 |
| 34683 |    | • If %Z is being scanned, then <i>getdate()</i> shall initialize the broken-down time to be the current time in the scanned timezone. Otherwise, it shall initialize the broken-down time based on the current local time as if <i>localtime()</i> had been called.                                                                        |
| 34684 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34685 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34686 |    | • If only the weekday is given, the day chosen shall be the day, starting with today and moving into the future, which first matches the named day.                                                                                                                                                                                        |
| 34687 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34688 |    | • If only the month (and no year) is given, the month chosen shall be the month, starting with the current month and moving into the future, which first matches the named month. The first day of the month shall be assumed if no day is given.                                                                                          |
| 34689 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34690 |    |                                                                                                                                                                                                                                                                                                                                            |
| 34691 |    | • If no hour, minute, and second are given, the current hour, minute, and second shall be assumed.                                                                                                                                                                                                                                         |
| 34692 |    |                                                                                                                                                                                                                                                                                                                                            |

- If no date is given, the hour chosen shall be the hour, starting with the current hour and moving into the future, which first matches the named hour.

If a conversion specification in the DATEMSK file does not correspond to one of the conversion specifications above, the behavior is unspecified.

The `getdate()` function need not be thread-safe.

## RETURN VALUE

Upon successful completion, `getdate()` shall return a pointer to a **struct tm**. Otherwise, it shall return a null pointer and set `getdate_err` to indicate the error.

## ERRORS

The `getdate()` function shall fail in the following cases, setting `getdate_err` to the value shown in the list below. Any changes to `errno` are unspecified.

1. The `DATEMSK` environment variable is null or undefined.
2. The template file cannot be opened for reading.
3. Failed to get file status information.
4. The template file is not a regular file.
5. An I/O error is encountered while reading the template file.
6. Memory allocation failed (not enough memory available).
7. There is no line in the template that matches the input.
8. Invalid input specification. For example, February 31; or a time is specified that cannot be represented in a **time\_t** (representing the time in seconds since the Epoch).

## EXAMPLES

1. The following example shows the possible contents of a template:
 

```
%m
%A %B %d, %Y, %H:%M:%S
%A
%B
%m/%d/%y %I %p
%d,%m,%Y %H:%M
at %A the %dst of %B in %Y
run job at %I %p,%B %dnd
%A den %d. %B %Y %H.%M Uhr
```
2. The following are examples of valid input specifications for the template in Example 1:

```
getdate("10/1/87 4 PM");
getdate("Friday");
getdate("Friday September 18, 1987, 10:30:30");
getdate("24,9,1986 10:30");
getdate("at monday the 1st of december in 1986");
getdate("run job at 3 PM, december 2nd");
```

If the `LC_TIME` category is set to a German locale that includes *freitag* as a weekday name and *oktober* as a month name, the following would be valid:

```
getdate("freitag den 10. oktober 1986 10.30 Uhr");
```

3. The following example shows how local date and time specification can be defined in the template:

| Invocation                 | Line in Template |
|----------------------------|------------------|
| getdate("11/27/86")        | %m/%d/%y         |
| getdate("27.11.86")        | %d.%m.%y         |
| getdate("86-11-27")        | %y-%m-%d         |
| getdate("Friday 12:00:00") | %A %H:%M:%S      |

4. The following examples help to illustrate the above rules assuming that the current date is Mon Sep 22 12:19:47 EDT 1986 and the *LC\_TIME* category is set to the default C or POSIX locale:

| Input        | Line in Template | Date                         |
|--------------|------------------|------------------------------|
| Mon          | %a               | Mon Sep 22 12:19:47 EDT 1986 |
| Sun          | %a               | Sun Sep 28 12:19:47 EDT 1986 |
| Fri          | %a               | Fri Sep 26 12:19:47 EDT 1986 |
| September    | %B               | Mon Sep 1 12:19:47 EDT 1986  |
| January      | %B               | Thu Jan 1 12:19:47 EST 1987  |
| December     | %B               | Mon Dec 1 12:19:47 EST 1986  |
| Sep Mon      | %b %a            | Mon Sep 1 12:19:47 EDT 1986  |
| Jan Fri      | %b %a            | Fri Jan 2 12:19:47 EST 1987  |
| Dec Mon      | %b %a            | Mon Dec 1 12:19:47 EST 1986  |
| Jan Wed 1989 | %b %a %Y         | Wed Jan 4 12:19:47 EST 1989  |
| Fri 9        | %a %H            | Fri Sep 26 09:00:00 EDT 1986 |
| Feb 10:30    | %b %H:%S         | Sun Feb 1 10:00:30 EST 1987  |
| 10:30        | %H:%M            | Tue Sep 23 10:30:00 EDT 1986 |
| 13:30        | %H:%M            | Mon Sep 22 13:30:00 EDT 1986 |

## APPLICATION USAGE

Although historical versions of *getdate()* did not require that **<time.h>** declare the external variable *getdate\_err*, this volume of POSIX.1-2008 does require it. The standard developers encourage applications to remove declarations of *getdate\_err* and instead incorporate the declaration by including **<time.h>**.

Applications should use %Y (4-digit years) in preference to %y (2-digit years).

## RATIONALE

In standard locales, the conversion specifications %c, %x, and %X do not include unsupported conversion specifiers and so the text regarding results being undefined is not a problem in that case.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*ctime()*, *localtime()*, *setlocale()*, *strftime()*, *times()*

XBD **<time.h>**

## CHANGE HISTORY

First released in Issue 4, Version 2.



**Issue 5**

- 34776 Moved from X/OPEN UNIX extension to BASE.
- 34777
- 34778 The last paragraph of the DESCRIPTION is added.
- 34779 The %C conversion specification is added, and the exact meaning of the %y conversion
- 34780 specification is clarified in the DESCRIPTION.
- 34781 A note indicating that this function need not be reentrant is added to the DESCRIPTION.
- 34782 The %R conversion specification is changed to follow historical practice.

**Issue 6**

- 34783 The DESCRIPTION is updated to refer to “seconds since the Epoch” rather than “seconds since
- 34784 00:00:00 UTC (Coordinated Universal Time), January 1 1970” for consistency with other *time*
- 34785 functions.
- 34786
- 34787 The description of %S is updated so that the valid range is [00,60] rather than [00,61].
- 34788 The DESCRIPTION is updated to refer to conversion specifications instead of field descriptors
- 34789 for consistency with other functions.

**Issue 7**

- 34790 Austin Group Interpretation 1003.1-2001 #156 is applied.
- 34791
- 34792 The description of the *getdate\_err* value is expanded.



**NAME**

getdelim, getline — read a delimited record from *stream*

**SYNOPSIS**

```
CX #include <stdio.h>

ssize_t getdelim(char **restrict lineptr, size_t *restrict n,
    int delimiter, FILE *restrict stream);
ssize_t getline(char **restrict lineptr, size_t *restrict n,
    FILE *restrict stream);
```

**DESCRIPTION**

The *getdelim()* function shall read from *stream* until it encounters a character matching the *delimiter* character. The *delimiter* argument is an **int**, the value of which the application shall ensure is a character representable as an **unsigned char** of equal value that terminates the read process. If the *delimiter* argument has any other value, the behavior is undefined.

The application shall ensure that *\*lineptr* is a valid argument that could be passed to the *free()* function. If *\*n* is non-zero, the application shall ensure that *\*lineptr* either points to an object of size at least *\*n* bytes, or is a null pointer.

If *\*lineptr* is a null pointer or if the object pointed to by *\*lineptr* is of insufficient size, an object shall be allocated as if by *malloc()* or the object shall be reallocated as if by *realloc()*, respectively, such that the object is large enough to hold the characters to be written to it, including the terminating NUL, and *\*n* shall be set to the new size. If the object was allocated, or if the reallocation operation moved the object, *\*lineptr* shall be updated to point to the new object or new location. The characters read, including any delimiter, shall be stored in the object, and a terminating NUL added when the delimiter or end-of-file is encountered.

The *getline()* function shall be equivalent to the *getdelim()* function with the *delimiter* character equal to the <newline> character.

The *getdelim()* and *getline()* functions may mark the last data access timestamp of the file associated with *stream* for update. The last data access timestamp shall be marked for update by the first successful execution of *fgetc()*, *fgets()*, *fread()*, *fscanf()*, *getc()*, *getchar()*, *getdelim()*, *getline()*, *gets()*, or *scanf()* using *stream* that returns data not supplied by a prior call to *ungetc()*.

**RETURN VALUE**

Upon successful completion, the *getline()* and *getdelim()* functions shall return the number of bytes written into the buffer, including the delimiter character if one was encountered before EOF, but excluding the terminating NUL character. If the end-of-file indicator for the stream is set, or if no characters were read and the stream is at end-of-file, the end-of-file indicator for the stream shall be set and the function shall return -1. If an error occurs, the error indicator for the stream shall be set, and the function shall return -1 and set *errno* to indicate the error.

**ERRORS**

For the conditions under which the *getdelim()* and *getline()* functions shall fail and may fail, refer to *fgetc()*.

In addition, these functions shall fail if:

[EINVAL] *lineptr* or *n* is a null pointer.

[ENOMEM] Insufficient memory is available.

These functions may fail if:

34836 [EOVERFLOW] The number of bytes to be written into the buffer, including the delimiter |  
34837 character (if encountered), would exceed {SSIZE\_MAX}.

## 34838 EXAMPLES

```
34839 #include <stdio.h>
34840 #include <stdlib.h>
34841 int main(void)
34842 {
34843     FILE *fp;
34844     char *line = NULL;
34845     size_t len = 0;
34846     ssize_t read;
34847     fp = fopen("/etc/motd", "r");
34848     if (fp == NULL)
34849         exit(1);
34850     while ((read = getline(&line, &len, fp)) != -1) {
34851         printf("Retrieved line of length %zu :\\n", read);
34852         printf("%s", line);
34853     }
34854     if (ferror(fp)) {
34855         /* handle error */
34856     }
34857     free(line);
34858     fclose(fp);
34859     return 0;
34860 }
```

## 34861 APPLICATION USAGE

34862 Setting *\*lineptr* to a null pointer and *\*n* to zero are allowed and a recommended way to start  
34863 parsing a file.

34864 The *ferror()* or *feof()* functions should be used to distinguish between an error condition and an  
34865 end-of-file condition.

34866 Although a NUL terminator is always supplied after the line, note that *strlen(\*lineptr)* will be  
34867 smaller than the return value if the line contains embedded NUL characters.

## 34868 RATIONALE

34869 These functions are widely used to solve the problem that the *fgets()* function has with long  
34870 lines. The functions automatically enlarge the target buffers if needed. These are especially  
34871 useful since they reduce code needed for applications.

## 34872 FUTURE DIRECTIONS

34873 None.

## 34874 SEE ALSO

34875 [Section 2.5](#) (on page 495), [fgetc\(\)](#), [fgets\(\)](#), [free\(\)](#), [malloc\(\)](#), [realloc\(\)](#)

34876 XBD [<stdio.h>](#)

## 34877 CHANGE HISTORY

34878 First released in Issue 7.

34879 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0237 [14] is applied.

34880 **NAME**

34881       getegid — get the effective group ID

34882 **SYNOPSIS**

34883       #include &lt;unistd.h&gt;

34884       gid\_t getegid(void);

34885 **DESCRIPTION**

34886       The *getegid()* function shall return the effective group ID of the calling process. The *getegid()* +  
 34887       function shall not modify *errno*.

34888 **RETURN VALUE**

34889       The *getegid()* function shall always be successful and no return value is reserved to indicate an  
 34890       error.

34891 **ERRORS**

34892       No errors are defined.

34893 **EXAMPLES**

34894       None.

34895 **APPLICATION USAGE**

34896       None.

34897 **RATIONALE**

34898       In a conforming environment, *getegid()* will always succeed. It is possible for implementations to |  
 34899       provide an extension where a process in a non-conforming environment will not be associated |  
 34900       with a user or group ID. It is recommended that such implementations return (**gid\_t**)–1 and set |  
 34901       *errno* to indicate such an environment; doing so does not violate this standard, since such an |  
 34902       environment is already an extension.

34903 **FUTURE DIRECTIONS**

34904       None.

34905 **SEE ALSO**34906       *geteuid()*, *getgid()*, *getuid()*, *setegid()*, *seteuid()*, *setgid()*, *setregid()*, *setreuid()*, *setuid()*

34907       XBD &lt;sys/types.h&gt;, &lt;unistd.h&gt;

34908 **CHANGE HISTORY**

34909       First released in Issue 1. Derived from Issue 1 of the SVID.

34910 **Issue 6**

34911       In the SYNOPSIS, the optional include of the &lt;sys/types.h&gt; header is removed.

34912       The following new requirements on POSIX implementations derive from alignment with the  
 34913       Single UNIX Specification:

- 34914       • The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was  
 34915       required for conforming implementations of previous POSIX specifications, it was not  
 34916       required for UNIX applications.

34917 **NAME**

34918        getenv — get value of an environment variable

34919 **SYNOPSIS**

34920        #include &lt;stdlib.h&gt;

34921        char \*getenv(const char \*name);

34922 **DESCRIPTION**

34923 CX        The functionality described on this reference page is aligned with the ISO C standard. Any  
 34924        conflict between the requirements described here and the ISO C standard is unintentional. This  
 34925        volume of POSIX.1-2008 defers to the ISO C standard.

34926        The *getenv()* function shall search the environment of the calling process (see XBD Chapter 8, on  
 34927        page 173) for the environment variable *name* if it exists and return a pointer to the value of the  
 34928        environment variable. If the specified environment variable cannot be found, a null pointer shall  
 34929        be returned. The application shall ensure that it does not modify the string pointed to by the  
 34930        *getenv()* function.

34931 CX        The returned string pointer might be invalidated or the string content might be overwritten by a  
 34932 CX        subsequent call to *getenv()*, *setenv()*, *unsetenv()*,  
 34933 XSI        or (if supported) *putenv()* but they shall not be affected by a call to any other function in this  
 34934        volume of POSIX.1-2008.

34935 CX        The returned string pointer might also be invalidated if the calling thread is terminated. +

34936 CX        The *getenv()* function need not be thread-safe.

34937 **RETURN VALUE**

34938        Upon successful completion, *getenv()* shall return a pointer to a string containing the *value* for  
 34939        the specified *name*. If the specified *name* cannot be found in the environment of the calling  
 34940        process, a null pointer shall be returned.

34941 **ERRORS**

34942        No errors are defined.

34943 **EXAMPLES**34944        **Getting the Value of an Environment Variable**34945        The following example gets the value of the *HOME* environment variable.

34946        #include &lt;stdlib.h&gt;

34947        ...

34948        const char \*name = "HOME";

34949        char \*value;

34950        value = getenv(name);

34951 **APPLICATION USAGE**

34952        None.

34953 **RATIONALE**

34954        The *clearenv()* function was considered but rejected. The *putenv()* function has now been  
 34955        included for alignment with the Single UNIX Specification.

34956        The *getenv()* function is inherently not thread-safe because it returns a value pointing to static  
 34957        data.

34958        Conforming applications are required not to directly modify the pointers to which *environ*  
 34959        points, but to use only the *setenv()*, *unsetenv()*, and *putenv()* functions, or assignment to *environ*

itself, to manipulate the process environment. This constraint allows the implementation to properly manage the memory it allocates. This enables the implementation to free any space it has allocated to strings (and perhaps the pointers to them) stored in *environ* when *unsetenv()* is called. A C runtime start-up procedure (that which invokes *main()* and perhaps initializes *environ*) can also initialize a flag indicating that none of the environment has yet been copied to allocated storage, or that the separate table has not yet been initialized. If the application switches to a complete new environment by assigning a new value to *environ*, this can be detected by *getenv()*, *setenv()*, *unsetenv()*, or *putenv()* and the implementation can at that point reinitialize based on the new environment. (This may include copying the environment strings into a new array and assigning *environ* to point to it.)

In fact, for higher performance of *getenv()*, implementations that do not provide *putenv()* could also maintain a separate copy of the environment in a data structure that could be searched much more quickly (such as an indexed hash table, or a binary tree), and update both it and the linear list at *environ* when *setenv()* or *unsetenv()* is invoked. On implementations that do provide *putenv()*, such a copy might still be worthwhile but would need to allow for the fact that applications can directly modify the content of environment strings added with *putenv()*. For example, if an environment string found by searching the copy is one that was added using *putenv()*, the implementation would need to check that the string in *environ* still has the same name (and value, if the copy includes values), and whenever searching the copy produces no match the implementation would then need to search each environment string in *environ* that was added using *putenv()* in case any of them have changed their names and now match. Thus, each use of *putenv()* to add to the environment would reduce the speed advantage of having the copy.

Performance of *getenv()* can be important for applications which have large numbers of environment variables. Typically, applications like this use the environment as a resource database of user-configurable parameters. The fact that these variables are in the user's shell environment usually means that any other program that uses environment variables (such as *ls*, which attempts to use *COLUMNS*), or really almost any utility (*LANG*, *LC\_ALL*, and so on) is similarly slowed down by the linear search through the variables.

An implementation that maintains separate data structures, or even one that manages the memory it consumes, is not currently required as it was thought it would reduce consensus among implementors who do not want to change their historical implementations.

#### FUTURE DIRECTIONS

A future version may add one or more functions to access and modify the environment in a thread-safe manner.

#### SEE ALSO

*exec*, *putenv()*, *setenv()*, *unsetenv()*

XBD Chapter 8 (on page 173), *<stdlib.h>*

#### CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

#### Issue 5

Normative text previously in the APPLICATION USAGE section is moved to the RETURN VALUE section.

A note indicating that this function need not be reentrant is added to the DESCRIPTION.

35004 **Issue 6**

35005 The following changes were made to align with the IEEE P1003.1a draft standard:

- 35006
  - References added to the new *setenv()* and *unsetenv()* functions.

35007 The normative text is updated to avoid use of the term “must” for application requirements.

35008 **Issue 7**

35009 Austin Group Interpretation 1003.1-2001 #062 is applied, clarifying that a call to *putenv()* may  
35010 also cause the string to be overwritten.

35011 Austin Group Interpretation 1003.1-2001 #148 is applied, adding the FUTURE DIRECTIONS.

35012 Austin Group Interpretation 1003.1-2001 #156 is applied.

35013 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0238 [75,428], XSH/TC1-2008/0239  
35014 [167], and XSH/TC1-2008/0240 [167] are applied.

**35015 NAME**

35016       geteuid — get the effective user ID

**35017 SYNOPSIS**

35018       #include <unistd.h>

35019       uid\_t geteuid(void);

**35020 DESCRIPTION**

35021       The *geteuid()* function shall return the effective user ID of the calling process. The *geteuid()* +  
35022       function shall not modify *errno*.

**35023 RETURN VALUE**

35024       The *geteuid()* function shall always be successful and no return value is reserved to indicate an  
35025       error.

**35026 ERRORS**

35027       No errors are defined.

**35028 EXAMPLES**

35029       None.

**35030 APPLICATION USAGE**

35031       None.

**35032 RATIONALE**

35033       In a conforming environment, *geteuid()* will always succeed. It is possible for implementations |  
35034       to provide an extension where a process in a non-conforming environment will not be associated |  
35035       with a user or group ID. It is recommended that such implementations return (**uid\_t**)–1 and set |  
35036       *errno* to indicate such an environment; doing so does not violate this standard, since such an |  
35037       environment is already an extension.

**35038 FUTURE DIRECTIONS**

35039       None.

**35040 SEE ALSO**

35041       *getegid()*, *getgid()*, *getuid()*, *setegid()*, *seteuid()*, *setgid()*, *setregid()*, *setreuid()*, *setuid()*

35042       XBD <sys/types.h>, <unistd.h>

**35043 CHANGE HISTORY**

35044       First released in Issue 1. Derived from Issue 1 of the SVID.

**35045 Issue 6**

35046       In the SYNOPSIS, the optional include of the <sys/types.h> header is removed.

35047       The following new requirements on POSIX implementations derive from alignment with the  
35048       Single UNIX Specification:

- 35049       • The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was  
35050       required for conforming implementations of previous POSIX specifications, it was not  
35051       required for UNIX applications.

**NAME**

getgid — get the real group ID

**SYNOPSIS**

```
#include <unistd.h>
```

```
gid_t getgid(void);
```

**DESCRIPTION**

The *getgid()* function shall return the real group ID of the calling process. The *getgid()* function shall not modify *errno*.

**RETURN VALUE**

The *getgid()* function shall always be successful and no return value is reserved to indicate an error.

**ERRORS**

No errors are defined.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

In a conforming environment, *getgid()* will always succeed. It is possible for implementations to provide an extension where a process in a non-conforming environment will not be associated with a user or group ID. It is recommended that such implementations return (**gid\_t**)–1 and set *errno* to indicate such an environment; doing so does not violate this standard, since such an environment is already an extension.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*getegid()*, *geteuid()*, *getuid()*, *setegid()*, *seteuid()*, *setgid()*, *setregid()*, *setreuid()*, *setuid()*

XBD [\*\*<sys/types.h>\*\*](#), [\*\*<unistd.h>\*\*](#)

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 6**

In the SYNOPSIS, the optional include of the **<sys/types.h>** header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include **<sys/types.h>** has been removed. Although **<sys/types.h>** was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.



35089 **NAME**

35090       getgrent — get the group database entry

35091 **SYNOPSIS**

```
35092 XSI      #include <grp.h>
35093          struct group *getgrent(void);
```

35094 **DESCRIPTION**35095       Refer to *endgrent()*.

**NAME**

getgrgid, getgrgid\_r — get group database entry for a group ID

**SYNOPSIS**

```
#include <grp.h>

struct group *getgrgid(gid_t gid);
int getgrgid_r(gid_t gid, struct group *grp, char *buffer,
               size_t bufsz, struct group **result);
```

**DESCRIPTION**

The *getgrgid()* function shall search the group database for an entry with a matching *gid*.

The *getgrgid()* function need not be thread-safe.

Applications wishing to check for error situations should set *errno* to 0 before calling *getgrgid()*.  
If *getgrgid()* returns a null pointer and *errno* is set to non-zero, an error occurred.

The *getgrgid\_r()* function shall update the **group** structure pointed to by *grp* and store a pointer to that structure at the location pointed to by *result*. The structure shall contain an entry from the group database with a matching *gid*. Storage referenced by the group structure is allocated from the memory provided with the *buffer* parameter, which is *bufsz* bytes in size. A call to *sysconf(\_SC\_GETGR\_R\_SIZE\_MAX)* returns either -1 without changing *errno* or an initial value suggested for the size of this buffer. A null pointer shall be returned at the location pointed to by *result* on error or if the requested entry is not found.

**RETURN VALUE**

Upon successful completion, *getgrgid()* shall return a pointer to a **struct group** with the structure defined in **<grp.h>** with a matching entry if one is found. The *getgrgid()* function shall return a null pointer if either the requested entry was not found, or an error occurred. If the requested entry was not found, *errno* shall not be changed. On error, *errno* shall be set to indicate the error.

The application shall not modify the structure to which the return value points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *getgrent()*, *getgrgid()*, or *getgrnam()*. The returned pointer, and pointers within the structure, might also be invalidated if the calling thread is terminated.

If successful, the *getgrgid\_r()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

**ERRORS**

The *getgrgid()* and *getgrgid\_r()* functions may fail if:

- [EIO] An I/O error has occurred.
- [EINTR] A signal was caught during *getgrgid()*.
- [EMFILE] All file descriptors available to the process are currently open.
- [ENFILE] The maximum allowable number of files is currently open in the system.

The *getgrgid\_r()* function may fail if:

- [ERANGE] Insufficient storage was supplied via *buffer* and *bufsz* to contain the data to be referenced by the resulting **group** structure.

**EXAMPLES**

Note that `sysconf(_SC_GETGR_R_SIZE_MAX)` may return `-1` if there is no hard limit on the size of the buffer needed to store all the groups returned. This example shows how an application can allocate a buffer of sufficient size to work with `getgrgid_r()`.

```

long int initlen = sysconf(_SC_GETGR_R_SIZE_MAX);
size_t len;
if (initlen == -1)
    /* Default initial length. */
    len = 1024;
else
    len = (size_t) initlen;
struct group result;
struct group *resultp;
char *buffer = malloc(len);
if (buffer == NULL)
    ...handle error...
int e;
while ((e = getgrgid_r(42, &result, buffer, len, &resultp)) == ERANGE)
{
    size_t newlen = 2 * len;
    if (newlen < len)
        ...handle error...
    len = newlen;
    char *newbuffer = realloc(buffer, len);
    if (newbuffer == NULL)
        ...handle error...
    buffer = newbuffer;
}
if (e != 0)
    ...handle error...
free (buffer);

```

**Finding an Entry in the Group Database**

The following example uses `getgrgid()` to search the group database for a group ID that was previously stored in a `stat` structure, then prints out the group name if it is found. If the group is not found, the program prints the numeric value of the group for the entry.

```

#include <sys/types.h>
#include <grp.h>
#include <stdio.h>
...
struct stat statbuf;
struct group *grp;
...
if ((grp = getgrgid(statbuf.st_gid)) != NULL)
    printf(" %-8s", grp->gr_name);
else
    printf(" %-8d", statbuf.st_gid);
...

```

**APPLICATION USAGE**

The *getgrgid\_r()* function is thread-safe and shall return values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

Portable applications should take into account that it is usual for an implementation to return -1 from *sysconf()* indicating that there is no maximum for *\_SC\_GETGR\_R\_SIZE\_MAX*.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*endgrent()*, *getgrnam()*, *sysconf()*

XBD *<grp.h>*, *<sys/types.h>*

**CHANGE HISTORY**

First released in Issue 1. Derived from System V Release 2.0.

**Issue 5**

Normative text previously in the APPLICATION USAGE section is moved to the RETURN VALUE section.

The *getgrgid\_r()* function is included for alignment with the POSIX Threads Extension.

A note indicating that the *getgrgid()* function need not be reentrant is added to the DESCRIPTION.

**Issue 6**

The *getgrgid\_r()* function is marked as part of the Thread-Safe Functions option.

The Open Group Corrigendum U028/3 is applied, correcting text in the DESCRIPTION describing matching the *gid*.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

In the SYNOPSIS, the optional include of the *<sys/types.h>* header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include *<sys/types.h>* has been removed. Although *<sys/types.h>* was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- In the RETURN VALUE section, the requirement to set *errno* on error is added.
- The [EIO], [EINTR], [EMFILE], and [ENFILE] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

IEEE PASC Interpretation 1003.1 #116 is applied, changing the description of the size of the buffer from *bufsize* characters to bytes.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #156 is applied.

SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

SD5-XSH-ERN-166 is applied.

- 35224 The `getgrgid_r()` function is moved from the Thread-Safe Functions option to the Base.
- 35225 A minor addition is made to the EXAMPLES section, reminding the application developer to
- 35226 free memory allocated as if by `malloc()`.
- 35227 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0241 [75] is applied.

**NAME**

getgrnam, getgrnam\_r — search group database for a name

**SYNOPSIS**

```
#include <grp.h>

struct group *getgrnam(const char *name);
int getgrnam_r(const char *name, struct group *grp, char *buffer,
               size_t bufsize, struct group **result);
```

**DESCRIPTION**

The *getgrnam()* function shall search the group database for an entry with a matching *name*.

The *getgrnam()* function need not be thread-safe.

Applications wishing to check for error situations should set *errno* to 0 before calling *getgrnam()*.  
If *getgrnam()* returns a null pointer and *errno* is set to non-zero, an error occurred.

The *getgrnam\_r()* function shall update the **group** structure pointed to by *grp* and store a pointer to that structure at the location pointed to by *result*. The structure shall contain an entry from the group database with a matching *name*. Storage referenced by the **group** structure is allocated from the memory provided with the *buffer* parameter, which is *bufsize* bytes in size. A call to *sysconf(\_SC\_GETGR\_R\_SIZE\_MAX)* returns either -1 without changing *errno* or an initial value suggested for the size of this buffer. A null pointer is returned at the location pointed to by *result* on error or if the requested entry is not found.

**RETURN VALUE**

The *getgrnam()* function shall return a pointer to a **struct group** with the structure defined in **<grp.h>** with a matching entry if one is found. The *getgrnam()* function shall return a null pointer if either the requested entry was not found, or an error occurred. On error, *errno* shall be set to indicate the error.

The application shall not modify the structure to which the return value points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *getgrent()*, *getgrgid()*, or *getgrnam()*. The returned pointer, and pointers within the structure, might also be invalidated if the calling thread is terminated.

The *getgrnam\_r()* function shall return zero on success or if the requested entry was not found and no error has occurred. If any error has occurred, an error number shall be returned to indicate the error.

**ERRORS**

The *getgrnam()* and *getgrnam\_r()* functions may fail if:

- [EIO] An I/O error has occurred.
- [EINTR] A signal was caught during *getgrnam()*.
- [EMFILE] All file descriptors available to the process are currently open.
- [ENFILE] The maximum allowable number of files is currently open in the system.

The *getgrnam\_r()* function may fail if:

- [ERANGE] Insufficient storage was supplied via *buffer* and *bufsize* to contain the data to be referenced by the resulting **group** structure.

**EXAMPLES**

Note that `sysconf(_SC_GETGR_R_SIZE_MAX)` may return `-1` if there is no hard limit on the size of the buffer needed to store all the groups returned. This example shows how an application can allocate a buffer of sufficient size to work with `getgrnam_r()`.

```

long int initlen = sysconf(_SC_GETGR_R_SIZE_MAX);
size_t len;
if (initlen == -1)
    /* Default initial length. */
    len = 1024;
else
    len = (size_t) initlen;
struct group result;
struct group *resultp;
char *buffer = malloc(len);
if (buffer == NULL)
    ...handle error...
int e;
while ((e = getgrnam_r("somegroup", &result, buffer, len, &resultp))
    == ERANGE)
{
    size_t newlen = 2 * len;
    if (newlen < len)
        ...handle error...
    len = newlen;
    char *newbuffer = realloc(buffer, len);
    if (newbuffer == NULL)
        ...handle error...
    buffer = newbuffer;
}
if (e != 0)
    ...handle error...
free (buffer);

```

**APPLICATION USAGE**

The `getgrnam_r()` function is thread-safe and shall return values in a user-supplied buffer instead - of possibly using a static data area that may be overwritten by each call.

Portable applications should take into account that it is usual for an implementation to return `-1` from `sysconf()` indicating that there is no maximum for `_SC_GETGR_R_SIZE_MAX`.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[\*endgrent\(\)\*](#), [\*getgrgid\(\)\*](#), [\*sysconf\(\)\*](#)

XBD [\*<grp.h>\*](#), [\*<sys/types.h>\*](#)

**CHANGE HISTORY**

First released in Issue 1. Derived from System V Release 2.0.

**Issue 5**

Normative text previously in the APPLICATION USAGE section is moved to the RETURN VALUE section.

The `getgrnam_r()` function is included for alignment with the POSIX Threads Extension.

A note indicating that the `getgrnam()` function need not be reentrant is added to the DESCRIPTION.

**Issue 6**

The `getgrnam_r()` function is marked as part of the Thread-Safe Functions option.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- In the RETURN VALUE section, the requirement to set `errno` on error is added.
- The [EIO], [EINTR], [EMFILE], and [ENFILE] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

IEEE PASC Interpretation 1003.1 #116 is applied, changing the description of the size of the buffer from *bufsize* characters to bytes.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #081 is applied, clarifying the RETURN VALUE section.

Austin Group Interpretation 1003.1-2001 #156 is applied.

SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

SD5-XSH-ERN-166 is applied.

The `getgrnam_r()` function is moved from the Thread-Safe Functions option to the Base.

A minor addition is made to the EXAMPLES section, reminding the application developer to free memory allocated as if by `malloc()`.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0242 [75] is applied.



35345 **NAME**35346 `getgroups` — get supplementary group IDs35347 **SYNOPSIS**35348 `#include <unistd.h>`35349 `int getgroups(int gidsetsize, gid_t grouplist[]);`35350 **DESCRIPTION**

35351 The `getgroups()` function shall fill in the array `grouplist` with the current supplementary group  
 35352 IDs of the calling process. It is implementation-defined whether `getgroups()` also returns the  
 35353 effective group ID in the `grouplist` array.

35354 The `gidsetsize` argument specifies the number of elements in the array `grouplist`. The actual  
 35355 number of group IDs stored in the array shall be returned. The values of array entries with  
 35356 indices greater than or equal to the value returned are undefined.

35357 If `gidsetsize` is 0, `getgroups()` shall return the number of group IDs that it would otherwise return  
 35358 without modifying the array pointed to by `grouplist`.

35359 If the effective group ID of the process is returned with the supplementary group IDs, the value  
 35360 returned shall always be greater than or equal to one and less than or equal to the value of  
 35361 `{NGROUPS_MAX}+1`.

35362 **RETURN VALUE**

35363 Upon successful completion, the number of supplementary group IDs shall be returned. A  
 35364 return value of `-1` indicates failure and `errno` shall be set to indicate the error.

35365 **ERRORS**35366 The `getgroups()` function shall fail if:

35367 `[EINVAL]` The `gidsetsize` argument is non-zero and less than the number of group IDs  
 35368 that would have been returned.

35369 **EXAMPLES**35370 **Getting the Supplementary Group IDs of the Calling Process**

35371 The following example places the current supplementary group IDs of the calling process into  
 35372 the `group` array.

35373 `#include <sys/types.h>`35374 `#include <unistd.h>`35375 `...`35376 `gid_t *group;`35377 `int nogroups;`35378 `long ngroups_max;`35379 `ngroups_max = sysconf(_SC_NGROUPS_MAX) + 1;`35380 `group = (gid_t *)malloc(ngroups_max * sizeof(gid_t));`35381 `ngroups = getgroups(ngroups_max, group);`35382 **APPLICATION USAGE**

35383 None.

35384 **RATIONALE**

35385 The related function `setgroups()` is a privileged operation and therefore is not covered by this  
 35386 volume of POSIX.1-2008.

35387 As implied by the definition of supplementary groups, the effective group ID may appear in the

array returned by *getgroups()* or it may be returned only by *getegid()*. Duplication may exist, but the application needs to call *getegid()* to be sure of getting all of the information. Various implementation variations and administrative sequences cause the set of groups appearing in the result of *getgroups()* to vary in order and as to whether the effective group ID is included, even when the set of groups is the same (in the mathematical sense of “set”). (The history of a process and its parents could affect the details of the result.)

Application developers should note that {NGROUPS\_MAX} is not necessarily a constant on all implementations.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*getegid()*, *setgid()*

XBD <sys/types.h>, <unistd.h>

#### CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

##### Issue 5

Normative text previously in the APPLICATION USAGE section is moved to the DESCRIPTION.

##### Issue 6

In the SYNOPSIS, the optional include of the <sys/types.h> header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- A return value of 0 is not permitted, because {NGROUPS\_MAX} cannot be 0. This is a FIPS requirement.

The following changes were made to align with the IEEE P1003.1a draft standard:

- An explanation is added that the effective group ID may be included in the supplementary group list.

35418 **NAME**

35419       gethostent — network host database functions

35420 **SYNOPSIS**

35421       #include &lt;netdb.h&gt;

35422       struct hostent \*gethostent(void);

35423 **DESCRIPTION**35424       Refer to *endhostent()*.

35425 **NAME**

35426       gethostid — get an identifier for the current host

35427 **SYNOPSIS**

```
35428 XSI      #include <unistd.h>
35429          long gethostid(void);
```

35430 **DESCRIPTION**35431       The *gethostid()* function shall retrieve a 32-bit identifier for the current host.35432 **RETURN VALUE**35433       Upon successful completion, *gethostid()* shall return an identifier for the current host.35434 **ERRORS**

35435       No errors are defined.

35436 **EXAMPLES**

35437       None.

35438 **APPLICATION USAGE**

35439       This volume of POSIX.1-2008 does not define the domain in which the return value is unique.

35440 **RATIONALE**

35441       None.

35442 **FUTURE DIRECTIONS**

35443       None.

35444 **SEE ALSO**35445       *initstate()*

35446       XBD &lt;unistd.h&gt;

35447 **CHANGE HISTORY**

35448       First released in Issue 4, Version 2.

35449 **Issue 5**

35450       Moved from X/OPEN UNIX extension to BASE.

35451 **NAME**

35452       gethostname — get name of current host

35453 **SYNOPSIS**

35454       #include &lt;unistd.h&gt;

35455       int gethostname(char \*name, size\_t namelen);

35456 **DESCRIPTION**

35457       The *gethostname()* function shall return the standard host name for the current machine. The  
35458       *namelen* argument shall specify the size of the array pointed to by the *name* argument. The  
35459       returned name shall be null-terminated, except that if *namelen* is an insufficient length to hold  
35460       the host name, then the returned name shall be truncated and it is unspecified whether the  
35461       returned name is null-terminated.

35462       Host names are limited to {HOST\_NAME\_MAX} bytes.

35463 **RETURN VALUE**

35464       Upon successful completion, 0 shall be returned; otherwise, -1 shall be returned.

35465 **ERRORS**

35466       No errors are defined.

35467 **EXAMPLES**

35468       None.

35469 **APPLICATION USAGE**

35470       None.

35471 **RATIONALE**

35472       None.

35473 **FUTURE DIRECTIONS**

35474       None.

35475 **SEE ALSO**35476       *gethostid()*, *uname()*

35477       XBD &lt;unistd.h&gt;

35478 **CHANGE HISTORY**

35479       First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

35480       The Open Group Base Resolution bwg2001-008 is applied, changing the *namelen* parameter from  
35481       *socklen\_t* to *size\_t*.

**NAME**

gettimer, setitimer — get and set value of interval timer

**SYNOPSIS**

OB XSI `#include <sys/time.h>`

```
int getitimer(int which, struct itimerval *value);
int setitimer(int which, const struct itimerval *restrict value,
              struct itimerval *restrict ovalue);
```

**DESCRIPTION**

The *getitimer()* function shall store the current value of the timer specified by *which* into the structure pointed to by *value*. The *setitimer()* function shall set the timer specified by *which* to the value specified in the structure pointed to by *value*, and if *ovalue* is not a null pointer, store the previous value of the timer in the structure pointed to by *ovalue*.

A timer value is defined by the **itimerval** structure, specified in **<sys/time.h>**. If *it\_value* is non-zero, it shall indicate the time to the next timer expiration. If *it\_interval* is non-zero, it shall specify a value to be used in reloading *it\_value* when the timer expires. Setting *it\_value* to 0 shall disable a timer, regardless of the value of *it\_interval*. Setting *it\_interval* to 0 shall disable a timer after its next expiration (assuming *it\_value* is non-zero).

Implementations may place limitations on the granularity of timer values. For each interval timer, if the requested timer value requires a finer granularity than the implementation supports, the actual timer value shall be rounded up to the next supported value.

An XSI-conforming implementation provides each process with at least three interval timers, which are indicated by the *which* argument:

|                |                                                                                                                                                                                                                                                                                        |
|----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ITIMER_PROF    | Decrements both in process virtual time and when the system is running on behalf of the process. It is designed to be used by interpreters in statistically profiling the execution of interpreted programs. Each time the ITIMER_PROF timer expires, the SIGPROF signal is delivered. |
| ITIMER_REAL    | Decrements in real time. A SIGALRM signal is delivered when this timer expires.                                                                                                                                                                                                        |
| ITIMER_VIRTUAL | Decrements in process virtual time. It runs only when the process is executing. A SIGVTALRM signal is delivered when it expires.                                                                                                                                                       |

The interaction between *setitimer()* and *alarm()* or *sleep()* is unspecified.

**RETURN VALUE**

Upon successful completion, *getitimer()* or *setitimer()* shall return 0; otherwise,  $-1$  shall be returned and *errno* set to indicate the error.

**ERRORS**

The *setitimer()* function shall fail if:

|          |                                                                                                                                                                                                        |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [EINVAL] | The <i>value</i> argument is not in canonical form. (In canonical form, the number of microseconds is a non-negative integer less than 1 000 000 and the number of seconds is a non-negative integer.) |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

The *getitimer()* and *setitimer()* functions may fail if:

|          |                                              |
|----------|----------------------------------------------|
| [EINVAL] | The <i>which</i> argument is not recognized. |
|----------|----------------------------------------------|

**35523 EXAMPLES**

35524 None.

**35525 APPLICATION USAGE**

35526 Applications should use the *timer\_gettime()* and *timer\_settime()* functions instead of the  
35527 obsolescent *getitimer()* and *setitimer()* functions, respectively.

**35528 RATIONALE**

35529 None.

**35530 FUTURE DIRECTIONS**

35531 The *getitimer()* and *setitimer()* functions may be removed in a future version.

**35532 SEE ALSO**

35533 *alarm()*, *exec*, *sleep()*, *timer\_getoverrun()*

35534 XBD <signal.h>, <sys/time.h>

**35535 CHANGE HISTORY**

35536 First released in Issue 4, Version 2.

**35537 Issue 5**

35538 Moved from X/OPEN UNIX extension to BASE.

**35539 Issue 6**

35540 The **restrict** keyword is added to the *setitimer()* prototype for alignment with the  
35541 ISO/IEC 9899:1999 standard.

**35542 Issue 7**

35543 The *getitimer()* and *setitimer()* functions are marked obsolescent.

35544 **NAME**35545       getline — read a delimited record from *stream*35546 **SYNOPSIS**

```
35547 CX      #include <stdio.h>
35548          ssize_t getline(char **restrict lineptr, size_t *restrict n,
35549                          FILE *restrict stream);
```

35550 **DESCRIPTION**35551       Refer to *getdelim()*.



**NAME**

getlogin, getlogin\_r — get login name

**SYNOPSIS**

```
#include <unistd.h>

char *getlogin(void);
int getlogin_r(char *name, size_t namesize);
```

**DESCRIPTION**

The *getlogin()* function shall return a pointer to a string containing the user name associated by the login activity with the controlling terminal of the current process. If *getlogin()* returns a non-null pointer, then that pointer points to the name that the user logged in under, even if there are several login names with the same user ID.

The *getlogin()* function need not be thread-safe.

The *getlogin\_r()* function shall put the name associated by the login activity with the controlling terminal of the current process in the character array pointed to by *name*. The array is *namesize* characters long and should have space for the name and the terminating null character. The maximum size of the login name is {LOGIN\_NAME\_MAX}.

If *getlogin\_r()* is successful, *name* points to the name the user used at login, even if there are several login names with the same user ID.

The *getlogin()* and *getlogin\_r()* functions may make use of file descriptors 0, 1, and 2 to find the controlling terminal of the current process, examining each in turn until the terminal is found. If in this case none of these three file descriptors is open to the controlling terminal, these functions may fail. The method used to find the terminal associated with a file descriptor may depend on the file descriptor being open to the actual terminal device, not */dev/tty*.

**RETURN VALUE**

Upon successful completion, *getlogin()* shall return a pointer to the login name or a null pointer if the user's login name cannot be found. Otherwise, it shall return a null pointer and set *errno* to indicate the error.

The application shall not modify the string returned. The returned pointer might be invalidated or the string content might be overwritten by a subsequent call to *getlogin()*. The returned pointer and the string content might also be invalidated if the calling thread is terminated. +

If successful, the *getlogin\_r()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

**ERRORS**

These functions may fail if:

- |          |                                                                                                     |
|----------|-----------------------------------------------------------------------------------------------------|
| [EMFILE] | All file descriptors available to the process are currently open.                                   |
| [ENFILE] | The maximum allowable number of files is currently open in the system.                              |
| [ENOTTY] | None of the file descriptors 0, 1, or 2 is open to the controlling terminal of the current process. |
| [ENXIO]  | The calling process has no controlling terminal.                                                    |
- The *getlogin\_r()* function may fail if:
- |          |                                                                                                                                |
|----------|--------------------------------------------------------------------------------------------------------------------------------|
| [ERANGE] | The value of <i>namesize</i> is smaller than the length of the string to be returned including the terminating null character. |
|----------|--------------------------------------------------------------------------------------------------------------------------------|

**EXAMPLES****Getting the User Login Name**

The following example calls the *getlogin()* function to obtain the name of the user associated with the calling process, and passes this information to the *getpwnam()* function to get the associated user database information.

```
#include <unistd.h>
#include <sys/types.h>
#include <pwd.h>
#include <stdio.h>
...
char *lgn;
struct passwd *pw;
...
if ((lgn = getlogin()) == NULL || (pw = getpwnam(lgn)) == NULL) {
    fprintf(stderr, "Get of user information failed.\n"); exit(1);
}
```

**APPLICATION USAGE**

Three names associated with the current process can be determined: *getpwuid(geteuid())* shall return the name associated with the effective user ID of the process; *getlogin()* shall return the name associated with the current login activity; and *getpwuid(getuid())* shall return the name associated with the real user ID of the process.

The *getlogin\_r()* function is thread-safe and returns values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

**RATIONALE**

The *getlogin()* function returns a pointer to the user's login name. The same user ID may be shared by several login names. If it is desired to get the user database entry that is used during login, the result of *getlogin()* should be used to provide the argument to the *getpwnam()* function. (This might be used to determine the user's login shell, particularly where a single user has multiple login shells with distinct login names, but the same user ID.)

The information provided by the *cuserid()* function, which was originally defined in the POSIX.1-1988 standard and subsequently removed, can be obtained by the following:

```
getpwuid(geteuid())
```

while the information provided by historical implementations of *cuserid()* can be obtained by:

```
getpwuid(getuid())
```

The thread-safe version of this function places the user name in a user-supplied buffer and returns a non-zero value if it fails. The non-thread-safe version may return the name in a static data area that may be overwritten by each call.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*getpwnam()*, *getpwuid()*, *geteuid()*, *getuid()*

XBD *<limits.h>*, *<unistd.h>*

**CHANGE HISTORY**

First released in Issue 1. Derived from System V Release 2.0.

**Issue 5**

Normative text previously in the APPLICATION USAGE section is moved to the RETURN VALUE section.

The `getlogin_r()` function is included for alignment with the POSIX Threads Extension.

A note indicating that the `getlogin()` function need not be reentrant is added to the DESCRIPTION.

**Issue 6**

The `getlogin_r()` function is marked as part of the Thread-Safe Functions option.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE section, the requirement to set *errno* on error is added.
- The [EMFILE], [ENFILE], and [ENXIO] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #156 is applied.

SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

The `getlogin_r()` function is moved from the Thread-Safe Functions option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0243 [172], XSH/TC1-2008/0244 [75], and XSH/TC1-2008/0245 [172] are applied.

**NAME**

getmsg, getpmsg — receive next message from a STREAMS file (**STREAMS**)

**SYNOPSIS**

```
#include <stropts.h>

int getmsg(int fildes, struct strbuf *restrict ctlptr,
           struct strbuf *restrict dataptr, int *restrict flagsp);
int getpmsg(int fildes, struct strbuf *restrict ctlptr,
            struct strbuf *restrict dataptr, int *restrict bandp,
            int *restrict flagsp);
```

**DESCRIPTION**

The `getmsg()` function shall retrieve the contents of a message located at the head of the STREAM head read queue associated with a STREAMS file and place the contents into one or more buffers. The message contains either a data part, a control part, or both. The data and control parts of the message shall be placed into separate buffers, as described below. The semantics of each part are defined by the originator of the message.

The `getpmsg()` function shall be equivalent to `getmsg()`, except that it provides finer control over the priority of the messages received. Except where noted, all requirements on `getmsg()` also pertain to `getpmsg()`.

The `fildes` argument specifies a file descriptor referencing a STREAMS-based file.

The `ctlptr` and `dataptr` arguments each point to a **strbuf** structure, in which the `buf` member points to a buffer in which the data or control information is to be placed, and the `maxlen` member indicates the maximum number of bytes this buffer can hold. On return, the `len` member shall contain the number of bytes of data or control information actually received. The `len` member shall be set to 0 if there is a zero-length control or data part and `len` shall be set to -1 if no data or control information is present in the message.

When `getmsg()` is called, `flagsp` should point to an integer that indicates the type of message the process is able to receive. This is described further below.

The `ctlptr` argument is used to hold the control part of the message, and `dataptr` is used to hold the data part of the message. If `ctlptr` (or `dataptr`) is a null pointer or the `maxlen` member is -1, the control (or data) part of the message shall not be processed and shall be left on the STREAM head read queue, and if the `ctlptr` (or `dataptr`) is not a null pointer, `len` shall be set to -1. If the `maxlen` member is set to 0 and there is a zero-length control (or data) part, that zero-length part shall be removed from the read queue and `len` shall be set to 0. If the `maxlen` member is set to 0 and there are more than 0 bytes of control (or data) information, that information shall be left on the read queue and `len` shall be set to 0. If the `maxlen` member in `ctlptr` (or `dataptr`) is less than the control (or data) part of the message, `maxlen` bytes shall be retrieved. In this case, the remainder of the message shall be left on the STREAM head read queue and a non-zero return value shall be provided.

By default, `getmsg()` shall process the first available message on the STREAM head read queue. However, a process may choose to retrieve only high-priority messages by setting the integer pointed to by `flagsp` to `RS_HIPRI`. In this case, `getmsg()` shall only process the next message if it is a high-priority message. When the integer pointed to by `flagsp` is 0, any available message shall be retrieved. In this case, on return, the integer pointed to by `flagsp` shall be set to `RS_HIPRI` if a high-priority message was retrieved, or 0 otherwise.

For `getpmsg()`, the flags are different. The `flagsp` argument points to a bitmask with the following mutually-exclusive flags defined: `MSG_HIPRI`, `MSG_BAND`, and `MSG_ANY`. Like `getmsg()`,

*getpmsg()* shall process the first available message on the STREAM head read queue. A process may choose to retrieve only high-priority messages by setting the integer pointed to by *flagsp* to MSG\_HIPRI and the integer pointed to by *bandp* to 0. In this case, *getpmsg()* shall only process the next message if it is a high-priority message. In a similar manner, a process may choose to retrieve a message from a particular priority band by setting the integer pointed to by *flagsp* to MSG\_BAND and the integer pointed to by *bandp* to the priority band of interest. In this case, *getpmsg()* shall only process the next message if it is in a priority band equal to, or greater than, the integer pointed to by *bandp*, or if it is a high-priority message. If a process wants to get the first message off the queue, the integer pointed to by *flagsp* should be set to MSG\_ANY and the integer pointed to by *bandp* should be set to 0. On return, if the message retrieved was a high-priority message, the integer pointed to by *flagsp* shall be set to MSG\_HIPRI and the integer pointed to by *bandp* shall be set to 0. Otherwise, the integer pointed to by *flagsp* shall be set to MSG\_BAND and the integer pointed to by *bandp* shall be set to the priority band of the message.

If O\_NONBLOCK is not set, *getmsg()* and *getpmsg()* shall block until a message of the type specified by *flagsp* is available at the front of the STREAM head read queue. If O\_NONBLOCK is set and a message of the specified type is not present at the front of the read queue, *getmsg()* and *getpmsg()* shall fail and set *errno* to [EAGAIN].

If a hangup occurs on the STREAM from which messages are retrieved, *getmsg()* and *getpmsg()* shall continue to operate normally, as described above, until the STREAM head read queue is empty. Thereafter, they shall return 0 in the *len* members of *ctlptr* and *dataptr*.

## RETURN VALUE

Upon successful completion, *getmsg()* and *getpmsg()* shall return a non-negative value. A value of 0 indicates that a full message was read successfully. A return value of MORECTL indicates that more control information is waiting for retrieval. A return value of MOREDATA indicates that more data is waiting for retrieval. A return value of the bitwise-logical OR of MORECTL and MOREDATA indicates that both types of information remain. Subsequent *getmsg()* and *getpmsg()* calls shall retrieve the remainder of the message. However, if a message of higher priority has come in on the STREAM head read queue, the next call to *getmsg()* or *getpmsg()* shall retrieve that higher-priority message before retrieving the remainder of the previous message.

If the high priority control part of the message is consumed, the message shall be placed back on the queue as a normal message of band 0. Subsequent *getmsg()* and *getpmsg()* calls shall retrieve the remainder of the message. If, however, a priority message arrives or already exists on the STREAM head, the subsequent call to *getmsg()* or *getpmsg()* shall retrieve the higher-priority message before retrieving the remainder of the message that was put back.

Upon failure, *getmsg()* and *getpmsg()* shall return -1 and set *errno* to indicate the error.

## ERRORS

The *getmsg()* and *getpmsg()* functions shall fail if:

|           |                                                                                                                                                                              |
|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [EAGAIN]  | The O_NONBLOCK flag is set and no messages are available.                                                                                                                    |
| [EBADF]   | The <i>fildev</i> argument is not a valid file descriptor open for reading.                                                                                                  |
| [EBADMSG] | The queued message to be read is not valid for <i>getmsg()</i> or <i>getpmsg()</i> or a pending file descriptor is at the STREAM head.                                       |
| [EINTR]   | A signal was caught during <i>getmsg()</i> or <i>getpmsg()</i> .                                                                                                             |
| [EINVAL]  | An illegal value was specified by <i>flagsp</i> , or the STREAM or multiplexer referenced by <i>fildev</i> is linked (directly or indirectly) downstream from a multiplexer. |

35751 [ENOSTR] A STREAM is not associated with *fildev*.

35752 In addition, *getmsg()* and *getpmsg()* shall fail if the STREAM head had processed an  
35753 asynchronous error before the call. In this case, the value of *errno* does not reflect the result of  
35754 *getmsg()* or *getpmsg()* but reflects the prior error.

## 35755 EXAMPLES

### 35756 Getting Any Message

35757 In the following example, the value of *fd* is assumed to refer to an open STREAMS file. The call  
35758 to *getmsg()* retrieves any available message on the associated STREAM-head read queue,  
35759 returning control and data information to the buffers pointed to by *ctrlbuf* and *databuf*,  
35760 respectively.

```
35761 #include <stropts.h>
35762 ...
35763 int fd;
35764 char ctrlbuf[128];
35765 char databuf[512];
35766 struct strbuf ctrl;
35767 struct strbuf data;
35768 int flags = 0;
35769 int ret;

35770 ctrl.buf = ctrlbuf;
35771 ctrl.maxlen = sizeof(ctrlbuf);

35772 data.buf = databuf;
35773 data.maxlen = sizeof(databuf);

35774 ret = getmsg (fd, &ctrl, &data, &flags);
```

### 35775 Getting the First Message off the Queue

35776 In the following example, the call to *getpmsg()* retrieves the first available message on the  
35777 associated STREAM-head read queue.

```
35778 #include <stropts.h>
35779 ...

35780 int fd;
35781 char ctrlbuf[128];
35782 char databuf[512];
35783 struct strbuf ctrl;
35784 struct strbuf data;
35785 int band = 0;
35786 int flags = MSG_ANY;
35787 int ret;

35788 ctrl.buf = ctrlbuf;
35789 ctrl.maxlen = sizeof(ctrlbuf);

35790 data.buf = databuf;
35791 data.maxlen = sizeof(databuf);

35792 ret = getpmsg (fd, &ctrl, &data, &band, &flags);
```

35793 **APPLICATION USAGE**

35794 None.

35795 **RATIONALE**

35796 None.

35797 **FUTURE DIRECTIONS**35798 The *getmsg()* and *getpmsg()* functions may be removed in a future version.35799 **SEE ALSO**35800 [Section 2.6](#) (on page 500), [poll\(\)](#), [putmsg\(\)](#), [read\(\)](#), [write\(\)](#)35801 XBD [<stropts.h>](#)35802 **CHANGE HISTORY**

35803 First released in Issue 4, Version 2.

35804 **Issue 5**

35805 Moved from X/OPEN UNIX extension to BASE.

35806 A paragraph regarding “high-priority control parts of messages” is added to the RETURN  
35807 VALUE section.35808 **Issue 6**

35809 This function is marked as part of the XSI STREAMS Option Group.

35810 The **restrict** keyword is added to the *getmsg()* and *getpmsg()* prototypes for alignment with the  
35811 ISO/IEC 9899:1999 standard.35812 **Issue 7**35813 The *getmsg()* and *getpmsg()* functions are marked obsolescent.



## NAME

getnameinfo — get name information

## SYNOPSIS

```
#include <sys/socket.h>
#include <netdb.h>

int getnameinfo(const struct sockaddr *restrict sa, socklen_t salen,
                char *restrict node, socklen_t nodelen, char *restrict service,
                socklen_t servicelen, int flags);
```

## DESCRIPTION

The `getnameinfo()` function shall translate a socket address to a node name and service location, all of which are defined as in [freeaddrinfo\(\)](#).

The *sa* argument points to a socket address structure to be translated. The *salen* argument contains the length of the address pointed to by *sa*.

IP6 If the socket address structure contains an IPv4-mapped IPv6 address or an IPv4-compatible IPv6 address, the implementation shall extract the embedded IPv4 address and lookup the node name for that IPv4 address.

If the address is the IPv6 unspecified address ("::"), a lookup shall not be performed and the behavior shall be the same as when the node's name cannot be located.

If the *node* argument is non-NULL and the *nodelen* argument is non-zero, then the *node* argument points to a buffer able to contain up to *nodelen* bytes that receives the node name as a null-terminated string. If the *node* argument is NULL or the *nodelen* argument is zero, the node name shall not be returned. If the node's name cannot be located, the numeric form of the address contained in the socket address structure pointed to by the *sa* argument is returned instead of its name.

If the *service* argument is non-NULL and the *servicelen* argument is non-zero, then the *service* argument points to a buffer able to contain up to *servicelen* bytes that receives the service name as a null-terminated string. If the *service* argument is NULL or the *servicelen* argument is zero, the service name shall not be returned. If the service's name cannot be located, the numeric form of the service address (for example, its port number) shall be returned instead of its name.

The *flags* argument is a flag that changes the default actions of the function. By default the fully-qualified domain name (FQDN) for the host shall be returned, but:

- If the flag bit NI\_NOFQDN is set, only the node name portion of the FQDN shall be returned for local hosts.
- If the flag bit NI\_NUMERICHOST is set, the numeric form of the address contained in the socket address structure pointed to by the *sa* argument shall be returned instead of its name.
- If the flag bit NI\_NAMEREQD is set, an error shall be returned if the host's name cannot be located.
- If the flag bit NI\_NUMERICSERV is set, the numeric form of the service address shall be returned (for example, its port number) instead of its name.
- If the flag bit NI\_NUMERICSERVICE is set, the numeric form of the scope identifier shall be returned (for example, interface index) instead of its name. This flag shall be ignored if the *sa* argument is not an IPv6 address.



- If the flag bit NI\_DGRAM is set, this indicates that the service is a datagram service (SOCK\_DGRAM). The default behavior shall assume that the service is a stream service (SOCK\_STREAM).

**Notes:**

1. The two NI\_NUMERICxxx flags are required to support the `-n` flag that many commands provide.
2. The NI\_DGRAM flag is required for the few AF\_INET and AF\_INET6 port numbers (for example, [512,514]) that represent different services for UDP and TCP.

The `getnameinfo()` function shall be thread-safe.

**RETURN VALUE**

A zero return value for `getnameinfo()` indicates successful completion; a non-zero return value indicates failure. The possible values for the failures are listed in the ERRORS section.

Upon successful completion, `getnameinfo()` shall return the *node* and *service* names, if requested, in the buffers provided. The returned names are always null-terminated strings.

**ERRORS**

The `getnameinfo()` function shall fail and return the corresponding value if:

[EAI\_AGAIN] The name could not be resolved at this time. Future attempts may succeed.

[EAI\_BADFLAGS]

The *flags* had an invalid value.

[EAI\_FAIL]

A non-recoverable error occurred.

[EAI\_FAMILY]

The address family was not recognized or the address length was invalid for the specified family.

[EAI\_MEMORY] There was a memory allocation failure.

[EAI\_NONAME] The name does not resolve for the supplied parameters.

NI\_NAMEREQD is set and the host's name cannot be located, or both *nodename* and *servname* were null.

[EAI\_OVERFLOW]

An argument buffer overflowed. The buffer pointed to by the *node* argument or the *service* argument was too small.

[EAI\_SYSTEM]

A system error occurred. The error code can be found in *errno*.

**EXAMPLES**

None.

**APPLICATION USAGE**

If the returned values are to be used as part of any further name resolution (for example, passed to `getaddrinfo()`), applications should provide buffers large enough to store any result possible on the system.

Given the IPv4-mapped IPv6 address "`::ffff:1.2.3.4`", the implementation performs a lookup as if the socket address structure contains the IPv4 address "`1.2.3.4`".

The IPv6 unspecified address ("`:::`") and the IPv6 loopback address ("`:::1`") are not IPv4-compatible addresses.

35897 **RATIONALE**

35898 None.

35899 **FUTURE DIRECTIONS**

35900 None.

35901 **SEE ALSO**

35902 *endservent(), freeaddrinfo(), gai\_strerror(), inet\_ntop(), socket()*

35903 XBD *<netdb.h>*, *<sys/socket.h>*

35904 **CHANGE HISTORY**

35905 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

35906 The **restrict** keyword is added to the *getnameinfo()* prototype for alignment with the  
35907 ISO/IEC 9899:1999 standard.

35908 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/23 is applied, making various changes in  
35909 the SYNOPSIS and DESCRIPTION for alignment with IPv6.

35910 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/24 is applied, adding the  
35911 [EAI\_OVERFLOW] error to the ERRORS section.

35912 **Issue 7**

35913 SD5-XSH-ERN-127 is applied, clarifying the behavior if the address is the IPv6 unspecified  
35914 address.

35915 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0246 [284] and XSH/TC1-2008/0247  
35916 [285] are applied.

35917 **NAME**

35918 getnetbyaddr, getnetbyname, getnetent — network database functions

35919 **SYNOPSIS**

35920 #include &lt;netdb.h&gt;

35921 struct netent \*getnetbyaddr(uint32\_t net, int type);

35922 struct netent \*getnetbyname(const char \*name);

35923 struct netent \*getnetent(void);

35924 **DESCRIPTION**35925 Refer to *endnetent()*.

**NAME**

getopt, optarg, opterr, optind, optopt — command option parsing

**SYNOPSIS**

```
#include <unistd.h>

int getopt(int argc, char * const argv[], const char *optstring);
extern char *optarg;
extern int opterr, optind, optopt;
```

**DESCRIPTION**

The `getopt()` function is a command-line parser that shall follow Utility Syntax Guidelines 3, 4, 5, 6, 7, 9, and 10 in XBD [Section 12.2](#) (on page 216).

The parameters *argc* and *argv* are the argument count and argument array as passed to `main()` (see `exec()`). The argument *optstring* is a string of recognized option characters; if a character is followed by a <colon>, the option takes an argument. All option characters allowed by Utility Syntax Guideline 3 are allowed in *optstring*. The implementation may accept other characters as an extension.

The variable *optind* is the index of the next element of the *argv*[] vector to be processed. It shall be initialized to 1 by the system, and `getopt()` shall update it when it finishes with each element of *argv* []. If the application sets *optind* to zero before calling `getopt()`, the behavior is unspecified. When an element of *argv* [] contains multiple option characters, it is unspecified how `getopt()` determines which options have already been processed.

The `getopt()` function shall return the next option character (if one is found) from *argv* that matches a character in *optstring*, if there is one that matches. If the option takes an argument, `getopt()` shall set the variable *optarg* to point to the option-argument as follows:

1. If the option was the last character in the string pointed to by an element of *argv*, then *optarg* shall contain the next element of *argv*, and *optind* shall be incremented by 2. If the resulting value of *optind* is greater than *argc*, this indicates a missing option-argument, and `getopt()` shall return an error indication.
2. Otherwise, *optarg* shall point to the string following the option character in that element of *argv*, and *optind* shall be incremented by 1.

If, when `getopt()` is called:

```
argv[optind]  is a null pointer
*argv[optind] is not the character -
argv[optind]  points to the string "--"
```

`getopt()` shall return -1 without changing *optind*. If:

```
argv[optind]  points to the string "--"
```

`getopt()` shall return -1 after incrementing *optind*.

If `getopt()` encounters an option character that is not contained in *optstring*, it shall return the <question-mark> ('?') character. If it detects a missing option-argument, it shall return the <colon> character (':') if the first character of *optstring* was a <colon>, or a <question-mark> character ('?') otherwise. In either case, `getopt()` shall set the variable *optopt* to the option character that caused the error. If the application has not set the variable *opterr* to 0 and the first character of *optstring* is not a <colon>, `getopt()` shall also print a diagnostic message to *stderr* in the format specified for the `getopts` utility, unless the *stderr* stream has wide orientation, in which case the behavior is undefined.

The `getopt()` function need not be thread-safe.

**RETURN VALUE**

The `getopt()` function shall return the next option character specified on the command line.

A <colon> (':') shall be returned if `getopt()` detects a missing argument and the first character of *optstring* was a <colon> (':').

A <question-mark> ('?') shall be returned if `getopt()` encounters an option character not in *optstring* or detects a missing argument and the first character of *optstring* was not a <colon> (':').

Otherwise, `getopt()` shall return -1 when all command line options are parsed.

**ERRORS**

If the application has not set the variable *opterr* to 0, the first character of *optstring* is not a <colon>, and a write error occurs while `getopt()` is printing a diagnostic message to *stderr*, then the error indicator for *stderr* shall be set; but `getopt()` shall still succeed and the value of *errno* after `getopt()` is unspecified.

**EXAMPLES****Parsing Command Line Options**

The following code fragment shows how you might process the arguments for a utility that can take the mutually-exclusive options *a* and *b* and the options *f* and *o*, both of which require arguments:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int
main(int argc, char *argv[ ])
{
    int c;
    int bflg = 0, aflg = 0, errflg = 0;
    char *ifile;
    char *ofile;
    . . .
    while ((c = getopt(argc, argv, ":abf:o:")) != -1) {
        switch(c) {
            case 'a':
                if (bflg)
                    errflg++;
                else
                    aflg++;
                break;
            case 'b':
                if (aflg)
                    errflg++;
                else
                    bflg++;
                break;
            case 'f':
                ifile = optarg;
                break;
            case 'o':
```

```

36018         ofile = optarg;
36019         break;
36020     case ':': /* -f or -o without operand */
36021         fprintf(stderr,
36022             "Option -%c requires an operand\n", optopt);
36023         errflg++;
36024         break;
36025     case '?':
36026         fprintf(stderr,
36027             "Unrecognized option: '-%c'\n", optopt);
36028         errflg++;
36029     }
36030 }
36031 if (errflg) {
36032     fprintf(stderr, "usage: . . . ");
36033     exit(2);
36034 }
36035 for ( ; optind < argc; optind++) {
36036     if (access(argv[optind], R_OK)) {
36037         . . .
36038     }

```

This code accepts any of the following as equivalent:

```

36040 cmd -ao arg path path
36041 cmd -a -o arg path path
36042 cmd -o arg -a path path
36043 cmd -a -o arg -- path path
36044 cmd -a -oarg path path
36045 cmd -aoarg path path

```

### Selecting Options from the Command Line

The following example selects the type of database routines the user wants to use based on the *Options* argument.

```

36049 #include <unistd.h>
36050 #include <string.h>
36051 ...
36052 const char *Options = "hdbt1";
36053 ...
36054 int dbtype, c;
36055 char *st;
36056 ...
36057 dbtype = 0;
36058 while ((c = getopt(argc, argv, Options)) != -1) {
36059     if ((st = strchr(Options, c)) != NULL) {
36060         dbtype = st - Options;
36061         break;
36062     }
36063 }

```

**APPLICATION USAGE**

The `getopt()` function is only required to support option characters included in Utility Syntax Guideline 3. Many historical implementations of `getopt()` support other characters as options. This is an allowed extension, but applications that use extensions are not maximally portable. Note that support for multi-byte option characters is only possible when such characters can be represented as type `int`.

Applications which use wide-character output functions with `stderr` should ensure that any calls to `getopt()` do not write to `stderr`, either by setting `opterr` to 0 or by ensuring the first character of `optstring` is always a `<colon>`.

While `ferror(stderr)` may be used to detect failures to write a diagnostic to `stderr` when `getopt()` returns `'?'`, the value of `errno` is unspecified in such a condition. Applications desiring more control over handling write failures should set `opterr` to 0 and independently perform output to `stderr`, rather than relying on `getopt()` to do the output.

**RATIONALE**

The `optopt` variable represents historical practice and allows the application to obtain the identity of the invalid option.

The description has been written to make it clear that `getopt()`, like the `getopts` utility, deals with option-arguments whether separated from the option by `<blank>` characters or not. Note that the requirements on `getopt()` and `getopts` are more stringent than the Utility Syntax Guidelines.

The `getopt()` function shall return `-1`, rather than EOF, so that `<stdio.h>` is not required.

The special significance of a `<colon>` as the first character of `optstring` makes `getopt()` consistent with the `getopts` utility. It allows an application to make a distinction between a missing argument and an incorrect option letter without having to examine the option letter. It is true that a missing argument can only be detected in one case, but that is a case that has to be considered.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*exec*

XBD Section 12.2 (on page 216), `<unistd.h>`

XCU *getopts*

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

A note indicating that the `getopt()` function need not be reentrant is added to the DESCRIPTION.

**Issue 6**

IEEE PASC Interpretation 1003.2 #150 is applied.

Austin Group Interpretation 1003.1-2001 #156 is applied.

**Issue 7**

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0248 [318], XSH/TC1-2008/0249 [460], XSH/TC1-2008/0250 [189], XSH/TC1-2008/0251 [189], XSH/TC1-2008/0252 [189], and XSH/TC1-2008/0253 [460] are applied.

36106 **NAME**

36107 `getpeername` — get the name of the peer socket

36108 **SYNOPSIS**

36109 `#include <sys/socket.h>`  
36110 `int getpeername(int socket, struct sockaddr *restrict address,`  
36111 `socklen_t *restrict address_len);`

36112 **DESCRIPTION**

36113 The `getpeername()` function shall retrieve the peer address of the specified socket, store this  
36114 address in the **sockaddr** structure pointed to by the *address* argument, and store the length of this  
36115 address in the object pointed to by the *address\_len* argument.

36116 The *address\_len* argument points to a **socklen\_t** object which on input specifies the length of the  
36117 supplied **sockaddr** structure, and on output specifies the length of the stored address. If the  
36118 actual length of the address is greater than the length of the supplied **sockaddr** structure, the  
36119 stored address shall be truncated.

36120 If the protocol permits connections by unbound clients, and the peer is not bound, then the  
36121 value stored in the object pointed to by *address* is unspecified.

36122 **RETURN VALUE**

36123 Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to  
36124 indicate the error.

36125 **ERRORS**

36126 The `getpeername()` function shall fail if:

- |       |              |                                                                              |
|-------|--------------|------------------------------------------------------------------------------|
| 36127 | [EBADF]      | The <i>socket</i> argument is not a valid file descriptor.                   |
| 36128 | [EINVAL]     | The socket has been shut down.                                               |
| 36129 | [ENOTCONN]   | The socket is not connected or otherwise has not had the peer pre-specified. |
| 36130 | [ENOTSOCK]   | The <i>socket</i> argument does not refer to a socket.                       |
| 36131 | [EOPNOTSUPP] | The operation is not supported for the socket protocol.                      |

36132 The `getpeername()` function may fail if:

- |       |           |                                                                           |
|-------|-----------|---------------------------------------------------------------------------|
| 36133 | [ENOBUFS] | Insufficient resources were available in the system to complete the call. |
|-------|-----------|---------------------------------------------------------------------------|

36134 **EXAMPLES**

36135 None.

36136 **APPLICATION USAGE**

36137 None.

36138 **RATIONALE**

36139 None.

36140 **FUTURE DIRECTIONS**

36141 None.

36142 **SEE ALSO**

36143 *accept(), bind(), getsockname(), socket()*

36144 XBD `<sys/socket.h>`



36145 **CHANGE HISTORY**

36146 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

36147 The **restrict** keyword is added to the `getpeername()` prototype for alignment with the  
36148 ISO/IEC 9899:1999 standard.36149 **Issue 7**

36150 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0254 [464] is applied.

36151 **NAME**

36152       getpgid — get the process group ID for a process

36153 **SYNOPSIS**

36154       #include &lt;unistd.h&gt;

36155       pid\_t getpgid(pid\_t pid);

36156 **DESCRIPTION**36157       The *getpgid()* function shall return the process group ID of the process whose process ID is equal  
36158       to *pid*. If *pid* is equal to 0, *getpgid()* shall return the process group ID of the calling process.36159 **RETURN VALUE**36160       Upon successful completion, *getpgid()* shall return a process group ID. Otherwise, it shall return  
36161       (*pid\_t*)-1 and set *errno* to indicate the error.36162 **ERRORS**36163       The *getpgid()* function shall fail if:36164       [EPERM]       The process whose process ID is equal to *pid* is not in the same session as the  
36165       calling process, and the implementation does not allow access to the process  
36166       group ID of that process from the calling process.36167       [ESRCH]       There is no process with a process ID equal to *pid*.36168       The *getpgid()* function may fail if:36169       [EINVAL]       The value of the *pid* argument is invalid.36170 **EXAMPLES**

36171       None.

36172 **APPLICATION USAGE**

36173       None.

36174 **RATIONALE**

36175       None.

36176 **FUTURE DIRECTIONS**

36177       None.

36178 **SEE ALSO**36179       *exec*, *fork()*, *getpgrp()*, *getpid()*, *getsid()*, *setpgid()*, *setsid()*

36180       XBD &lt;unistd.h&gt;

36181 **CHANGE HISTORY**

36182       First released in Issue 4, Version 2.

36183 **Issue 5**

36184       Moved from X/OPEN UNIX extension to BASE.

36185 **Issue 7**36186       The *getpgid()* function is moved from the XSI option to the Base.

36187 **NAME**

36188       getpgrp — get the process group ID of the calling process

36189 **SYNOPSIS**

36190       #include &lt;unistd.h&gt;

36191       pid\_t getpgrp(void);

36192 **DESCRIPTION**36193       The *getpgrp()* function shall return the process group ID of the calling process.36194 **RETURN VALUE**36195       The *getpgrp()* function shall always be successful and no return value is reserved to indicate an  
36196       error.36197 **ERRORS**

36198       No errors are defined.

36199 **EXAMPLES**

36200       None.

36201 **APPLICATION USAGE**

36202       None.

36203 **RATIONALE**36204       4.3 BSD provides a *getpgrp()* function that returns the process group ID for a specified process.  
36205       Although this function supports job control, all known job control shells always specify the  
36206       calling process with this function. Thus, the simpler System V *getpgrp()* suffices, and the added  
36207       complexity of the 4.3 BSD *getpgrp()* is provided by the XSI extension *getpgid()*.36208 **FUTURE DIRECTIONS**

36209       None.

36210 **SEE ALSO**36211       *exec*, *fork()*, *getpgid()*, *getpid()*, *getppid()*, *kill()*, *setpgid()*, *setsid()*

36212       XBD &lt;sys/types.h&gt;, &lt;unistd.h&gt;

36213 **CHANGE HISTORY**

36214       First released in Issue 1. Derived from Issue 1 of the SVID.

36215 **Issue 6**

36216       In the SYNOPSIS, the optional include of the &lt;sys/types.h&gt; header is removed.

36217       The following new requirements on POSIX implementations derive from alignment with the  
36218       Single UNIX Specification:

- 36219
- The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was  
36220       required for conforming implementations of previous POSIX specifications, it was not  
36221       required for UNIX applications.

36222 **NAME**

36223       getpid — get the process ID

36224 **SYNOPSIS**

36225       #include &lt;unistd.h&gt;

36226       pid\_t getpid(void);

36227 **DESCRIPTION**36228       The *getpid()* function shall return the process ID of the calling process.36229 **RETURN VALUE**36230       The *getpid()* function shall always be successful and no return value is reserved to indicate an  
36231       error.36232 **ERRORS**

36233       No errors are defined.

36234 **EXAMPLES**

36235       None.

36236 **APPLICATION USAGE**

36237       None.

36238 **RATIONALE**

36239       None.

36240 **FUTURE DIRECTIONS**

36241       None.

36242 **SEE ALSO**36243       *exec*, *fork()*, *getpgrp()*, *getppid()*, *kill()*, *mkdtemp()*, *setpgid()*, *setsid()*

36244       XBD &lt;sys/types.h&gt;, &lt;unistd.h&gt;

36245 **CHANGE HISTORY**

36246       First released in Issue 1. Derived from Issue 1 of the SVID.

36247 **Issue 6**

36248       In the SYNOPSIS, the optional include of the &lt;sys/types.h&gt; header is removed.

36249       The following new requirements on POSIX implementations derive from alignment with the  
36250       Single UNIX Specification:

- 36251
- The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was  
36252       required for conforming implementations of previous POSIX specifications, it was not  
36253       required for UNIX applications.

36254 **NAME**

36255 getpmsg — receive next message from a STREAMS file

36256 **SYNOPSIS**

```
36257 OB XSI #include <stropts.h>
36258 int getpmsg(int fildes, struct strbuf *restrict ctlptr,
36259             struct strbuf *restrict dataptr, int *restrict bandp,
36260             int *restrict flagsp);
```

36261 **DESCRIPTION**36262 Refer to *getmsg()*.

36263 **NAME**

36264       getppid — get the parent process ID

36265 **SYNOPSIS**

36266       #include &lt;unistd.h&gt;

36267       pid\_t getppid(void);

36268 **DESCRIPTION**36269       The *getppid()* function shall return the parent process ID of the calling process.36270 **RETURN VALUE**36271       The *getppid()* function shall always be successful and no return value is reserved to indicate an  
36272       error.36273 **ERRORS**

36274       No errors are defined.

36275 **EXAMPLES**

36276       None.

36277 **APPLICATION USAGE**

36278       None.

36279 **RATIONALE**

36280       None.

36281 **FUTURE DIRECTIONS**

36282       None.

36283 **SEE ALSO**36284       *exec*, *fork()*, *getpgid()*, *getpgrp()*, *getpid()*, *kill()*, *setpgid()*, *setsid()*

36285       XBD &lt;sys/types.h&gt;, &lt;unistd.h&gt;

36286 **CHANGE HISTORY**

36287       First released in Issue 1. Derived from Issue 1 of the SVID.

36288 **Issue 6**

36289       In the SYNOPSIS, the optional include of the &lt;sys/types.h&gt; header is removed.

36290       The following new requirements on POSIX implementations derive from alignment with the  
36291       Single UNIX Specification:

- 36292
- The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was  
36293       required for conforming implementations of previous POSIX specifications, it was not  
36294       required for UNIX applications.

**NAME**

getpriority, setpriority — get and set the nice value

**SYNOPSIS**

```
#include <sys/resource.h>

int getpriority(int which, id_t who);
int setpriority(int which, id_t who, int value);
```

**DESCRIPTION**

The *getpriority()* function shall obtain the nice value of a process, process group, or user. The *setpriority()* function shall set the nice value of a process, process group, or user to *value*+{NZERO}.

Target processes are specified by the values of the *which* and *who* arguments. The *which* argument may be one of the following values: PRIO\_PROCESS, PRIO\_PGRP, or PRIO\_USER, indicating that the *who* argument is to be interpreted as a process ID, a process group ID, or an effective user ID, respectively. A 0 value for the *who* argument specifies the current process, process group, or user.

The nice value set with *setpriority()* shall be applied to the process. If the process is multi-threaded, the nice value shall affect all system scope threads in the process.

If more than one process is specified, *getpriority()* shall return value {NZERO} less than the lowest nice value pertaining to any of the specified processes, and *setpriority()* shall set the nice values of all of the specified processes to *value*+{NZERO}.

The default nice value is {NZERO}; lower nice values shall cause more favorable scheduling. While the range of valid nice values is [0,{NZERO}\*2-1], implementations may enforce more restrictive limits. If *value*+{NZERO} is less than the system's lowest supported nice value, *setpriority()* shall set the nice value to the lowest supported value; if *value*+{NZERO} is greater than the system's highest supported nice value, *setpriority()* shall set the nice value to the highest supported value.

Only a process with appropriate privileges can lower its nice value.

Any processes or threads using SCHED\_FIFO or SCHED\_RR shall be unaffected by a call to *setpriority()*. This is not considered an error. A process which subsequently reverts to SCHED\_OTHER need not have its priority affected by such a *setpriority()* call.

The effect of changing the nice value may vary depending on the process-scheduling algorithm in effect.

Since *getpriority()* can return the value -1 upon successful completion, it is necessary to set *errno* to 0 prior to a call to *getpriority()*. If *getpriority()* returns the value -1, then *errno* can be checked to see if an error occurred or if the value is a legitimate nice value.

**RETURN VALUE**

Upon successful completion, *getpriority()* shall return an integer in the range -{NZERO} to {NZERO}-1. Otherwise, -1 shall be returned and *errno* set to indicate the error.

Upon successful completion, *setpriority()* shall return 0; otherwise, -1 shall be returned and *errno* set to indicate the error.

**ERRORS**

The *getpriority()* and *setpriority()* functions shall fail if:

[ESRCH] No process could be located using the *which* and *who* argument values specified.

[EINVAL] The value of the *which* argument was not recognized, or the value of the *who* argument is not a valid process ID, process group ID, or user ID.

In addition, *setpriority()* may fail if:

[EPERM] A process was located, but neither the real nor effective user ID of the executing process match the effective user ID of the process whose nice value is being changed.

[EACCES] A request was made to change the nice value to a lower numeric value and the current process does not have appropriate privileges.

**EXAMPLES****Using getpriority()**

The following example returns the current scheduling priority for the process ID returned by the call to *getpid()*.

```
#include <sys/resource.h>
...
int which = PRIO_PROCESS;
id_t pid;
int ret;

pid = getpid();
ret = getpriority(which, pid);
```

**Using setpriority()**

The following example sets the priority for the current process ID to -20.

```
#include <sys/resource.h>
...
int which = PRIO_PROCESS;
id_t pid;
int priority = -20;
int ret;

pid = getpid();
ret = setpriority(which, pid, priority);
```

**APPLICATION USAGE**

The *getpriority()* and *setpriority()* functions work with an offset nice value (nice value -{NZERO}). The nice value is in the range [0,2\*{NZERO} -1], while the return value for *getpriority()* and the third parameter for *setpriority()* are in the range [-{NZERO},{NZERO} -1].

**RATIONALE**

None.



36374 **FUTURE DIRECTIONS**

36375 None.

36376 **SEE ALSO**36377 *nice()*, *sched\_get\_priority\_max()*, *sched\_setscheduler()*

36378 XBD &lt;sys/resource.h&gt;

36379 **CHANGE HISTORY**

36380 First released in Issue 4, Version 2.

36381 **Issue 5**

36382 Moved from X/OPEN UNIX extension to BASE.

36383 The DESCRIPTION is reworded in terms of the nice value rather than *priority* to avoid confusion  
36384 with functionality in the POSIX Realtime Extension.

36385 **NAME**

36386 getprotobyname, getprotobynumber, getprotent — network protocol database functions

36387 **SYNOPSIS**

36388 #include &lt;netdb.h&gt;

36389 struct protoent \*getprotobyname(const char \*name);

36390 struct protoent \*getprotobynumber(int proto);

36391 struct protoent \*getprotoent(void);

36392 **DESCRIPTION**36393 Refer to *endprotoent()*.

36394 **NAME**

36395       getpwent — get user database entry

36396 **SYNOPSIS**

```
36397 XSI      #include <pwd.h>
36398          struct passwd *getpwent(void);
```

36399 **DESCRIPTION**36400       Refer to *endpwent()*.

**NAME**

getpwnam, getpwnam\_r — search user database for a name

**SYNOPSIS**

```
#include <pwd.h>

struct passwd *getpwnam(const char *name);
int getpwnam_r(const char *name, struct passwd *pwd, char *buffer,
               size_t bufsize, struct passwd **result);
```

**DESCRIPTION**

The *getpwnam()* function shall search the user database for an entry with a matching *name*.

The *getpwnam()* function need not be thread-safe.

Applications wishing to check for error situations should set *errno* to 0 before calling *getpwnam()*. If *getpwnam()* returns a null pointer and *errno* is non-zero, an error occurred.

The *getpwnam\_r()* function shall update the **passwd** structure pointed to by *pwd* and store a pointer to that structure at the location pointed to by *result*. The structure shall contain an entry from the user database with a matching *name*. Storage referenced by the structure is allocated from the memory provided with the *buffer* parameter, which is *bufsize* bytes in size. A call to *sysconf*(*\_SC\_GETPW\_R\_SIZE\_MAX*) returns either -1 without changing *errno* or an initial value suggested for the size of this buffer. A null pointer shall be returned at the location pointed to by *result* on error or if the requested entry is not found.

**RETURN VALUE**

The *getpwnam()* function shall return a pointer to a **struct passwd** with the structure as defined in **<pwd.h>** with a matching entry if found. A null pointer shall be returned if the requested entry is not found, or an error occurs. On error, *errno* shall be set to indicate the error.

The application shall not modify the structure to which the return value points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *getpwent()*, *getpwnam()*, or *getpwuid()*. The returned pointer, and pointers within the structure, might also be invalidated if the calling thread is terminated.

The *getpwnam\_r()* function shall return zero on success or if the requested entry was not found and no error has occurred. If an error has occurred, an error number shall be returned to indicate the error.

**ERRORS**

These functions may fail if:

- |          |                                                                        |
|----------|------------------------------------------------------------------------|
| [EIO]    | An I/O error has occurred.                                             |
| [EINTR]  | A signal was caught during <i>getpwnam()</i> .                         |
| [EMFILE] | All file descriptors available to the process are currently open.      |
| [ENFILE] | The maximum allowable number of files is currently open in the system. |

The *getpwnam\_r()* function may fail if:

- |          |                                                                                                                                                       |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| [ERANGE] | Insufficient storage was supplied via <i>buffer</i> and <i>bufsize</i> to contain the data to be referenced by the resulting <b>passwd</b> structure. |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------------------|

**EXAMPLES**

Note that `sysconf(_SC_GETPW_R_SIZE_MAX)` may return `-1` if there is no hard limit on the size of the buffer needed to store all the groups returned. This example shows how an application can allocate a buffer of sufficient size to work with `getpwnam_r()`.

```

long int initlen = sysconf(_SC_GETPW_R_SIZE_MAX);
size_t len;
if (initlen == -1)
    /* Default initial length. */
    len = 1024;
else
    len = (size_t) initlen;
struct passwd result;
struct passwd *resultp;
char *buffer = malloc(len);
if (buffer == NULL)
    ...handle error...
int e;
while ((e = getpwnam_r("someuser", &result, buffer, len, &resultp))
    == ERANGE)
{
    size_t newlen = 2 * len;
    if (newlen < len)
        ...handle error...
    len = newlen;
    char *newbuffer = realloc(buffer, len);
    if (newbuffer == NULL)
        ...handle error...
    buffer = newbuffer;
}
if (e != 0)
    ...handle error...
free (buffer);

```

**Getting an Entry for the Login Name**

The following example uses the `getlogin()` function to return the name of the user who logged in; this information is passed to the `getpwnam()` function to get the user database entry for that user.

```

#include <sys/types.h>
#include <pwd.h>
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
...
char *lgn;
struct passwd *pw;
...
if ((lgn = getlogin()) == NULL || (pw = getpwnam(lgn)) == NULL) {
    fprintf(stderr, "Get of user information failed.\n"); exit(1);
}
...

```

**APPLICATION USAGE**

Three names associated with the current process can be determined: *getpwuid(geteuid())* returns the name associated with the effective user ID of the process; *getlogin()* returns the name associated with the current login activity; and *getpwuid(getuid())* returns the name associated with the real user ID of the process.

The *getpwnam\_r()* function is thread-safe and returns values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

Portable applications should take into account that it is usual for an implementation to return `-1` from *sysconf()* indicating that there is no maximum for `_SC_GETPW_R_SIZE_MAX`.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*getpwuid()*, *sysconf()*

XBD `<pwd.h>`, `<sys/types.h>`

**CHANGE HISTORY**

First released in Issue 1. Derived from System V Release 2.0.

**Issue 5**

Normative text previously in the APPLICATION USAGE section is moved to the RETURN VALUE section.

The *getpwnam\_r()* function is included for alignment with the POSIX Threads Extension.

A note indicating that the *getpwnam()* function need not be reentrant is added to the DESCRIPTION.

**Issue 6**

The *getpwnam\_r()* function is marked as part of the Thread-Safe Functions option.

The Open Group Corrigendum U028/3 is applied, correcting text in the DESCRIPTION describing matching the *name*.

In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- In the RETURN VALUE section, the requirement to set *errno* on error is added.
- The [EMFILE], [ENFILE], and [ENXIO] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

IEEE PASC Interpretation 1003.1 #116 is applied, changing the description of the size of the buffer from *bufsize* characters to bytes.

36530 **Issue 7**

36531 Austin Group Interpretation 1003.1-2001 #156 is applied.

36532 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

36533 SD5-XSH-ERN-166 is applied.

36534 The *getpwnam\_r()* function is moved from the Thread-Safe Functions option to the Base.

36535 A minor addition is made to the EXAMPLES section, reminding the application developer to  
36536 free memory allocated as if by *malloc()*.

36537 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0255 [75,428] is applied.

**NAME**

getpwuid, getpwuid\_r — search user database for a user ID

**SYNOPSIS**

```
#include <pwd.h>

struct passwd *getpwuid(uid_t uid);
int getpwuid_r(uid_t uid, struct passwd *pwd, char *buffer,
               size_t bufsize, struct passwd **result);
```

**DESCRIPTION**

The *getpwuid()* function shall search the user database for an entry with a matching *uid*.

The *getpwuid()* function need not be thread-safe.

Applications wishing to check for error situations should set *errno* to 0 before calling *getpwuid()*. If *getpwuid()* returns a null pointer and *errno* is set to non-zero, an error occurred.

The *getpwuid\_r()* function shall update the **passwd** structure pointed to by *pwd* and store a pointer to that structure at the location pointed to by *result*. The structure shall contain an entry from the user database with a matching *uid*. Storage referenced by the structure is allocated from the memory provided with the *buffer* parameter, which is *bufsize* bytes in size. A call to *sysconf(\_SC\_GETPW\_R\_SIZE\_MAX)* returns either -1 without changing *errno* or an initial value suggested for the size of this buffer. A null pointer shall be returned at the location pointed to by *result* on error or if the requested entry is not found.

**RETURN VALUE**

The *getpwuid()* function shall return a pointer to a **struct passwd** with the structure as defined in **<pwd.h>** with a matching entry if found. A null pointer shall be returned if the requested entry is not found, or an error occurs. On error, *errno* shall be set to indicate the error.

The application shall not modify the structure to which the return value points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *getpwent()*, *getpwnam()*, or *getpwuid()*. The returned pointer, and pointers within the structure, might also be invalidated if the calling thread is terminated.

If successful, the *getpwuid\_r()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

**ERRORS**

These functions may fail if:

- [EIO] An I/O error has occurred.
- [EINTR] A signal was caught during *getpwuid()*.
- [EMFILE] All file descriptors available to the process are currently open.
- [ENFILE] The maximum allowable number of files is currently open in the system.

The *getpwuid\_r()* function may fail if:

- [ERANGE] Insufficient storage was supplied via *buffer* and *bufsize* to contain the data to be referenced by the resulting **passwd** structure.



**EXAMPLES**

Note that `sysconf(_SC_GETPW_R_SIZE_MAX)` may return `-1` if there is no hard limit on the size of the buffer needed to store all the groups returned. This example shows how an application can allocate a buffer of sufficient size to work with `getpwuid_r()`.

```
long int initlen = sysconf(_SC_GETPW_R_SIZE_MAX);
size_t len;
if (initlen == -1)
    /* Default initial length. */
    len = 1024;
else
    len = (size_t) initlen;
struct passwd result;
struct passwd *resultp;
char *buffer = malloc(len);
if (buffer == NULL)
    ...handle error...
int e;
while ((e = getpwuid_r(42, &result, buffer, len, &resultp)) == ERANGE)
{
    size_t newlen = 2 * len;
    if (newlen < len)
        ...handle error...
    len = newlen;
    char *newbuffer = realloc(buffer, len);
    if (newbuffer == NULL)
        ...handle error...
    buffer = newbuffer;
}
if (e != 0)
    ...handle error...
free (buffer);
```

**Getting an Entry for the Root User**

The following example gets the user database entry for the user with user ID 0 (root).

```
#include <sys/types.h>
#include <pwd.h>
...
uid_t id = 0;
struct passwd *pwd;
pwd = getpwuid(id);
```

### Finding the Name for the Effective User ID

The following example defines *pws* as a pointer to a structure of type **passwd**, which is used to store the structure pointer returned by the call to the *getpwuid()* function. The *geteuid()* function shall return the effective user ID of the calling process; this is used as the search criteria for the *getpwuid()* function. The call to *getpwuid()* shall return a pointer to the structure containing that user ID value.

```
#include <unistd.h>
#include <sys/types.h>
#include <pwd.h>
...
struct passwd *pws;
pws = getpwuid(geteuid());
```

### Finding an Entry in the User Database

The following example uses *getpwuid()* to search the user database for a user ID that was previously stored in a **stat** structure, then prints out the user name if it is found. If the user is not found, the program prints the numeric value of the user ID for the entry.

```
#include <sys/types.h>
#include <pwd.h>
#include <stdio.h>
...
struct stat statbuf;
struct passwd *pwd;
...
if ((pwd = getpwuid(statbuf.st_uid)) != NULL)
    printf(" %-8.8s", pwd->pw_name);
else
    printf(" %-8d", statbuf.st_uid);
```

### APPLICATION USAGE

Three names associated with the current process can be determined: *getpwuid(geteuid())* returns the name associated with the effective user ID of the process; *getlogin()* returns the name associated with the current login activity; and *getpwuid(getuid())* returns the name associated with the real user ID of the process.

The *getpwuid\_r()* function is thread-safe and returns values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

Portable applications should take into account that it is usual for an implementation to return `-1` from *sysconf()* indicating that there is no maximum for `_SC_GETPW_R_SIZE_MAX`.

### RATIONALE

None.

### FUTURE DIRECTIONS

None.

### SEE ALSO

*getpwnam()*, *geteuid()*, *getuid()*, *getlogin()*, *sysconf()*

XBD **<pwd.h>**, **<sys/types.h>**

**CHANGE HISTORY**

First released in Issue 1. Derived from System V Release 2.0.

**Issue 5**

Normative text previously in the APPLICATION USAGE section is moved to the RETURN VALUE section.

The `getpwuid_r()` function is included for alignment with the POSIX Threads Extension.

A note indicating that the `getpwuid()` function need not be reentrant is added to the DESCRIPTION.

**Issue 6**

The `getpwuid_r()` function is marked as part of the Thread-Safe Functions option.

The Open Group Corrigendum U028/3 is applied, correcting text in the DESCRIPTION describing matching the `uid`.

In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.

In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- In the RETURN VALUE section, the requirement to set `errno` on error is added.
- The [EIO], [EINTR], [EMFILE], and [ENFILE] optional error conditions are added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

IEEE PASC Interpretation 1003.1 #116 is applied, changing the description of the size of the buffer from `bufsize` characters to bytes.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #156 is applied.

SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

SD5-XSH-ERN-166 is applied.

The `getpwuid_r()` function is moved from the Thread-Safe Functions option to the Base.

A minor addition is made to the EXAMPLES section, reminding the application developer to free memory allocated as if by `malloc()`.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0256 [75] is applied.

## 36692 NAME

36693 getrlimit, setrlimit — control maximum resource consumption

## 36694 SYNOPSIS

```
36695 XSI      #include <sys/resource.h>
36696
36696 int getrlimit(int resource, struct rlimit *rlp);
36697 int setrlimit(int resource, const struct rlimit *rlp);
```

## 36698 DESCRIPTION

36699 The *getrlimit()* function shall get, and the *setrlimit()* function shall set, limits on the consumption  
 36700 of a variety of resources.

36701 Each call to either *getrlimit()* or *setrlimit()* identifies a specific resource to be operated upon as  
 36702 well as a resource limit. A resource limit is represented by an **rlimit** structure. The *rlim\_cur*  
 36703 member specifies the current or soft limit and the *rlim\_max* member specifies the maximum or  
 36704 hard limit. Soft limits may be changed by a process to any value that is less than or equal to the  
 36705 hard limit. A process may (irreversibly) lower its hard limit to any value that is greater than or  
 36706 equal to the soft limit. Only a process with appropriate privileges can raise a hard limit. Both  
 36707 hard and soft limits can be changed in a single call to *setrlimit()* subject to the constraints  
 36708 described above.

36709 The value RLIM\_INFINITY, defined in **<sys/resource.h>**, shall be considered to be larger than  
 36710 any other limit value. If a call to *getrlimit()* returns RLIM\_INFINITY for a resource, it means the  
 36711 implementation shall not enforce limits on that resource. Specifying RLIM\_INFINITY as any  
 36712 resource limit value on a successful call to *setrlimit()* shall inhibit enforcement of that resource  
 36713 limit.

36714 The following resources are defined:

|       |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------|---------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 36715 | RLIMIT_CORE   | This is the maximum size of a <b>core</b> file, in bytes, that may be created by a process. A limit of 0 shall prevent the creation of a <b>core</b> file. If this limit is exceeded, the writing of a <b>core</b> file shall terminate at this size.                                                                                                                                                             |
| 36716 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36717 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36718 | RLIMIT_CPU    | This is the maximum amount of CPU time, in seconds, used by a process. If this limit is exceeded, SIGXCPU shall be generated for the process. If the process is catching or ignoring SIGXCPU, or all threads belonging to that process are blocking SIGXCPU, the behavior is unspecified.                                                                                                                         |
| 36719 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36720 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36721 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36722 | RLIMIT_DATA   | This is the maximum size of a data segment of the process, in bytes. If this limit is exceeded, the <i>malloc()</i> function shall fail with <i>errno</i> set to [ENOMEM].                                                                                                                                                                                                                                        |
| 36723 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36724 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36725 | RLIMIT_FSIZE  | This is the maximum size of a file, in bytes, that may be created by a process. If a write or truncate operation would cause this limit to be exceeded, SIGXFSZ shall be generated for the thread. If the thread is blocking, or the process is catching or ignoring SIGXFSZ, continued attempts to increase the size of a file from end-of-file to beyond the limit shall fail with <i>errno</i> set to [EFBIG]. |
| 36726 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36727 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36728 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36729 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36730 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36731 | RLIMIT_NOFILE | This is a number one greater than the maximum value that the system may assign to a newly-created descriptor. If this limit is exceeded, functions that allocate a file descriptor shall fail with <i>errno</i> set to [EMFILE]. This limit constrains the number of file descriptors that a process may allocate.                                                                                                |
| 36732 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36733 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36734 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 36735 |               |                                                                                                                                                                                                                                                                                                                                                                                                                   |

36736 RLIMIT\_STACK This is the maximum size of the initial thread's stack, in bytes. The  
 36737 implementation does not automatically grow the stack beyond this limit.  
 36738 If this limit is exceeded, SIGSEGV shall be generated for the thread. If the  
 36739 thread is blocking SIGSEGV, or the process is ignoring or catching  
 36740 SIGSEGV and has not made arrangements to use an alternate stack, the  
 36741 disposition of SIGSEGV shall be set to SIG\_DFL before it is generated.

36742 RLIMIT\_AS This is the maximum size of total available memory of the process, in  
 36743 bytes. If this limit is exceeded, the *malloc()* and *mmap()* functions shall fail  
 36744 with *errno* set to [ENOMEM]. In addition, the automatic stack growth  
 36745 fails with the effects outlined above.

36746 When using the *getrlimit()* function, if a resource limit can be represented correctly in an object  
 36747 of type **rlim\_t**, then its representation is returned; otherwise, if the value of the resource limit is  
 36748 equal to that of the corresponding saved hard limit, the value returned shall be  
 36749 RLIM\_SAVED\_MAX; otherwise, the value returned shall be RLIM\_SAVED\_CUR.

36750 When using the *setrlimit()* function, if the requested new limit is RLIM\_INFINITY, the new limit  
 36751 shall be "no limit"; otherwise, if the requested new limit is RLIM\_SAVED\_MAX, the new limit  
 36752 shall be the corresponding saved hard limit; otherwise, if the requested new limit is  
 36753 RLIM\_SAVED\_CUR, the new limit shall be the corresponding saved soft limit; otherwise, the  
 36754 new limit shall be the requested value. In addition, if the corresponding saved limit can be  
 36755 represented correctly in an object of type **rlim\_t** then it shall be overwritten with the new limit.

36756 The result of setting a limit to RLIM\_SAVED\_MAX or RLIM\_SAVED\_CUR is unspecified unless  
 36757 a previous call to *getrlimit()* returned that value as the soft or hard limit for the corresponding  
 36758 resource limit.

36759 The determination of whether a limit can be correctly represented in an object of type **rlim\_t** is  
 36760 implementation-defined. For example, some implementations permit a limit whose value is  
 36761 greater than RLIM\_INFINITY and others do not.

36762 The *exec* family of functions shall cause resource limits to be saved.

## 36763 RETURN VALUE

36764 Upon successful completion, *getrlimit()* and *setrlimit()* shall return 0. Otherwise, these functions  
 36765 shall return -1 and set *errno* to indicate the error.

## 36766 ERRORS

36767 The *getrlimit()* and *setrlimit()* functions shall fail if:

36768 [EINVAL] An invalid *resource* was specified; or in a *setrlimit()* call, the new *rlim\_cur*  
 36769 exceeds the new *rlim\_max*.

36770 [EPERM] The limit specified to *setrlimit()* would have raised the maximum limit value,  
 36771 and the calling process does not have appropriate privileges.

36772 The *setrlimit()* function may fail if:

36773 [EINVAL] The limit specified cannot be lowered because current usage is already higher  
 36774 than the limit.

**EXAMPLES**

None.

**APPLICATION USAGE**

If a process attempts to set the hard limit or soft limit for RLIMIT\_NOFILE to less than the value of {\_POSIX\_OPEN\_MAX} from <limits.h>, unexpected behavior may occur.

If a process attempts to set the hard limit or soft limit for RLIMIT\_NOFILE to less than the highest currently open file descriptor +1, unexpected behavior may occur.

**RATIONALE**

It should be noted that RLIMIT\_STACK applies “at least” to the stack of the initial thread in the process, and not to the sum of all the stacks in the process, as that would be very limiting unless the value is so big as to provide no value at all with a single thread.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*exec, fork(), malloc(), open(), sigaltstack(), sysconf(), ulimit()*

XBD <stropts.h>, <sys/resource.h>

**CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

An APPLICATION USAGE section is added.

Large File Summit extensions are added.

**Issue 6**

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/25 is applied, changing wording for RLIMIT\_NOFILE in the DESCRIPTION related to functions that allocate a file descriptor failing with [EMFILE]. Text is added to the APPLICATION USAGE section noting the consequences of a process attempting to set the hard or soft limit for RLIMIT\_NOFILE less than the highest currently open file descriptor +1.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/46 is applied, updating the definition of RLIMIT\_STACK in the DESCRIPTION from “the maximum size of a process stack” to “the maximum size of the initial thread’s stack”. Text is added to the RATIONALE section.

36806 **NAME**

36807 getrusage — get information about resource utilization

36808 **SYNOPSIS**

```
36809 XSI    #include <sys/resource.h>
36810        int getrusage(int who, struct rusage *r_usage);
```

36811 **DESCRIPTION**

36812 The *getrusage()* function shall provide measures of the resources used by the current process or  
 36813 its terminated and waited-for child processes. If the value of the *who* argument is  
 36814 RUSAGE\_SELF, information shall be returned about resources used by the current process. If the  
 36815 value of the *who* argument is RUSAGE\_CHILDREN, information shall be returned about  
 36816 resources used by the terminated and waited-for children of the current process. If the child is  
 36817 never waited for (for example, if the parent has SA\_NOCLDWAIT set or sets SIGCHLD to  
 36818 SIG\_IGN), the resource information for the child process is discarded and not included in the  
 36819 resource information provided by *getrusage()*.

36820 The *r\_usage* argument is a pointer to an object of type **struct rusage** in which the returned  
 36821 information is stored.

36822 **RETURN VALUE**

36823 Upon successful completion, *getrusage()* shall return 0; otherwise, -1 shall be returned and *errno*  
 36824 set to indicate the error.

36825 **ERRORS**

36826 The *getrusage()* function shall fail if:

36827 [EINVAL]        The value of the *who* argument is not valid.

36828 **EXAMPLES**36829 **Using getrusage()**

36830 The following example returns information about the resources used by the current process.

```
36831 #include <sys/resource.h>
36832 ...
36833 int who = RUSAGE_SELF;
36834 struct rusage usage;
36835 int ret;
36836
36837 ret = getrusage(who, &usage);
```

36837 **APPLICATION USAGE**

36838 None.

36839 **RATIONALE**

36840 None.

36841 **FUTURE DIRECTIONS**

36842 None.

36843 **SEE ALSO**

36844 *exit()*, *sigaction()*, *time()*, *times()*, *wait()*

36845 XBD [<sys/resource.h>](#)

36846 **CHANGE HISTORY**

36847 First released in Issue 4, Version 2.

36848 **Issue 5**

36849 Moved from X/OPEN UNIX extension to BASE.



36850 **NAME**36851 gets — get a string from a *stdin* stream36852 **SYNOPSIS**

```
36853 OB      #include <stdio.h>
36854      char *gets(char *s);
```

36855 **DESCRIPTION**

36856 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 36857 conflict between the requirements described here and the ISO C standard is unintentional. This  
 36858 volume of POSIX.1-2008 defers to the ISO C standard.

36859 The *gets()* function shall read bytes from the standard input stream, *stdin*, into the array pointed  
 36860 to by *s*, until a <newline> is read or an end-of-file condition is encountered. Any <newline> shall  
 36861 be discarded and a null byte shall be placed immediately after the last byte read into the array.

36862 CX The *gets()* function may mark the last data access timestamp of the file associated with *stream* for  
 36863 update. The last data access timestamp shall be marked for update by the first successful  
 36864 execution of *fgetc()*, *fgets()*, *fread()*, *fscanf()*, *getc()*, *getchar()*, *getdelim()*, *getline()*, *gets()*, or  
 36865 *scanf()* using *stream* that returns data not supplied by a prior call to *ungetc()*.

36866 **RETURN VALUE**

36867 Upon successful completion, *gets()* shall return *s*. If the end-of-file indicator for the stream is set,  
 36868 or if the stream is at end-of-file, the end-of-file indicator for the stream shall be set and *gets()*  
 36869 shall return a null pointer. If a read error occurs, the error indicator for the stream shall be set,  
 36870 CX *gets()* shall return a null pointer, and set *errno* to indicate the error.

36871 **ERRORS**36872 Refer to *fgetc()*.36873 **EXAMPLES**

36874 None.

36875 **APPLICATION USAGE**

36876 Reading a line that overflows the array pointed to by *s* results in undefined behavior. The use of  
 36877 *fgets()* is recommended.

36878 Since the user cannot specify the length of the buffer passed to *gets()*, use of this function is  
 36879 discouraged. The length of the string read is unlimited. It is possible to overflow this buffer in  
 36880 such a way as to cause applications to fail, or possible system security violations.

36881 Applications should use the *fgets()* function instead of the obsolescent *gets()* function.

36882 **RATIONALE**

36883 The standard developers decided to mark the *gets()* function as obsolescent even though it is in  
 36884 the ISO C standard due to the possibility of buffer overflow.

36885 **FUTURE DIRECTIONS**

36886 The *gets()* function may be removed in a future version.

36887 **SEE ALSO**36888 [Section 2.5](#) (on page 495), [feof\(\)](#), [ferror\(\)](#), [fgets\(\)](#)36889 XBD [<stdio.h>](#)

**CHANGE HISTORY**

36890 First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 6**

36892 Extensions beyond the ISO C standard are marked.

**Issue 7**

36894 Austin Group Interpretation 1003.1-2001 #051 is applied, clarifying the RETURN VALUE section.

36895 The *gets()* function is marked obsolescent.

36897 Changes are made related to support for finegrained timestamps.

36898 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0257 [14] is applied.

36899 **NAME**

36900        getservbyname, getservbyport, getservent — network services database functions

36901 **SYNOPSIS**

36902        #include &lt;netdb.h&gt;

36903        struct servent \*getservbyname(const char \*name, const char \*proto);

36904        struct servent \*getservbyport(int port, const char \*proto);

36905        struct servent \*getservent(void);

36906 **DESCRIPTION**36907        Refer to *endservent()*.

36908 **NAME**

36909        getsid — get the process group ID of a session leader

36910 **SYNOPSIS**

36911        #include &lt;unistd.h&gt;

36912        pid\_t getsid(pid\_t pid);

36913 **DESCRIPTION**36914        The *getsid()* function shall obtain the process group ID of the process that is the session leader of  
36915        the process specified by *pid*. If *pid* is (**pid\_t**)0, it specifies the calling process.36916 **RETURN VALUE**36917        Upon successful completion, *getsid()* shall return the process group ID of the session leader of  
36918        the specified process. Otherwise, it shall return  $-1$  and set *errno* to indicate the error.36919 **ERRORS**36920        The *getsid()* function shall fail if:36921        [EPERM]        The process specified by *pid* is not in the same session as the calling process,  
36922        and the implementation does not allow access to the process group ID of the  
36923        session leader of that process from the calling process.36924        [ESRCH]        There is no process with a process ID equal to *pid*.36925 **EXAMPLES**

36926        None.

36927 **APPLICATION USAGE**

36928        None.

36929 **RATIONALE**

36930        None.

36931 **FUTURE DIRECTIONS**

36932        None.

36933 **SEE ALSO**36934        *exec*, *fork()*, *getpid()*, *getpgid()*, *setpgid()*, *setsid()*

36935        XBD &lt;unistd.h&gt;

36936 **CHANGE HISTORY**

36937        First released in Issue 4, Version 2.

36938 **Issue 5**

36939        Moved from X/OPEN UNIX extension to BASE.

36940 **Issue 7**36941        The *getsid()* function is moved from the XSI option to the Base.

36942        POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0258 [421] is applied.

36943 **NAME**

36944 getsockname — get the socket name

36945 **SYNOPSIS**

36946 #include &lt;sys/socket.h&gt;

36947 int getsockname(int socket, struct sockaddr \*restrict address,  
36948 socklen\_t \*restrict address\_len);36949 **DESCRIPTION**36950 The *getsockname()* function shall retrieve the locally-bound name of the specified socket, store  
36951 this address in the **sockaddr** structure pointed to by the *address* argument, and store the length of  
36952 this address in the object pointed to by the *address\_len* argument.36953 The *address\_len* argument points to a **socklen\_t** object which on input specifies the length of the  
36954 supplied **sockaddr** structure, and on output specifies the length of the stored address. If the  
36955 actual length of the address is greater than the length of the supplied **sockaddr** structure, the  
36956 stored address shall be truncated.36957 If the socket has not been bound to a local name, the value stored in the object pointed to by  
36958 *address* is unspecified.36959 **RETURN VALUE**36960 Upon successful completion, 0 shall be returned, the *address* argument shall point to the address  
36961 of the socket, and the *address\_len* argument shall point to the length of the address. Otherwise, -1  
36962 shall be returned and *errno* set to indicate the error.36963 **ERRORS**36964 The *getsockname()* function shall fail if:36965 [EBADF] The *socket* argument is not a valid file descriptor.36966 [ENOTSOCK] The *socket* argument does not refer to a socket.

36967 [EOPNOTSUPP] The operation is not supported for this socket's protocol.

36968 The *getsockname()* function may fail if:

36969 [EINVAL] The socket has been shut down.

36970 [ENOBUFS] Insufficient resources were available in the system to complete the function.

36971 **EXAMPLES**

36972 None.

36973 **APPLICATION USAGE**

36974 None.

36975 **RATIONALE**

36976 None.

36977 **FUTURE DIRECTIONS**

36978 None.

36979 **SEE ALSO**36980 *accept()*, *bind()*, *getpeername()*, *socket()*

36981 XBD &lt;sys/socket.h&gt;

**CHANGE HISTORY**

- First released in Issue 6. Derived from the XNS, Issue 5.2 specification.
- The **restrict** keyword is added to the `getsockname()` prototype for alignment with the ISO/IEC 9899:1999 standard.
- Issue 7**
- POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0259 [464] is applied.

36988 **NAME**

36989       getsockopt — get the socket options

36990 **SYNOPSIS**

```
36991       #include <sys/socket.h>
36992       int getsockopt(int socket, int level, int option_name,
36993                      void *restrict option_value, socklen_t *restrict option_len);
```

36994 **DESCRIPTION**36995       The *getsockopt()* function manipulates options associated with a socket.

36996       The *getsockopt()* function shall retrieve the value for the option specified by the *option\_name*  
 36997       argument for the socket specified by the *socket* argument. If the size of the option value is greater  
 36998       than *option\_len*, the value stored in the object pointed to by the *option\_value* argument shall be  
 36999       silently truncated. Otherwise, the object pointed to by the *option\_len* argument shall be modified  
 37000       to indicate the actual length of the value.

37001       The *level* argument specifies the protocol level at which the option resides. To retrieve options at  
 37002       the socket level, specify the *level* argument as SOL\_SOCKET. To retrieve options at other levels,  
 37003       supply the appropriate level identifier for the protocol controlling the option. For example, to  
 37004       indicate that an option is interpreted by the TCP (Transmission Control Protocol), set *level* to  
 37005       IPPROTO\_TCP as defined in the **<netinet/in.h>** header.

37006       The socket in use may require the process to have appropriate privileges to use the *getsockopt()*  
 37007       function.

37008       The *option\_name* argument specifies a single option to be retrieved. It can be one of the socket-  
 37009       level options defined in **<sys/socket.h>** and described in [Section 2.10.16](#) (on page 528).

37010 **RETURN VALUE**

37011       Upon successful completion, *getsockopt()* shall return 0; otherwise, -1 shall be returned and *errno*  
 37012       set to indicate the error.

37013 **ERRORS**37014       The *getsockopt()* function shall fail if:

- |       |               |                                                                |
|-------|---------------|----------------------------------------------------------------|
| 37015 | [EBADF]       | The <i>socket</i> argument is not a valid file descriptor.     |
| 37016 | [EINVAL]      | The specified option is invalid at the specified socket level. |
| 37017 | [ENOPROTOOPT] | The option is not supported by the protocol.                   |
| 37018 |               |                                                                |
| 37019 | [ENOTSOCK]    | The <i>socket</i> argument does not refer to a socket.         |

37020       The *getsockopt()* function may fail if:

- |       |           |                                                                              |
|-------|-----------|------------------------------------------------------------------------------|
| 37021 | [EACCES]  | The calling process does not have appropriate privileges.                    |
| 37022 | [EINVAL]  | The socket has been shut down.                                               |
| 37023 | [ENOBUFS] | Insufficient resources are available in the system to complete the function. |

37024 **EXAMPLES**

37025 None.

37026 **APPLICATION USAGE**

37027 None.

37028 **RATIONALE**

37029 None.

37030 **FUTURE DIRECTIONS**

37031 None.

37032 **SEE ALSO**37033 [Section 2.10.16](#) (on page 528), [bind\(\)](#), [close\(\)](#), [endprotoent\(\)](#), [setsockopt\(\)](#), [socket\(\)](#)37034 XBD [<sys/socket.h>](#), [<netinet/in.h>](#)37035 **CHANGE HISTORY**

37036 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

37037 The **restrict** keyword is added to the `getsockopt()` prototype for alignment with the  
37038 ISO/IEC 9899:1999 standard.37039 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/47 is applied, updating the description of  
37040 SO\_LINGER in the DESCRIPTION so that it blocks the calling thread rather than the process.37041 **Issue 7**37042 Austin Group Interpretation 1003.1-2001 #158 is applied, removing text relating to socket options  
37043 that is now in [Section 2.10.16](#) (on page 528).



37044 **NAME**

37045       getsubopt — parse suboption arguments from a string

37046 **SYNOPSIS**

```
37047 CX      #include <stdlib.h>
37048          int getsubopt(char **optionp, char * const *keylistp, char **valuep);
```

37049 **DESCRIPTION**

37050       The *getsubopt()* function shall parse suboption arguments in a flag argument. Such options often  
 37051       result from the use of *getopt()*.

37052       The *getsubopt()* argument *optionp* is a pointer to a pointer to the option argument string. The  
 37053       suboption arguments shall be separated by <comma> characters and each may consist of either  
 37054       a single token, or a token-value pair separated by an <equals-sign>.

37055       The *keylistp* argument shall be a pointer to a vector of strings. The end of the vector is identified  
 37056       by a null pointer. Each entry in the vector is one of the possible tokens that might be found in  
 37057       *\*optionp*. Since <comma> characters delimit suboption arguments in *optionp*, they should not  
 37058       appear in any of the strings pointed to by *keylistp*. Similarly, because an <equals-sign> separates  
 37059       a token from its value, the application should not include an <equals-sign> in any of the strings  
 37060       pointed to by *keylistp*. The *getsubopt()* function shall not modify the *keylistp* vector.

37061       The *valuep* argument is the address of a value string pointer.

37062       If a <comma> appears in *optionp*, it shall be interpreted as a suboption separator. After <comma>  
 37063       characters have been processed, if there are one or more <equals-sign> characters in a suboption  
 37064       string, the first <equals-sign> in any suboption string shall be interpreted as a separator between  
 37065       a token and a value. Subsequent <equals-sign> characters in a suboption string shall be  
 37066       interpreted as part of the value.

37067       If the string at *\*optionp* contains only one suboption argument (equivalently, no <comma>  
 37068       characters), *getsubopt()* shall update *\*optionp* to point to the null character at the end of the  
 37069       string. Otherwise, it shall isolate the suboption argument by replacing the <comma> separator  
 37070       with a null character, and shall update *\*optionp* to point to the start of the next suboption  
 37071       argument. If the suboption argument has an associated value (equivalently, contains an <equals-  
 37072       sign>), *getsubopt()* shall update *\*valuep* to point to the value's first character. Otherwise, it shall  
 37073       set *\*valuep* to a null pointer. The calling application may use this information to determine  
 37074       whether the presence or absence of a value for the suboption is an error.

37075       Additionally, when *getsubopt()* fails to match the suboption argument with a token in the *keylistp*  
 37076       array, the calling application should decide if this is an error, or if the unrecognized option  
 37077       should be processed in another way.

37078 **RETURN VALUE**

37079       The *getsubopt()* function shall return the index of the matched token string, or -1 if no token  
 37080       strings were matched.

37081 **ERRORS**

37082       No errors are defined.

## 37083 EXAMPLES

## 37084 Parsing Suboptions

37085 The following example uses the `getsubopt()` function to parse a *value* argument in the *optarg*  
37086 external variable returned by a call to `getopt()`.

```
37087 #include <stdio.h>
37088 #include <stdlib.h>
37089 #include <unistd.h>
37090
37091 int do_all;
37092 const char *type;
37093 int read_size;
37094 int write_size;
37095 int read_only;
37096
37097 enum
37098 {
37099     RO_OPTION = 0,
37100     RW_OPTION,
37101     READ_SIZE_OPTION,
37102     WRITE_SIZE_OPTION
37103 };
37104
37105 const char *mount_opts[] =
37106 {
37107     [RO_OPTION] = "ro",
37108     [RW_OPTION] = "rw",
37109     [READ_SIZE_OPTION] = "rsize",
37110     [WRITE_SIZE_OPTION] = "wsize",
37111     NULL
37112 };
37113
37114 int
37115 main(int argc, char *argv[])
37116 {
37117     char *subopts, *value;
37118     int opt;
37119
37120     while ((opt = getopt(argc, argv, "at:o:")) != -1)
37121         switch(opt)
37122         {
37123             case 'a':
37124                 do_all = 1;
37125                 break;
37126             case 't':
37127                 type = optarg;
37128                 break;
37129             case 'o':
37130                 subopts = optarg;
37131                 while (*subopts != ' ')
37132                 {
37133                     char *saved = subopts;
37134                     switch(getsubopt(&subopts, (char **)mount_opts,
```

```

37130         &value))
37131     {
37132     case RO_OPTION:
37133         read_only = 1;
37134         break;
37135     case RW_OPTION:
37136         read_only = 0;
37137         break;
37138     case READ_SIZE_OPTION:
37139         if (value == NULL)
37140             abort();
37141         read_size = atoi(value);
37142         break;
37143     case WRITE_SIZE_OPTION:
37144         if (value == NULL)
37145             abort();
37146         write_size = atoi(value);
37147         break;
37148     default:
37149         /* Unknown suboption. */
37150         printf("Unknown suboption '%s'\n", saved);
37151         abort();
37152     }
37153 }
37154 break;
37155 default:
37156     abort();
37157 }
37158
37159     /* Do the real work. */
37160     return 0;
37161 }

```

If the above example is invoked with:

```
program -o ro,rsiz=512
```

then after option parsing, the variable *do\_all* will be 0, *type* will be a null pointer, *read\_size* will be 512, *write\_size* will be 0, and *read\_only* will be 1. If it is invoked with:

```
program -o oops
```

it will print:

```
"Unknown suboption 'oops'"
```

before aborting.

## APPLICATION USAGE

The value of *\*valuep* when *getsubopt()* returns -1 is unspecified. Historical implementations provide various incompatible extensions to allow an application to access the suboption text that was not found in the *keylistp* array.

**RATIONALE**

The *keylistp* argument of *getsubopt()* is typed as **char \* const \*** to match historical practice. However, the standard is clear that implementations will not modify either the array or the strings contained in the array, as if the argument had been typed **const char \* const \***.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*getopt()*

XBD <stdlib.h>

**CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

**Issue 6**

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/26 is applied, correcting an editorial error in the SYNOPSIS.

**Issue 7**

The *getsubopt()* function is moved from the XSI option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0260 [196], XSH/TC1-2008/0261 [196], XSH/TC1-2008/0262 [196], XSH/TC1-2008/0263 [196], XSH/TC1-2008/0264 [196], XSH/TC1-2008/0265 [196], XSH/TC1-2008/0266 [196], XSH/TC1-2008/0267 [196], XSH/TC1-2008/0268 [196], and XSH/TC1-2008/0269 [196] are applied.

37195 **NAME**

37196       gettimeofday — get the date and time

37197 **SYNOPSIS**

```
37198 OB XSI #include <sys/time.h>
37199 int gettimeofday(struct timeval *restrict tp, void *restrict tzp);
```

37200 **DESCRIPTION**

37201       The *gettimeofday()* function shall obtain the current time, expressed as seconds and microseconds  
37202       since the Epoch, and store it in the **timeval** structure pointed to by *tp*. The resolution of the  
37203       system clock is unspecified.

37204       If *tzp* is not a null pointer, the behavior is unspecified.

37205 **RETURN VALUE**

37206       The *gettimeofday()* function shall return 0 and no value shall be reserved to indicate an error.

37207 **ERRORS**

37208       No errors are defined.

37209 **EXAMPLES**

37210       None.

37211 **APPLICATION USAGE**

37212       Applications should use the *clock\_gettime()* function instead of the obsolescent *gettimeofday()*  
37213       function.

37214 **RATIONALE**

37215       None.

37216 **FUTURE DIRECTIONS**

37217       The *gettimeofday()* function may be removed in a future version.

37218 **SEE ALSO**

37219       *clock\_getres()*, *ctime()*

37220       XBD **<sys/time.h>**

37221 **CHANGE HISTORY**

37222       First released in Issue 4, Version 2.

37223 **Issue 5**

37224       Moved from X/OPEN UNIX extension to BASE.

37225 **Issue 6**

37226       The DESCRIPTION is updated to refer to “seconds since the Epoch” rather than “seconds since  
37227       00:00:00 UTC (Coordinated Universal Time), January 1 1970” for consistency with other *time*  
37228       functions.

37229       The **restrict** keyword is added to the *gettimeofday()* prototype for alignment with the  
37230       ISO/IEC 9899:1999 standard.

37231 **Issue 7**

37232       The *gettimeofday()* function is marked obsolescent.

37233 **NAME**

37234       getuid — get a real user ID

37235 **SYNOPSIS**

37236       #include &lt;unistd.h&gt;

37237       uid\_t getuid(void);

37238 **DESCRIPTION**

37239       The *getuid()* function shall return the real user ID of the calling process. The *getuid()* function +  
 37240       shall not modify *errno*.

37241 **RETURN VALUE**

37242       The *getuid()* function shall always be successful and no return value is reserved to indicate the  
 37243       error.

37244 **ERRORS**

37245       No errors are defined.

37246 **EXAMPLES**37247       **Setting the Effective User ID to the Real User ID**

37248       The following example sets the effective user ID of the calling process to the real user ID. |

37249       #include &lt;unistd.h&gt;

37250       ...

37251       seteuid(getuid()); +

37252 **APPLICATION USAGE**

37253       None.

37254 **RATIONALE**

37255       In a conforming environment, *getuid()* will always succeed. It is possible for implementations to |  
 37256       provide an extension where a process in a non-conforming environment will not be associated |  
 37257       with a user or group ID. It is recommended that such implementations return (**uid\_t**)–1 and set |  
 37258       *errno* to indicate such an environment; doing so does not violate this standard, since such an |  
 37259       environment is already an extension.

37260 **FUTURE DIRECTIONS**

37261       None.

37262 **SEE ALSO**37263       *getegid()*, *geteuid()*, *getgid()*, *setegid()*, *seteuid()*, *setgid()*, *setregid()*, *setreuid()*, *setuid()*

37264       XBD &lt;sys/types.h&gt;, &lt;unistd.h&gt;

37265 **CHANGE HISTORY**

37266       First released in Issue 1. Derived from Issue 1 of the SVID.

37267 **Issue 6**

37268       In the SYNOPSIS, the optional include of the &lt;sys/types.h&gt; header is removed.

37269       The following new requirements on POSIX implementations derive from alignment with the  
 37270       Single UNIX Specification:

- 37271       • The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was  
 37272       required for conforming implementations of previous POSIX specifications, it was not  
 37273       required for UNIX applications.

37274 **NAME**

37275 getutxent, getutxid, getutxline — get user accounting database entries

37276 **SYNOPSIS**

```
37277 XSI      #include <utmpx.h>
37278          struct utmpx *getutxent(void);
37279          struct utmpx *getutxid(const struct utmpx *id);
37280          struct utmpx *getutxline(const struct utmpx *line);
```

37281 **DESCRIPTION**37282 Refer to *endutxent()*.

37283 **NAME**

37284 getwc — get a wide character from a stream

37285 **SYNOPSIS**

37286 #include &lt;stdio.h&gt;

37287 #include &lt;wchar.h&gt;

37288 wint\_t getwc(FILE \*stream);

37289 **DESCRIPTION**

37290 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
37291 conflict between the requirements described here and the ISO C standard is unintentional. This  
37292 volume of POSIX.1-2008 defers to the ISO C standard.

37293 The *getwc()* function shall be equivalent to *fgetwc()*, except that if it is implemented as a macro it  
37294 may evaluate *stream* more than once, so the argument should never be an expression with side-  
37295 effects.

37296 **RETURN VALUE**37297 Refer to *fgetwc()*.37298 **ERRORS**37299 Refer to *fgetwc()*.37300 **EXAMPLES**

37301 None.

37302 **APPLICATION USAGE**

37303 Since it may be implemented as a macro, *getwc()* may treat incorrectly a *stream* argument with  
37304 side-effects. In particular, *getwc(\*f++)* does not necessarily work as expected. Therefore, use of  
37305 this function is not recommended; *fgetwc()* should be used instead.

37306 **RATIONALE**

37307 None.

37308 **FUTURE DIRECTIONS**

37309 None.

37310 **SEE ALSO**37311 [Section 2.5](#) (on page 495), *fgetwc()*37312 XBD [<stdio.h>](#), [<wchar.h>](#)37313 **CHANGE HISTORY**

37314 First released as a World-wide Portability Interface in Issue 4. Derived from the MSE working  
37315 draft.

37316 **Issue 5**37317 The Optional Header (OH) marking is removed from [<stdio.h>](#).37318 **Issue 7**

37319 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0270 [14] is applied.



37320 **NAME**37321       getwchar — get a wide character from a *stdin* stream37322 **SYNOPSIS**

37323       #include &lt;wchar.h&gt;

37324       wint\_t getwchar(void);

37325 **DESCRIPTION**

37326 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
37327       conflict between the requirements described here and the ISO C standard is unintentional. This  
37328       volume of POSIX.1-2008 defers to the ISO C standard.

37329       The *getwchar()* function shall be equivalent to *getwc(stdin)*.37330 **RETURN VALUE**37331       Refer to *fgetwc()*.37332 **ERRORS**37333       Refer to *fgetwc()*.37334 **EXAMPLES**

37335       None.

37336 **APPLICATION USAGE**

37337       If the **wint\_t** value returned by *getwchar()* is stored into a variable of type **wchar\_t** and then  
37338       compared against the **wint\_t** macro WEOF, the result may be incorrect. Only the **wint\_t** type is  
37339       guaranteed to be able to represent any wide character and WEOF.

37340 **RATIONALE**

37341       None.

37342 **FUTURE DIRECTIONS**

37343       None.

37344 **SEE ALSO**37345       Section 2.5 (on page 495), *fgetwc()*, *getwc()*

37346       XBD &lt;wchar.h&gt;

37347 **CHANGE HISTORY**

37348       First released as a World-wide Portability Interface in Issue 4. Derived from the MSE working  
37349       draft.

37350 **Issue 7**

37351       POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0271 [14] is applied.

**NAME**

glob, globfree — generate pathnames matching a pattern

**SYNOPSIS**

```
#include <glob.h>

int glob(const char *restrict pattern, int flags,
         int (*errfunc)(const char *epath, int eerrno),
         glob_t *restrict pglob);
void globfree(glob_t *pglob);
```

**DESCRIPTION**

The `glob()` function is a pathname generator that shall implement the rules defined in XCU [Section 2.13](#) (on page 2372), with optional support for rule 3 in XCU [Section 2.13.3](#) (on page 2373).

The structure type `glob_t` is defined in `<glob.h>` and includes at least the following members:

| Member Type          | Member Name           | Description                                                  |
|----------------------|-----------------------|--------------------------------------------------------------|
| <code>size_t</code>  | <code>gl_pathc</code> | Count of paths matched by <i>pattern</i> .                   |
| <code>char **</code> | <code>gl_pathv</code> | Pointer to a list of matched pathnames.                      |
| <code>size_t</code>  | <code>gl_offs</code>  | Slots to reserve at the beginning of <code>gl_pathv</code> . |

The argument *pattern* is a pointer to a pathname pattern to be expanded. The `glob()` function shall match all accessible pathnames against this pattern and develop a list of all pathnames that match. In order to have access to a pathname, `glob()` requires search permission on every component of a path except the last, and read permission on each directory of any filename component of *pattern* that contains any of the following special characters: `'*'`, `'?'`, and `'['`.

The `glob()` function shall store the number of matched pathnames into `pglob->gl_pathc` and a pointer to a list of pointers to pathnames into `pglob->gl_pathv`. The pathnames shall be in sort order as defined by the current setting of the `LC_COLLATE` category; see XBD [Section 7.3.2](#) (on page 147). The first pointer after the last pathname shall be a null pointer. If the pattern does not match any pathnames, the returned number of matched paths is set to 0, and the contents of `pglob->gl_pathv` are implementation-defined.

It is the caller's responsibility to create the structure pointed to by *pglob*. The `glob()` function shall allocate other space as needed, including the memory pointed to by `gl_pathv`. The `globfree()` function shall free any space associated with *pglob* from a previous call to `glob()`.

The *flags* argument is used to control the behavior of `glob()`. The value of *flags* is a bitwise-inclusive OR of zero or more of the following constants, which are defined in `<glob.h>`:

|                          |                                                                                                                                                                                                                                                                                                                                                                                                                 |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>GLOB_APPEND</code> | Append pathnames generated to the ones from a previous call to <code>glob()</code> .                                                                                                                                                                                                                                                                                                                            |
| <code>GLOB_DOOFFS</code> | Make use of <code>pglob-&gt;gl_offs</code> . If this flag is set, <code>pglob-&gt;gl_offs</code> is used to specify how many null pointers to add to the beginning of <code>pglob-&gt;gl_pathv</code> . In other words, <code>pglob-&gt;gl_pathv</code> shall point to <code>pglob-&gt;gl_offs</code> null pointers, followed by <code>pglob-&gt;gl_pathc</code> pathname pointers, followed by a null pointer. |
| <code>GLOB_ERR</code>    | Cause <code>glob()</code> to return when it encounters a directory that it cannot open or read. Ordinarily, <code>glob()</code> continues to find matches.                                                                                                                                                                                                                                                      |
| <code>GLOB_MARK</code>   | Each pathname that is a directory that matches <i>pattern</i> shall have a <code>&lt;slash&gt;</code> appended.                                                                                                                                                                                                                                                                                                 |

GLOB\_NOCHECK Supports rule 3 in XCU [Section 2.13.3](#) (on page 2373). If *pattern* does not match any pathname, then *glob()* shall return a list consisting of only *pattern*, and the number of matched pathnames is 1.

GLOB\_NOESCAPE Disable backslash escaping.

GLOB\_NOSORT Ordinarily, *glob()* sorts the matching pathnames according to the current setting of the *LC\_COLLATE* category; see XBD [Section 7.3.2](#) (on page 147). When this flag is used, the order of pathnames returned is unspecified.

The GLOB\_APPEND flag can be used to append a new set of pathnames to those found in a previous call to *glob()*. The following rules apply to applications when two or more calls to *glob()* are made with the same value of *pglob* and without intervening calls to *globfree()*:

1. The first such call shall not set GLOB\_APPEND. All subsequent calls shall set it.
2. All the calls shall set GLOB\_DOOFFS, or all shall not set it.
3. After the second call, *pglob->gl\_pathv* points to a list containing the following:
  - a. Zero or more null pointers, as specified by GLOB\_DOOFFS and *pglob->gl\_offs*.
  - b. Pointers to the pathnames that were in the *pglob->gl\_pathv* list before the call, in the same order as before.
  - c. Pointers to the new pathnames generated by the second call, in the specified order.
4. The count returned in *pglob->gl\_pathc* shall be the total number of pathnames from the two calls.
5. The application can change any of the fields after a call to *glob()*. If it does, the application shall reset them to the original value before a subsequent call, using the same *pglob* value, to *globfree()* or *glob()* with the GLOB\_APPEND flag.

If, during the search, a directory is encountered that cannot be opened or read and *errfunc* is not a null pointer, *glob()* calls *(\*errfunc())* with two arguments:

1. The *epath* argument is a pointer to the path that failed.
2. The *eerrno* argument is the value of *errno* from the failure, as set by *opendir()*, *readdir()*, or *stat()*. (Other values may be used to report other errors not explicitly documented for those functions.)

If *(\*errfunc())* is called and returns non-zero, or if the GLOB\_ERR flag is set in *flags*, *glob()* shall stop the scan and return GLOB\_ABORTED after setting *gl\_pathc* and *gl\_pathv* in *pglob* to reflect the paths already scanned. If GLOB\_ERR is not set and either *errfunc* is a null pointer or *(\*errfunc())* returns 0, the error shall be ignored.

The *glob()* function shall not fail because of large files.

## RETURN VALUE

Upon successful completion, *glob()* shall return 0. The argument *pglob->gl\_pathc* shall return the number of matched pathnames and the argument *pglob->gl\_pathv* shall contain a pointer to a null-terminated list of matched and sorted pathnames. However, if *pglob->gl\_pathc* is 0, the content of *pglob->gl\_pathv* is undefined.

The *globfree()* function shall not return a value.

If *glob()* terminates due to an error, it shall return one of the non-zero constants defined in **<glob.h>**. The arguments *pglob->gl\_pathc* and *pglob->gl\_pathv* are still set as defined above.

**ERRORS**

The *glob()* function shall fail and return the corresponding value if:

**GLOB\_ABORTED** The scan was stopped because **GLOB\_ERR** was set or *(\*errfunc())* returned non-zero.

**GLOB\_NOMATCH** The pattern does not match any existing pathname, and **GLOB\_NOCHECK** was not set in flags.

**GLOB\_NOSPACE** An attempt to allocate memory failed.

**EXAMPLES**

One use of the **GLOB\_DOOFFS** flag is by applications that build an argument list for use with *execv()*, *execve()*, or *execvp()*. Suppose, for example, that an application wants to do the equivalent of:

```
ls -l *.c
```

but for some reason:

```
system("ls -l *.c")
```

is not acceptable. The application could obtain approximately the same result using the sequence:

```
globbuf.gl_offs = 2;
glob("*.c", GLOB_DOOFFS, NULL, &globbuf);
globbuf.gl_pathv[0] = "ls";
globbuf.gl_pathv[1] = "-l";
execvp("ls", &globbuf.gl_pathv[0]);
```

Using the same example:

```
ls -l *.c *.h
```

could be approximately simulated using **GLOB\_APPEND** as follows:

```
globbuf.gl_offs = 2;
glob("*.c", GLOB_DOOFFS, NULL, &globbuf);
glob("*.h", GLOB_DOOFFS|GLOB_APPEND, NULL, &globbuf);
...
```

**APPLICATION USAGE**

This function is not provided for the purpose of enabling utilities to perform pathname expansion on their arguments, as this operation is performed by the shell, and utilities are explicitly not expected to redo this. Instead, it is provided for applications that need to do pathname expansion on strings obtained from other sources, such as a pattern typed by a user or read from a file.

If a utility needs to see if a pathname matches a given pattern, it can use *fnmatch()*.

Note that *gl\_pathc* and *gl\_pathv* have meaning even if *glob()* fails. This allows *glob()* to report partial results in the event of an error. However, if *gl\_pathc* is 0, *gl\_pathv* is unspecified even if *glob()* did not return an error.

The **GLOB\_NOCHECK** option could be used when an application wants to expand a pathname if wildcards are specified, but wants to treat the pattern as just a string otherwise. The *sh* utility might use this for option-arguments, for example.

The new pathnames generated by a subsequent call with **GLOB\_APPEND** are not sorted together with the previous pathnames. This mirrors the way that the shell handles pathname

expansion when multiple expansions are done on a command line.

Applications that need tilde and parameter expansion should use *wordexp()*.

#### RATIONALE

It was claimed that the GLOB\_DOOFFS flag is unnecessary because it could be simulated using:

```
new = (char **)malloc((n + pglob->gl_pathc + 1)
    * sizeof(char *));
(void) memcpy(new+n, pglob->gl_pathv,
    pglob->gl_pathc * sizeof(char *));
(void) memset(new, 0, n * sizeof(char *));
free(pglob->gl_pathv);
pglob->gl_pathv = new;
```

However, this assumes that the memory pointed to by *gl\_pathv* is a block that was separately created using *malloc()*. This is not necessarily the case. An application should make no assumptions about how the memory referenced by fields in *pglob* was allocated. It might have been obtained from *malloc()* in a large chunk and then carved up within *glob()*, or it might have been created using a different memory allocator. It is not the intent of the standard developers to specify or imply how the memory used by *glob()* is managed.

The GLOB\_APPEND flag would be used when an application wants to expand several different patterns into a single list.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*exec*, *fdopendir()*, *fnmatch()*, *fstatat()*, *readdir()*, Section 2.6

XBD Section 7.3.2 (on page 147), **<glob.h>**

#### CHANGE HISTORY

First released in Issue 4. Derived from the ISO POSIX-2 standard.

##### Issue 5

Moved from POSIX2 C-language Binding to BASE.

##### Issue 6

The normative text is updated to avoid use of the term “must” for application requirements.

The **restrict** keyword is added to the *glob()* prototype for alignment with the ISO/IEC 9899:1999 standard.

## 37512 NAME

37513 gmtime, gmtime\_r — convert a time value to a broken-down UTC time

## 37514 SYNOPSIS

```
37515 #include <time.h>
37516 struct tm *gmtime(const time_t *timer);
37517 CX struct tm *gmtime_r(const time_t *restrict timer,
37518 struct tm *restrict result);
```

## 37519 DESCRIPTION

37520 CX For *gmtime()*: The functionality described on this reference page is aligned with the ISO C  
37521 standard. Any conflict between the requirements described here and the ISO C standard is  
37522 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

37523 The *gmtime()* function shall convert the time in seconds since the Epoch pointed to by *timer* into  
37524 a broken-down time, expressed as Coordinated Universal Time (UTC).

37525 CX The relationship between a time in seconds since the Epoch used as an argument to *gmtime()*  
37526 and the **tm** structure (defined in the **<time.h>** header) is that the result shall be as specified in  
37527 the expression given in the definition of seconds since the Epoch (see XBD [Section 4.16](#), on page  
37528 113), where the names in the structure and in the expression correspond.

37529 The same relationship shall apply for *gmtime\_r()*.

37530 The *gmtime()* function need not be thread-safe.

37531 The *asctime()*, *ctime()*, *gmtime()*, and *localtime()* functions shall return values in one of two static  
37532 objects: a broken-down time structure and an array of type **char**. Execution of any of the  
37533 functions may overwrite the information returned in either of these objects by any of the other  
37534 functions.

37535 The *gmtime\_r()* function shall convert the time in seconds since the Epoch pointed to by *timer*  
37536 into a broken-down time expressed as Coordinated Universal Time (UTC). The broken-down  
37537 time is stored in the structure referred to by *result*. The *gmtime\_r()* function shall also return the  
37538 address of the same structure.

## 37539 RETURN VALUE

37540 Upon successful completion, the *gmtime()* function shall return a pointer to a **struct tm**. If an  
37541 CX error is detected, *gmtime()* shall return a null pointer and set *errno* to indicate the error.

37542 Upon successful completion, *gmtime\_r()* shall return the address of the structure pointed to by  
37543 the argument *result*. If an error is detected, *gmtime\_r()* shall return a null pointer and set *errno* to  
37544 indicate the error.

## 37545 ERRORS

37546 CX The *gmtime()* and *gmtime\_r()* functions shall fail if:

37547 CX [EOVERFLOW] The result cannot be represented.

**EXAMPLES**

None.

**APPLICATION USAGE**

The *gmtime\_r()* function is thread-safe and returns values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*asctime()*, *clock()*, *ctime()*, *difftime()*, *localtime()*, *mktime()*, *strftime()*, *strptime()*, *time()*, *utime()*

XBD Section 4.16 (on page 113), [<time.h>](#)

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

A note indicating that the *gmtime()* function need not be reentrant is added to the DESCRIPTION.

The *gmtime\_r()* function is included for alignment with the POSIX Threads Extension.

**Issue 6**

The *gmtime\_r()* function is marked as part of the Thread-Safe Functions option.

Extensions beyond the ISO C standard are marked.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

The **restrict** keyword is added to the *gmtime\_r()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/27 is applied, adding the [EOVERFLOW] error.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/48 is applied, updating the error handling for *gmtime\_r()*.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #156 is applied.

The *gmtime\_r()* function is moved from the Thread-Safe Functions option to the Base.



37580 **NAME**

37581 grantpt — grant access to the slave pseudo-terminal device

37582 **SYNOPSIS**

```
37583 XSI      #include <stdlib.h>
37584         int grantpt(int fildes);
```

37585 **DESCRIPTION**

37586 The *grantpt()* function shall change the mode and ownership of the slave pseudo-terminal  
37587 device associated with its master pseudo-terminal counterpart. The *fildes* argument is a file  
37588 descriptor that refers to a master pseudo-terminal device. The user ID of the slave shall be set to  
37589 the real UID of the calling process and the group ID shall be set to an unspecified group ID. The  
37590 permission mode of the slave pseudo-terminal shall be set to readable and writable by the  
37591 owner, and writable by the group.

37592 The behavior of the *grantpt()* function is unspecified if the application has installed a signal  
37593 handler to catch SIGCHLD signals.

37594 **RETURN VALUE**

37595 Upon successful completion, *grantpt()* shall return 0; otherwise, it shall return -1 and set *errno* to  
37596 indicate the error.

37597 **ERRORS**

37598 The *grantpt()* function may fail if:

|       |          |                                                                                    |   |
|-------|----------|------------------------------------------------------------------------------------|---|
| 37599 | [EACCES] | The corresponding slave pseudo-terminal device could not be accessed.              | + |
| 37600 | [EBADF]  | The <i>fildes</i> argument is not a valid open file descriptor.                    |   |
| 37601 | [EINVAL] | The <i>fildes</i> argument is not associated with a master pseudo-terminal device. | - |

37602 **EXAMPLES**

37603 None.

37604 **APPLICATION USAGE**

37605 None.

37606 **RATIONALE**

37607 See the RATIONALE section for *posix\_openpt()*.

37608 **FUTURE DIRECTIONS**

37609 None.

37610 **SEE ALSO**

37611 *open()*, *posix\_openpt()*, *ptsname()*, *unlockpt()*

37612 XBD *<stdlib.h>*

37613 **CHANGE HISTORY**

37614 First released in Issue 4, Version 2.

37615 **Issue 5**

37616 Moved from X/OPEN UNIX extension to BASE.

37617 The last paragraph of the DESCRIPTION is moved from the APPLICATION USAGE section.



37618 **Issue 7**  
37619

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0272 [96] is applied.

**NAME**

hcreate, hdestroy, hsearch — manage hash search table

**SYNOPSIS**

```
XSI    #include <search.h>

int hcreate(size_t nel);
void hdestroy(void);
ENTRY *hsearch(ENTRY item, ACTION action);
```

**DESCRIPTION**

The *hcreate()*, *hdestroy()*, and *hsearch()* functions shall manage hash search tables.

The *hcreate()* function shall allocate sufficient space for the table, and the application shall ensure it is called before *hsearch()* is used. The *nel* argument is an estimate of the maximum number of entries that the table shall contain. This number may be adjusted upward by the algorithm in order to obtain certain mathematically favorable circumstances.

The *hdestroy()* function shall dispose of the search table, and may be followed by another call to *hcreate()*. After the call to *hdestroy()*, the data can no longer be considered accessible.

The *hsearch()* function is a hash-table search routine. It shall return a pointer into a hash table indicating the location at which an entry can be found. The *item* argument is a structure of type **ENTRY** (defined in the **<search.h>** header) containing two pointers: *item.key* points to the comparison key (a **char \***), and *item.data* (a **void \***) points to any other data to be associated with that key. The comparison function used by *hsearch()* is *strcmp()*. The *action* argument is a member of an enumeration type **ACTION** indicating the disposition of the entry if it cannot be found in the table. **ENTER** indicates that the item should be inserted in the table at an appropriate point. **FIND** indicates that no entry should be made. Unsuccessful resolution is indicated by the return of a null pointer.

These functions need not be thread-safe.

**RETURN VALUE**

The *hcreate()* function shall return 0 if it cannot allocate sufficient space for the table; otherwise, it shall return non-zero.

The *hdestroy()* function shall not return a value.

The *hsearch()* function shall return a null pointer if either the action is **FIND** and the item could not be found or the action is **ENTER** and the table is full.

**ERRORS**

The *hcreate()* and *hsearch()* functions may fail if:

[ENOMEM]      Insufficient storage space is available.

**EXAMPLES**

The following example reads in strings followed by two numbers and stores them in a hash table, discarding duplicates. It then reads in strings and finds the matching entry in the hash table and prints it out.

```
#include <stdio.h>
#include <search.h>
#include <string.h>

struct info {          /* This is the info stored in the table */
    int age, room;      /* other than the key. */
};
```

```

37664     #define NUM_EMPL      5000      /* # of elements in search table. */
37665     int main(void)
37666     {
37667         char string_space[NUM_EMPL*20]; /* Space to store strings. */
37668         struct info info_space[NUM_EMPL]; /* Space to store employee info. */
37669         char *str_ptr = string_space; /* Next space in string_space. */
37670         struct info *info_ptr = info_space;
37671                                     /* Next space in info_space. */
37672         ENTRY item;
37673         ENTRY *found_item; /* Name to look for in table. */
37674         char name_to_find[30];
37675
37676         int i = 0;
37677
37678         /* Create table; no error checking is performed. */
37679         (void) hcreate(NUM_EMPL);
37680         while (scanf("%s%d%d", str_ptr, &info_ptr->age,
37681                     &info_ptr->room) != EOF && i++ < NUM_EMPL) {
37682             /* Put information in structure, and structure in item. */
37683             item.key = str_ptr;
37684             item.data = info_ptr;
37685             str_ptr += strlen(str_ptr) + 1;
37686             info_ptr++;
37687
37688             /* Put item into table. */
37689             (void) hsearch(item, ENTER);
37690         }
37691
37692         /* Access table. */
37693         item.key = name_to_find;
37694         while (scanf("%s", item.key) != EOF) {
37695             if ((found_item = hsearch(item, FIND)) != NULL) {
37696                 /* If item is in the table. */
37697                 (void)printf("found %s, age = %d, room = %d\n",
37698                             found_item->key,
37699                             ((struct info *)found_item->data)->age,
37700                             ((struct info *)found_item->data)->room);
37701             } else
37702                 (void)printf("no such employee %s\n", name_to_find);
37703         }
37704         return 0;
37705     }

```

**APPLICATION USAGE**

The *hcreate()* and *hsearch()* functions may use *malloc()* to allocate space.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

37708 **SEE ALSO**37709 *bsearch()*, *lsearch()*, *malloc()*, *strcmp()*, *tdelete()*

37710 XBD &lt;search.h&gt;

37711 **CHANGE HISTORY**

37712 First released in Issue 1. Derived from Issue 1 of the SVID.

37713 **Issue 6**

37714 The normative text is updated to avoid use of the term “must” for application requirements.

37715 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

37716 **Issue 7**

37717 Austin Group Interpretation 1003.1-2001 #156 is applied.

**NAME**

htonl, htons, ntohl, ntohs — convert values between host and network byte order

**SYNOPSIS**

```
#include <arpa/inet.h>

uint32_t htonl(uint32_t hostlong);
uint16_t htons(uint16_t hostshort);
uint32_t ntohl(uint32_t netlong);
uint16_t ntohs(uint16_t netshort);
```

**DESCRIPTION**

These functions shall convert 16-bit and 32-bit quantities between network byte order and host byte order.

On some implementations, these functions are defined as macros.

The `uint32_t` and `uint16_t` types are defined in `<inttypes.h>`.

**RETURN VALUE**

The `htonl()` and `htons()` functions shall return the argument value converted from host to network byte order.

The `ntohl()` and `ntohs()` functions shall return the argument value converted from network to host byte order.

**ERRORS**

No errors are defined.

**EXAMPLES**

None.

**APPLICATION USAGE**

These functions are most often used in conjunction with IPv4 addresses and ports as returned by `gethostent()` and `getservent()`.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*endhostent()*, *endservent()*

XBD `<arpa/inet.h>`, `<inttypes.h>`

**CHANGE HISTORY**

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

37752 **NAME**

37753 hypot, hypotf, hypotl — Euclidean distance function

37754 **SYNOPSIS**

```
37755 #include <math.h>
37756 double hypot(double x, double y);
37757 float hypotf(float x, float y);
37758 long double hypotl(long double x, long double y);
```

37759 **DESCRIPTION**

37760 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 37761 conflict between the requirements described here and the ISO C standard is unintentional. This  
 37762 volume of POSIX.1-2008 defers to the ISO C standard.

37763 These functions shall compute the value of the square root of  $x^2+y^2$  without undue overflow or  
 37764 underflow.

37765 An application wishing to check for error situations should set *errno* to zero and call  
 37766 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 37767 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 37768 zero, an error has occurred.

37769 **RETURN VALUE**

37770 Upon successful completion, these functions shall return the length of the hypotenuse of a right-  
 37771 angled triangle with sides of length *x* and *y*.

37772 If the correct value would cause overflow, a range error shall occur and *hypot()*, *hypotf()*, and  
 37773 *hypotl()* shall return the value of the macro HUGE\_VAL, HUGE\_VALF, and HUGE\_VALL,  
 37774 respectively.

37775 MX If *x* or *y* is  $\pm\text{Inf}$ ,  $+\text{Inf}$  shall be returned (even if one of *x* or *y* is NaN).

37776 If *x* or *y* is NaN, and the other is not  $\pm\text{Inf}$ , a NaN shall be returned.

37777 MXX If both arguments are subnormal and the correct result is subnormal, a range error may occur  
 37778 and the correct result shall be returned.

37779 **ERRORS**

37780 These functions shall fail if:

37781 Range Error The result overflows.

37782 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 37783 then *errno* shall be set to [ERANGE]. If the integer expression  
 37784 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the overflow  
 37785 floating-point exception shall be raised.

37786 These functions may fail if:

37787 MX Range Error The result underflows.

37788 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 37789 then *errno* shall be set to [ERANGE]. If the integer expression  
 37790 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 37791 floating-point exception shall be raised.

**EXAMPLES**

See the EXAMPLES section in *atan2()*.

**APPLICATION USAGE**

*hypot(x,y)*, *hypot(y,x)*, and *hypot(x, -y)* are equivalent.

*hypot(x, ±0)* is equivalent to *fabs(x)*.

Underflow only happens when both *x* and *y* are subnormal and the (inexact) result is also subnormal.

These functions take precautions against overflow during intermediate steps of the computation.

On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* & MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*atan2()*, *feclearexcept()*, *fetestexcept()*, *isnan()*, *sqrt()*

XBD Section 4.20 (on page 117), <math.h>

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

The DESCRIPTION is updated to indicate how an application should check for an error. This text was previously published in the APPLICATION USAGE section.

**Issue 6**

The *hypot()* function is no longer marked as an extension.

The *hypotf()* and *hypotl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/49 is applied, updating the EXAMPLES section.

**Issue 7**

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0273 [68] is applied.

**NAME**

iconv — codeset conversion function

**SYNOPSIS**

```
#include <iconv.h>

size_t iconv(iconv_t cd, char **restrict inbuf,
             size_t *restrict inbytesleft, char **restrict outbuf,
             size_t *restrict outbytesleft);
```

**DESCRIPTION**

The *iconv()* function shall convert the sequence of characters from one codeset, in the array specified by *inbuf*, into a sequence of corresponding characters in another codeset, in the array specified by *outbuf*. The codesets are those specified in the *iconv\_open()* call that returned the conversion descriptor, *cd*. The *inbuf* argument points to a variable that points to the first character in the input buffer and *inbytesleft* indicates the number of bytes to the end of the buffer to be converted. The *outbuf* argument points to a variable that points to the first available byte in the output buffer and *outbytesleft* indicates the number of the available bytes to the end of the buffer.

For state-dependent encodings, the conversion descriptor *cd* is placed into its initial shift state by a call for which *inbuf* is a null pointer, or for which *inbuf* points to a null pointer. When *iconv()* is called in this way, and if *outbuf* is not a null pointer or a pointer to a null pointer, and *outbytesleft* points to a positive value, *iconv()* shall place, into the output buffer, the byte sequence to change the output buffer to its initial shift state. If the output buffer is not large enough to hold the entire reset sequence, *iconv()* shall fail and set *errno* to [E2BIG]. Subsequent calls with *inbuf* as other than a null pointer or a pointer to a null pointer cause the conversion to take place from the current state of the conversion descriptor.

If a sequence of input bytes does not form a valid character in the specified codeset, conversion shall stop after the previous successfully converted character. If the input buffer ends with an incomplete character or shift sequence, conversion shall stop after the previous successfully converted bytes. If the output buffer is not large enough to hold the entire converted input, conversion shall stop just prior to the input bytes that would cause the output buffer to overflow. The variable pointed to by *inbuf* shall be updated to point to the byte following the last byte successfully used in the conversion. The value pointed to by *inbytesleft* shall be decremented to reflect the number of bytes still not converted in the input buffer. The variable pointed to by *outbuf* shall be updated to point to the byte following the last byte of converted output data. The value pointed to by *outbytesleft* shall be decremented to reflect the number of bytes still available in the output buffer. For state-dependent encodings, the conversion descriptor shall be updated to reflect the shift state in effect at the end of the last successfully converted byte sequence.

If *iconv()* encounters a character in the input buffer that is valid, but for which an identical character does not exist in the target codeset, *iconv()* shall perform an implementation-defined conversion on this character.

**RETURN VALUE**

The *iconv()* function shall update the variables pointed to by the arguments to reflect the extent of the conversion and return the number of non-identical conversions performed. If the entire string in the input buffer is converted, the value pointed to by *inbytesleft* shall be 0. If the input conversion is stopped due to any conditions mentioned above, the value pointed to by *inbytesleft* shall be non-zero and *errno* shall be set to indicate the condition. If an error occurs, *iconv()* shall return (**size\_t**)–1 and set *errno* to indicate the error.



**ERRORS**

The *iconv()* function shall fail if:

[EILSEQ] Input conversion stopped due to an input byte that does not belong to the input codeset.

[E2BIG] Input conversion stopped due to lack of space in the output buffer.

[EINVAL] Input conversion stopped due to an incomplete character or shift sequence at the end of the input buffer.

The *iconv()* function may fail if:

[EBADF] The *cd* argument is not a valid open conversion descriptor.

**EXAMPLES**

None.

**APPLICATION USAGE**

The *inbuf* argument indirectly points to the memory area which contains the conversion input data. The *outbuf* argument indirectly points to the memory area which is to contain the result of the conversion. The objects indirectly pointed to by *inbuf* and *outbuf* are not restricted to containing data that is directly representable in the ISO C standard language **char** data type. The type of *inbuf* and *outbuf*, **char \*\***, does not imply that the objects pointed to are interpreted as null-terminated C strings or arrays of characters. Any interpretation of a byte sequence that represents a character in a given character set encoding scheme is done internally within the codeset converters. For example, the area pointed to indirectly by *inbuf* and/or *outbuf* can contain all zero octets that are not interpreted as string terminators but as coded character data according to the respective codeset encoding scheme. The type of the data (**char**, **short**, **long**, and so on) read or stored in the objects is not specified, but may be inferred for both the input and output data by the converters determined by the *fromcode* and *tocode* arguments of *iconv\_open()*.

Regardless of the data type inferred by the converter, the size of the remaining space in both input and output objects (the *intbytesleft* and *outbytesleft* arguments) is always measured in bytes.

For implementations that support the conversion of state-dependent encodings, the conversion descriptor must be able to accurately reflect the shift-state in effect at the end of the last successful conversion. It is not required that the conversion descriptor itself be updated, which would require it to be a pointer type. Thus, implementations are free to implement the descriptor as a handle (other than a pointer type) by which the conversion information can be accessed and updated.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*iconv\_open()*, *iconv\_close()*, *mbsrtowcs()*

XBD <**iconv.h**>

**CHANGE HISTORY**

First released in Issue 4. Derived from the HP-UX Manual.

37914 **Issue 6**

37915 The SYNOPSIS has been corrected to align with the **<iconv.h>** reference page.

37916 The **restrict** keyword is added to the *iconv()* prototype for alignment with the  
37917 ISO/IEC 9899:1999 standard.

37918 **Issue 7**

37919 The *iconv()* function is moved from the XSI option to the Base.

37920 **NAME**

37921 iconv\_close — codeset conversion deallocation function

37922 **SYNOPSIS**

37923 #include &lt;iconv.h&gt;

37924 int iconv\_close(iconv\_t *cd*);37925 **DESCRIPTION**37926 The *iconv\_close()* function shall deallocate the conversion descriptor *cd* and all other associated  
37927 resources allocated by *iconv\_open()*.37928 If a file descriptor is used to implement the type **iconv\_t**, that file descriptor shall be closed.37929 **RETURN VALUE**37930 Upon successful completion, 0 shall be returned; otherwise, -1 shall be returned and *errno* set to  
37931 indicate the error.37932 **ERRORS**37933 The *iconv\_close()* function may fail if:

37934 [EBADF] The conversion descriptor is invalid.

37935 **EXAMPLES**

37936 None.

37937 **APPLICATION USAGE**

37938 None.

37939 **RATIONALE**

37940 None.

37941 **FUTURE DIRECTIONS**

37942 None.

37943 **SEE ALSO**37944 *iconv()*, *iconv\_open()*

37945 XBD &lt;iconv.h&gt;

37946 **CHANGE HISTORY**

37947 First released in Issue 4. Derived from the HP-UX Manual.

37948 **Issue 7**37949 The *iconv\_close()* function is moved from the XSI option to the Base.

**NAME**

iconv\_open — codeset conversion allocation function

**SYNOPSIS**

```
#include <iconv.h>
```

```
iconv_t iconv_open(const char *tocode, const char *fromcode);
```

**DESCRIPTION**

The *iconv\_open()* function shall return a conversion descriptor that describes a conversion from the codeset specified by the string pointed to by the *fromcode* argument to the codeset specified by the string pointed to by the *tocode* argument. For state-dependent encodings, the conversion descriptor shall be in a codeset-dependent initial shift state, ready for immediate use with *iconv()*.

Settings of *fromcode* and *tocode* and their permitted combinations are implementation-defined.

A conversion descriptor shall remain valid until it is closed by *iconv\_close()* or an implicit close.

If a file descriptor is used to implement conversion descriptors, the FD\_CLOEXEC flag shall be set; see <fcntl.h>.

**RETURN VALUE**

Upon successful completion, *iconv\_open()* shall return a conversion descriptor for use on subsequent calls to *iconv()*. Otherwise, *iconv\_open()* shall return (**iconv\_t**)−1 and set *errno* to indicate the error.

**ERRORS**

The *iconv\_open()* function may fail if:

[EMFILE] All file descriptors available to the process are currently open.

[ENFILE] Too many files are currently open in the system.

[ENOMEM] Insufficient storage space is available.

[EINVAL] The conversion specified by *fromcode* and *tocode* is not supported by the implementation.

**EXAMPLES**

None.

**APPLICATION USAGE**

Some implementations of *iconv\_open()* use *malloc()* to allocate space for internal buffer areas. The *iconv\_open()* function may fail if there is insufficient storage space to accommodate these buffers.

Conforming applications must assume that conversion descriptors are not valid after a call to one of the *exec* functions.

Application developers should consult the system documentation to determine the supported codesets and their naming schemes.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

37990 **SEE ALSO**37991 *iconv()*, *iconv\_close()*

37992 XBD &lt;fcntl.h&gt;, &lt;iconv.h&gt;

37993 **CHANGE HISTORY**

37994 First released in Issue 4. Derived from the HP-UX Manual.

37995 **Issue 7**

37996 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

37997 The *iconv\_open()* function is moved from the XSI option to the Base.

**37998 NAME**

37999       if\_freenameindex — free memory allocated by if\_nameindex

**38000 SYNOPSIS**

38001       #include <net/if.h>

38002       void if\_freenameindex(struct if\_nameindex \*ptr);

**38003 DESCRIPTION**

38004       The *if\_freenameindex()* function shall free the memory allocated by *if\_nameindex()*. The *ptr*  
 38005       argument shall be a pointer that was returned by *if\_nameindex()*. After *if\_freenameindex()* has  
 38006       been called, the application shall not use the array of which *ptr* is the address.

**38007 RETURN VALUE**

38008       None.

**38009 ERRORS**

38010       No errors are defined.

**38011 EXAMPLES**

38012       None.

**38013 APPLICATION USAGE**

38014       None.

**38015 RATIONALE**

38016       None.

**38017 FUTURE DIRECTIONS**

38018       None.

**38019 SEE ALSO**

38020       *getsockopt()*, *if\_indextoname()*, *if\_nameindex()*, *if\_nametoindex()*, *setsockopt()*

38021       XBD <net/if.h>

**38022 CHANGE HISTORY**

38023       First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

38024 **NAME**

38025 if\_indextoname — map a network interface index to its corresponding name

38026 **SYNOPSIS**

38027 #include &lt;net/if.h&gt;

38028 char \*if\_indextoname(unsigned ifindex, char \*ifname);

38029 **DESCRIPTION**38030 The *if\_indextoname()* function shall map an interface index to its corresponding name.38031 When this function is called, *ifname* shall point to a buffer of at least {IF\_NAMESIZE} bytes. The  
38032 function shall place in this buffer the name of the interface with index *ifindex*.38033 **RETURN VALUE**38034 If *ifindex* is an interface index, then the function shall return the value supplied in *ifname*, which  
38035 points to a buffer now containing the interface name. Otherwise, the function shall return a null  
38036 pointer and set *errno* to indicate the error.38037 **ERRORS**38038 The *if\_indextoname()* function shall fail if:

38039 [ENXIO] The interface does not exist.

38040 **EXAMPLES**

38041 None.

38042 **APPLICATION USAGE**

38043 None.

38044 **RATIONALE**

38045 None.

38046 **FUTURE DIRECTIONS**

38047 None.

38048 **SEE ALSO**38049 *getsockopt()*, *if\_freenameindex()*, *if\_nameindex()*, *if\_nametoindex()*, *setsockopt()*

38050 XBD &lt;net/if.h&gt;

38051 **CHANGE HISTORY**

38052 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

38053 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/28 is applied, changing {IFNAMSIZ} to  
38054 {IF\_NAMESIZ} in the DESCRIPTION.

38055 **NAME**

38056 if\_nameindex — return all network interface names and indexes

38057 **SYNOPSIS**

38058 #include <net/if.h>

38059 struct if\_nameindex \*if\_nameindex(void);

38060 **DESCRIPTION**

38061 The *if\_nameindex()* function shall return an array of *if\_nameindex* structures, one structure per  
38062 interface. The end of the array is indicated by a structure with an *if\_index* field of zero and an  
38063 *if\_name* field of NULL.

38064 Applications should call *if\_freenameindex()* to release the memory that may be dynamically  
38065 allocated by this function, after they have finished using it.

38066 **RETURN VALUE**

38067 An array of structures identifying local interfaces. A null pointer is returned upon an error, with  
38068 *errno* set to indicate the error.

38069 **ERRORS**

38070 The *if\_nameindex()* function may fail if:

38071 [ENOBUFS] Insufficient resources are available to complete the function.

38072 **EXAMPLES**

38073 None.

38074 **APPLICATION USAGE**

38075 None.

38076 **RATIONALE**

38077 None.

38078 **FUTURE DIRECTIONS**

38079 None.

38080 **SEE ALSO**

38081 *getsockopt()*, *if\_freenameindex()*, *if\_indextoname()*, *if\_nametoindex()*, *setsockopt()*

38082 XBD <net/if.h>

38083 **CHANGE HISTORY**

38084 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.



**38085 NAME**

38086 if\_nametoindex — map a network interface name to its corresponding index

**38087 SYNOPSIS**

38088 #include <net/if.h>

38089 unsigned if\_nametoindex(const char \*ifname);

**38090 DESCRIPTION**

38091 The *if\_nametoindex()* function shall return the interface index corresponding to name *ifname*.

**38092 RETURN VALUE**

38093 The corresponding index if *ifname* is the name of an interface; otherwise, zero.

**38094 ERRORS**

38095 No errors are defined.

**38096 EXAMPLES**

38097 None.

**38098 APPLICATION USAGE**

38099 None.

**38100 RATIONALE**

38101 None.

**38102 FUTURE DIRECTIONS**

38103 None.

**38104 SEE ALSO**

38105 *getsockopt()*, *if\_freenameindex()*, *if\_indextoname()*, *if\_nameindex()*, *setsockopt()*

38106 XBD <net/if.h>

**38107 CHANGE HISTORY**

38108 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

38109 **NAME**

38110       ilogb, ilogbf, ilogbl — return an unbiased exponent

38111 **SYNOPSIS**

```
38112       #include <math.h>
38113       int ilogb(double x);
38114       int ilogbf(float x);
38115       int ilogbl(long double x);
```

38116 **DESCRIPTION**

38117 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 38118       conflict between the requirements described here and the ISO C standard is unintentional. This  
 38119       volume of POSIX.1-2008 defers to the ISO C standard.

38120       These functions shall return the exponent part of their argument  $x$ . Formally, the return value is  
 38121       the integral part of  $\log_r |x|$  as a signed integral value, for non-zero  $x$ , where  $r$  is the radix of the  
 38122       machine's floating-point arithmetic, which is the value of FLT\_RADIX defined in <float.h>.

38123       An application wishing to check for error situations should set *errno* to zero and call  
 38124       *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 38125       *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 38126       zero, an error has occurred.

38127 **RETURN VALUE**

38128       Upon successful completion, these functions shall return the exponent part of  $x$  as a signed  
 38129       integer value. They are equivalent to calling the corresponding *logb*() function and casting the  
 38130       returned value to type **int**.

38131 XSI       If  $x$  is 0, the value FP\_ILOGB0 shall be returned. On XSI-conformant systems, a domain error  
 38132       shall occur;

38133 CX       otherwise, a domain error may occur.

38134 XSI       If  $x$  is  $\pm\text{Inf}$ , the value {INT\_MAX} shall be returned. On XSI-conformant systems, a domain  
 38135       error shall occur;

38136 CX       otherwise, a domain error may occur.

38137 XSI       If  $x$  is a NaN, the value FP\_ILOGBNAN shall be returned. On XSI-conformant systems, a  
 38138       domain error shall occur;

38139 CX       otherwise, a domain error may occur.

38140 MX       If the correct value is greater than {INT\_MAX}, a domain error shall occur and an unspecified  
 38141 XSI       value shall be returned. On XSI-conformant systems, a domain error shall occur and  
 38142       {INT\_MAX} shall be returned.

38143 MX       If the correct value is less than {INT\_MIN}, a domain error shall occur and an unspecified value  
 38144 XSI       shall be returned. On XSI-conformant systems, a domain error shall occur and {INT\_MIN} shall  
 38145       be returned.

38146 **ERRORS**

38147       These functions shall fail if:

38148 XSI|MX   **Domain Error**   The correct value is not representable as an integer.

38149 XSI       The  $x$  argument is zero, NaN, or  $\pm\text{Inf}$ .

38150 XSI|MX   If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 38151       then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 38152       & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 38153       shall be raised.

38154 These functions may fail if:

38155 Domain Error The  $x$  argument is zero, NaN, or  $\pm\text{Inf}$ .

38156 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 38157 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 38158 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 38159 shall be raised.

#### 38160 EXAMPLES

38161 None.

#### 38162 APPLICATION USAGE

38163 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 38164 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

#### 38165 RATIONALE

38166 The errors come from taking the expected floating-point value and converting it to **int**, which is  
 38167 an invalid operation in IEEE Std 754-1985 (since overflow, infinity, and NaN are not  
 38168 representable in a type **int**), so should be a domain error.

38169 There are no known implementations that overflow. For overflow to happen, {INT\_MAX} must  
 38170 be less than  $\text{LDBL\_MAX\_EXP} \cdot \log_2(\text{FLT\_RADIX})$  or {INT\_MIN} must be greater than  
 38171  $\text{LDBL\_MIN\_EXP} \cdot \log_2(\text{FLT\_RADIX})$  if subnormals are not supported, or {INT\_MIN} must be  
 38172 greater than  $(\text{LDBL\_MIN\_EXP} - \text{LDBL\_MANT\_DIG}) \cdot \log_2(\text{FLT\_RADIX})$  if subnormals are  
 38173 supported.

#### 38174 FUTURE DIRECTIONS

38175 None.

#### 38176 SEE ALSO

38177 *feclearexcept()*, *fetestexcept()*, *logb()*, *scalbln()*

38178 XBD Section 4.20 (on page 117), [<float.h>](#), [<math.h>](#)

#### 38179 CHANGE HISTORY

38180 First released in Issue 4, Version 2.

#### 38181 Issue 5

38182 Moved from X/OPEN UNIX extension to BASE.

#### 38183 Issue 6

38184 The *ilogb()* function is no longer marked as an extension.

38185 The *ilogbf()* and *ilogbl()* functions are added for alignment with the ISO/IEC 9899:1999  
 38186 standard.

38187 The RETURN VALUE section is revised for alignment with the ISO/IEC 9899:1999 standard.

38188 Functionality relating to the XSI option is marked.

#### 38189 Issue 7

38190 ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #48 (SD5-XSH-ERN-71), #49, and #79  
 38191 (SD5-XSH-ERN-72) are applied.

38192 **NAME**

38193 imaxabs — return absolute value

38194 **SYNOPSIS**

38195 #include &lt;inttypes.h&gt;

38196 intmax\_t imaxabs(intmax\_t j);

38197 **DESCRIPTION**

38198 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
38199 conflict between the requirements described here and the ISO C standard is unintentional. This  
38200 volume of POSIX.1-2008 defers to the ISO C standard.

38201 The *imaxabs()* function shall compute the absolute value of an integer *j*. If the result cannot be  
38202 represented, the behavior is undefined.

38203 **RETURN VALUE**38204 The *imaxabs()* function shall return the absolute value.38205 **ERRORS**

38206 No errors are defined.

38207 **EXAMPLES**

38208 None.

38209 **APPLICATION USAGE**

38210 The absolute value of the most negative number cannot be represented in two's complement.

38211 **RATIONALE**

38212 None.

38213 **FUTURE DIRECTIONS**

38214 None.

38215 **SEE ALSO**38216 *imaxdiv()*

38217 XBD &lt;inttypes.h&gt;

38218 **CHANGE HISTORY**

38219 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

38220 **NAME**

38221 imaxdiv — return quotient and remainder

38222 **SYNOPSIS**

38223 #include &lt;inttypes.h&gt;

38224 imaxdiv\_t imaxdiv(intmax\_t numer, intmax\_t denom);

38225 **DESCRIPTION**

38226 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
38227 conflict between the requirements described here and the ISO C standard is unintentional. This  
38228 volume of POSIX.1-2008 defers to the ISO C standard.

38229 The *imaxdiv()* function shall compute *numer* / *denom* and *numer* % *denom* in a single operation.38230 **RETURN VALUE**

38231 The *imaxdiv()* function shall return a structure of type **imaxdiv\_t**, comprising both the quotient  
38232 and the remainder. The structure shall contain (in either order) the members *quot* (the quotient)  
38233 and *rem* (the remainder), each of which has type **intmax\_t**.

38234 If either part of the result cannot be represented, the behavior is undefined.

38235 **ERRORS**

38236 No errors are defined.

38237 **EXAMPLES**

38238 None.

38239 **APPLICATION USAGE**

38240 None.

38241 **RATIONALE**

38242 None.

38243 **FUTURE DIRECTIONS**

38244 None.

38245 **SEE ALSO**38246 *imaxabs()*

38247 XBD &lt;inttypes.h&gt;

38248 **CHANGE HISTORY**

38249 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

**NAME**

inet\_addr, inet\_ntoa — IPv4 address manipulation

**SYNOPSIS**

```
#include <arpa/inet.h>

in_addr_t inet_addr(const char *cp);
char *inet_ntoa(struct in_addr in);
```

**DESCRIPTION**

The *inet\_addr()* function shall convert the string pointed to by *cp*, in the standard IPv4 dotted decimal notation, to an integer value suitable for use as an Internet address.

The *inet\_ntoa()* function shall convert the Internet host address specified by *in* to a string in the Internet standard dot notation.

The *inet\_ntoa()* function need not be thread-safe.

All Internet addresses shall be returned in network order (bytes ordered from left to right).

Values specified using IPv4 dotted decimal notation take one of the following forms:

*a.b.c.d* When four parts are specified, each shall be interpreted as a byte of data and assigned, from left to right, to the four bytes of an Internet address.

*a.b.c* When a three-part address is specified, the last part shall be interpreted as a 16-bit quantity and placed in the rightmost two bytes of the network address. This makes the three-part address format convenient for specifying Class B network addresses as "128.net.host".

*a.b* When a two-part address is supplied, the last part shall be interpreted as a 24-bit quantity and placed in the rightmost three bytes of the network address. This makes the two-part address format convenient for specifying Class A network addresses as "net.host".

*a* When only one part is given, the value shall be stored directly in the network address without any byte rearrangement.

All numbers supplied as parts in IPv4 dotted decimal notation may be decimal, octal, or hexadecimal, as specified in the ISO C standard (that is, a leading 0x or 0X implies hexadecimal; otherwise, a leading '0' implies octal; otherwise, the number is interpreted as decimal).

**RETURN VALUE**

Upon successful completion, *inet\_addr()* shall return the Internet address. Otherwise, it shall return (*in\_addr\_t*)(-1).

The *inet\_ntoa()* function shall return a pointer to the network address in Internet standard dot notation.

**ERRORS**

No errors are defined.

**38286 EXAMPLES**

38287       None.

**38288 APPLICATION USAGE**

38289       The return value of *inet\_ntoa()* may point to static data that may be overwritten by subsequent  
38290       calls to *inet\_ntoa()*.

**38291 RATIONALE**

38292       None.

**38293 FUTURE DIRECTIONS**

38294       None.

**38295 SEE ALSO**

38296       *endhostent()*, *endnetent()*

38297       XBD <arpa/inet.h>

**38298 CHANGE HISTORY**

38299       First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

**38300 Issue 7**

38301       Austin Group Interpretation 1003.1-2001 #156 is applied.

## 38302 NAME

38303 inet\_ntop, inet\_pton — convert IPv4 and IPv6 addresses between binary and text form

## 38304 SYNOPSIS

```
38305 #include <arpa/inet.h>
38306
38306 const char *inet_ntop(int af, const void *restrict src,
38307                       char *restrict dst, socklen_t size);
38308
38308 int inet_pton(int af, const char *restrict src, void *restrict dst);
```

## 38309 DESCRIPTION

38310 The *inet\_ntop()* function shall convert a numeric address into a text string suitable for  
 38311 IP6 presentation. The *af* argument shall specify the family of the address. This can be AF\_INET or  
 38312 AF\_INET6. The *src* argument points to a buffer holding an IPv4 address if the *af* argument is  
 38313 IP6 AF\_INET, or an IPv6 address if the *af* argument is AF\_INET6; the address must be in network  
 38314 byte order. The *dst* argument points to a buffer where the function stores the resulting text string;  
 38315 it shall not be NULL. The *size* argument specifies the size of this buffer, which shall be large  
 38316 IP6 enough to hold the text string (INET\_ADDRSTRLEN characters for IPv4,  
 38317 INET6\_ADDRSTRLEN characters for IPv6).

38318 The *inet\_pton()* function shall convert an address in its standard text presentation form into its  
 38319 IP6 numeric binary form. The *af* argument shall specify the family of the address. The AF\_INET and  
 38320 AF\_INET6 address families shall be supported. The *src* argument points to the string being  
 38321 passed in. The *dst* argument points to a buffer into which the function stores the numeric  
 38322 IP6 address; this shall be large enough to hold the numeric address (32 bits for AF\_INET, 128 bits  
 38323 for AF\_INET6).

38324 If the *af* argument of *inet\_pton()* is AF\_INET, the *src* string shall be in the standard IPv4 dotted-  
 38325 decimal form:

38326 ddd.ddd.ddd.ddd

38327 where "ddd" is a one to three digit decimal number between 0 and 255 (see *inet\_addr()*). The  
 38328 *inet\_pton()* function does not accept other formats (such as the octal numbers, hexadecimal  
 38329 numbers, and fewer than four numbers that *inet\_addr()* accepts).

38330 IP6 If the *af* argument of *inet\_pton()* is AF\_INET6, the *src* string shall be in one of the following  
 38331 standard IPv6 text forms:

- 38332 1. The preferred form is "x:x:x:x:x:x:x:x", where the 'x's are the hexadecimal values  
 38333 of the eight 16-bit pieces of the address. Leading zeros in individual fields can be  
 38334 omitted, but there shall be one to four hexadecimal digits in every field.
- 38335 2. A string of contiguous zero fields in the preferred form can be shown as "::". The "::"  
 38336 can only appear once in an address. Unspecified addresses ("0:0:0:0:0:0:0:0") may  
 38337 be represented simply as "::".
- 38338 3. A third form that is sometimes more convenient when dealing with a mixed environment  
 38339 of IPv4 and IPv6 nodes is "x:x:x:x:x:x.d.d.d.d", where the 'x's are the  
 38340 hexadecimal values of the six high-order 16-bit pieces of the address, and the 'd's are the  
 38341 decimal values of the four low-order 8-bit pieces of the address (standard IPv4  
 38342 representation).

38343 **Note:** A more extensive description of the standard representations of IPv6 addresses can be found in  
 38344 RFC 2373.



**RETURN VALUE**

The *inet\_ntop()* function shall return a pointer to the buffer containing the text string if the conversion succeeds, and NULL otherwise, and set *errno* to indicate the error.

The *inet\_pton()* function shall return 1 if the conversion succeeds, with the address pointed to by *dst* in network byte order. It shall return 0 if the input is not a valid IPv4 dotted-decimal string or a valid IPv6 address string, or -1 with *errno* set to [EAFNOSUPPORT] if the *af* argument is unknown.

**ERRORS**

The *inet\_ntop()* and *inet\_pton()* functions shall fail if:

[EAFNOSUPPORT]

The *af* argument is invalid.

[ENOSPC]

The size of the *inet\_ntop()* result buffer is inadequate.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

XBD <arpa/inet.h>

**CHANGE HISTORY**

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

IPv6 extensions are marked.

The **restrict** keyword is added to the *inet\_ntop()* and *inet\_pton()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/29 is applied, adding “the address must be in network byte order” to the end of the fourth sentence of the first paragraph in the DESCRIPTION.

**NAME**

initstate, random, setstate, srandom — pseudo-random number functions

**SYNOPSIS**

```
XSI #include <stdlib.h>

char *initstate(unsigned seed, char *state, size_t size);
long random(void);
char *setstate(char *state);
void srandom(unsigned seed);
```

**DESCRIPTION**

The *random()* function shall use a non-linear additive feedback random-number generator employing a default state array size of 31 **long** integers to return successive pseudo-random numbers in the range from 0 to  $2^{31}-1$ . The period of this random-number generator is approximately  $16 \times (2^{31}-1)$ . The size of the state array determines the period of the random-number generator. Increasing the state array size shall increase the period.

With 256 bytes of state information, the period of the random-number generator shall be greater than  $2^{69}$ .

Like *rand()*, *random()* shall produce by default a sequence of numbers that can be duplicated by calling *srandom()* with 1 as the seed.

The *srandom()* function shall initialize the current state array using the value of *seed*.

The *initstate()* and *setstate()* functions handle restarting and changing random-number generators. The *initstate()* function allows a state array, pointed to by the *state* argument, to be initialized for future use. The *size* argument, which specifies the size in bytes of the state array, shall be used by *initstate()* to decide what type of random-number generator to use; the larger the state array, the more random the numbers. Values for the amount of state information are 8, 32, 64, 128, and 256 bytes. Other values greater than 8 bytes are rounded down to the nearest one of these values. If *initstate()* is called with  $8 \leq \text{size} < 32$ , then *random()* shall use a simple linear congruential random number generator. The *seed* argument specifies a starting point for the random-number sequence and provides for restarting at the same point. The *initstate()* function shall return a pointer to the previous state information array.

If *initstate()* has not been called, then *random()* shall behave as though *initstate()* had been called with *seed*=1 and *size*=128.

Once a state has been initialized, *setstate()* allows switching between state arrays. The array defined by the *state* argument shall be used for further random-number generation until *initstate()* is called or *setstate()* is called again. The *setstate()* function shall return a pointer to the previous state array.

**RETURN VALUE**

If *initstate()* is called with *size* less than 8, it shall return NULL.

The *random()* function shall return the generated pseudo-random number.

The *srandom()* function shall not return a value.

Upon successful completion, *initstate()* and *setstate()* shall return a pointer to the previous state array; otherwise, a null pointer shall be returned.

**ERRORS**

No errors are defined.

**EXAMPLES**

None.

**APPLICATION USAGE**

After initialization, a state array can be restarted at a different point in one of two ways:

1. The *initstate()* function can be used, with the desired seed, state array, and size of the array.
2. The *setstate()* function, with the desired state, can be used, followed by *srandom()* with the desired seed. The advantage of using both of these functions is that the size of the state array does not have to be saved once it is initialized.

Although some implementations of *random()* have written messages to standard error, such implementations do not conform to POSIX.1-2008.

Issue 5 restored the historical behavior of this function.

Threaded applications should use *erand48()*, *nrand48()*, or *jrand48()* instead of *random()* when an independent random number sequence in multiple threads is required.

These functions should be avoided whenever non-trivial requirements (including safety) have to be fulfilled.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*drand48()*, *rand()*

XBD <stdlib.h>

**CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

In the DESCRIPTION, the phrase “values smaller than 8” is replaced with “values greater than or equal to 8, or less than 32”, “size<8” is replaced with “8≤size <32”, and a new first paragraph is added to the RETURN VALUE section. A note is added to the APPLICATION USAGE indicating that these changes restore the historical behavior of the function.

**Issue 6**

In the DESCRIPTION, duplicate text “For values greater than or equal to 8...” is removed.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/30 is applied, removing *rand\_r()* from the list of suggested functions in the APPLICATION USAGE section.

**Issue 7**

The type of the first argument to *setstate()* is changed from **const char \*** to **char \***.

**NAME**

insque, remque — insert or remove an element in a queue

**SYNOPSIS**

```
XSI    #include <search.h>

void insque(void *element, void *pred);
void remque(void *element);
```

**DESCRIPTION**

The *insque()* and *remque()* functions shall manipulate queues built from doubly-linked lists. The queue can be either circular or linear. An application using *insque()* or *remque()* shall ensure it defines a structure in which the first two members of the structure are pointers to the same type of structure, and any further members are application-specific. The first member of the structure is a forward pointer to the next entry in the queue. The second member is a backward pointer to the previous entry in the queue. If the queue is linear, the queue is terminated with null pointers. The names of the structure and of the pointer members are not subject to any special restriction.

The *insque()* function shall insert the element pointed to by *element* into a queue immediately after the element pointed to by *pred*.

The *remque()* function shall remove the element pointed to by *element* from a queue.

If the queue is to be used as a linear list, invoking *insque(&element, NULL)*, where *element* is the initial element of the queue, shall initialize the forward and backward pointers of *element* to null pointers.

If the queue is to be used as a circular list, the application shall ensure it initializes the forward pointer and the backward pointer of the initial element of the queue to the element's own address.

**RETURN VALUE**

The *insque()* and *remque()* functions do not return a value.

**ERRORS**

No errors are defined.

**EXAMPLES****Creating a Linear Linked List**

The following example creates a linear linked list.

```
#include <search.h>
...
struct myque element1;
struct myque element2;

char *data1 = "DATA1";
char *data2 = "DATA2";
...
element1.data = data1;
element2.data = data2;

insque (&element1, NULL);
insque (&element2, &element1);
```

### Creating a Circular Linked List

The following example creates a circular linked list.

```
#include <search.h>
...
struct myque element1;
struct myque element2;

char *data1 = "DATA1";
char *data2 = "DATA2";
...
element1.data = data1;
element2.data = data2;

element1.fwd = &element1;
element1.bck = &element1;

insque (&element2, &element1);
```

### Removing an Element

The following example removes the element pointed to by *element1*.

```
#include <search.h>
...
struct myque element1;
...
remque (&element1);
```

## APPLICATION USAGE

The historical implementations of these functions described the arguments as being of type **struct qelem \*** rather than as being of type **void \*** as defined here. In those implementations, **struct qelem** was commonly defined in **<search.h>** as:

```
struct qelem {
    struct qelem *q_forw;
    struct qelem *q_back;
};
```

Applications using these functions, however, were never able to use this structure directly since it provided no room for the actual data contained in the elements. Most applications defined structures that contained the two pointers as the initial elements and also provided space for, or pointers to, the object's data. Applications that used these functions to update more than one type of table also had the problem of specifying two or more different structures with the same name, if they literally used **struct qelem** as specified.

As described here, the implementations were actually expecting a structure type where the first two members were forward and backward pointers to structures. With C compilers that didn't provide function prototypes, applications used structures as specified in the DESCRIPTION above and the compiler did what the application expected.

If this method had been carried forward with an ISO C standard compiler and the historical function prototype, most applications would have to be modified to cast pointers to the structures actually used to be pointers to **struct qelem** to avoid compilation warnings. By specifying **void \*** as the argument type, applications do not need to change (unless they specifically referenced **struct qelem** and depended on it being defined in **<search.h>**).

38541 **RATIONALE**

38542 None.

38543 **FUTURE DIRECTIONS**

38544 None.

38545 **SEE ALSO**38546 XBD [<search.h>](#)38547 **CHANGE HISTORY**

38548 First released in Issue 4, Version 2.

38549 **Issue 5**

38550 Moved from X/OPEN UNIX extension to BASE.

38551 **Issue 6**

38552 The normative text is updated to avoid use of the term “must” for application requirements.

38553 **NAME**38554        **ioctl** — control a STREAMS device (**STREAMS**)38555 **SYNOPSIS**

```
38556 OB XSR  #include <stropts.h>
38557         int ioctl(int fildes, int request, ... /* arg */);
```

38558 **DESCRIPTION**

38559        The *ioctl()* function shall perform a variety of control functions on STREAMS devices. For non-  
 38560        STREAMS devices, the functions performed by this call are unspecified. The *request* argument  
 38561        and an optional third argument (with varying type) shall be passed to and interpreted by the  
 38562        appropriate part of the STREAM associated with *fildes*.

38563        The *fildes* argument is an open file descriptor that refers to a device.

38564        The *request* argument selects the control function to be performed and shall depend on the  
 38565        STREAMS device being addressed.

38566        The *arg* argument represents additional information that is needed by this specific STREAMS  
 38567        device to perform the requested function. The type of *arg* depends upon the particular control  
 38568        request, but it shall be either an integer or a pointer to a device-specific data structure.

38569        The *ioctl()* commands applicable to STREAMS, their arguments, and error conditions that apply  
 38570        to each individual command are described below.

38571        The following *ioctl()* commands, with error values indicated, are applicable to all STREAMS  
 38572        files:

38573        **I\_PUSH**        Pushes the module whose name is pointed to by *arg* onto the top of the current  
 38574        STREAM, just below the STREAM head. It then calls the *open()* function of the  
 38575        newly-pushed module.

38576        The *ioctl()* function with the **I\_PUSH** command shall fail if:

38577        [EINVAL]        Invalid module name.

38578        [ENXIO]        Open function of new module failed.

38579        [ENXIO]        Hangup received on *fildes*.

38580        **I\_POP**        Removes the module just below the STREAM head of the STREAM pointed to  
 38581        by *fildes*. The *arg* argument should be 0 in an **I\_POP** request.

38582        The *ioctl()* function with the **I\_POP** command shall fail if:

38583        [EINVAL]        No module present in the STREAM.

38584        [ENXIO]        Hangup received on *fildes*.

38585        **I\_LOOK**        Retrieves the name of the module just below the STREAM head of the  
 38586        STREAM pointed to by *fildes*, and places it in a character string pointed to by  
 38587        *arg*. The buffer pointed to by *arg* should be at least FMNAMESZ+1 bytes long,  
 38588        where FMNAMESZ is defined in **<stropts.h>**.

38589        The *ioctl()* function with the **I\_LOOK** command shall fail if:

38590        [EINVAL]        No module present in the STREAM.

38591        **I\_FLUSH**       Flushes read and/or write queues, depending on the value of *arg*. Valid *arg*  
 38592        values are:

|       |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------|----------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 38593 | FLUSHR                                                                     | Flush all read queues.                                                                                                                                                                                                                                                                                                                                                                                                                |
| 38594 | FLUSHW                                                                     | Flush all write queues.                                                                                                                                                                                                                                                                                                                                                                                                               |
| 38595 | FLUSHRW                                                                    | Flush all read and all write queues.                                                                                                                                                                                                                                                                                                                                                                                                  |
| 38596 | The <i>ioctl()</i> function with the <i>I_FLUSH</i> command shall fail if: |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38597 | [EINVAL]                                                                   | Invalid <i>arg</i> value.                                                                                                                                                                                                                                                                                                                                                                                                             |
| 38598 | [EAGAIN] or [ENOSR]                                                        | Unable to allocate buffers for flush message.                                                                                                                                                                                                                                                                                                                                                                                         |
| 38599 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38600 | [ENXIO]                                                                    | Hangup received on <i>fildes</i> .                                                                                                                                                                                                                                                                                                                                                                                                    |
| 38601 | I_FLUSHBAND                                                                | Flushes a particular band of messages. The <i>arg</i> argument points to a <b>bandinfo</b> structure. The <i>bi_flag</i> member may be one of FLUSHR, FLUSHW, or FLUSHRW as described above. The <i>bi_pri</i> member determines the priority band to be flushed.                                                                                                                                                                     |
| 38602 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38603 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38604 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38605 | I_SETSIG                                                                   | Requests that the STREAMS implementation send the SIGPOLL signal to the calling process when a particular event has occurred on the STREAM associated with <i>fildes</i> . I_SETSIG supports an asynchronous processing capability in STREAMS. The value of <i>arg</i> is a bitmask that specifies the events for which the process should be signaled. It is the bitwise-inclusive OR of any combination of the following constants: |
| 38606 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38607 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38608 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38609 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38610 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38611 | S_RDNORM                                                                   | A normal (priority band set to 0) message has arrived at the head of a STREAM head read queue. A signal shall be generated even if the message is of zero length.                                                                                                                                                                                                                                                                     |
| 38612 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38613 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38614 | S_RDBAND                                                                   | A message with a non-zero priority band has arrived at the head of a STREAM head read queue. A signal shall be generated even if the message is of zero length.                                                                                                                                                                                                                                                                       |
| 38615 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38616 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38617 | S_INPUT                                                                    | A message, other than a high-priority message, has arrived at the head of a STREAM head read queue. A signal shall be generated even if the message is of zero length.                                                                                                                                                                                                                                                                |
| 38618 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38619 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38620 | S_HIPRI                                                                    | A high-priority message is present on a STREAM head read queue. A signal shall be generated even if the message is of zero length.                                                                                                                                                                                                                                                                                                    |
| 38621 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38622 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38623 | S_OUTPUT                                                                   | The write queue for normal data (priority band 0) just below the STREAM head is no longer full. This notifies the process that there is room on the queue for sending (or writing) normal data downstream.                                                                                                                                                                                                                            |
| 38624 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38625 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38626 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38627 | S_WRNORM                                                                   | Equivalent to S_OUTPUT.                                                                                                                                                                                                                                                                                                                                                                                                               |
| 38628 | S_WRBAND                                                                   | The write queue for a non-zero priority band just below the STREAM head is no longer full. This notifies the process that there is room on the queue for sending (or writing) priority data downstream.                                                                                                                                                                                                                               |
| 38629 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38630 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38631 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38632 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38632 | S_MSG                                                                      | A STREAMS signal message that contains the SIGPOLL signal has reached the front of the STREAM head read queue.                                                                                                                                                                                                                                                                                                                        |
| 38633 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 38634 |                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                       |



|       |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|-------|-----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 38635 | S_ERROR   | Notification of an error condition has reached the STREAM head.                                                                                                                                                                                                                                                                                                                                                                                            |
| 38636 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38637 | S_HANGUP  | Notification of a hangup has reached the STREAM head.                                                                                                                                                                                                                                                                                                                                                                                                      |
| 38638 | S_BANDURG | When used in conjunction with S_RDBAND, SIGURG is generated instead of SIGPOLL when a priority message reaches the front of the STREAM head read queue.                                                                                                                                                                                                                                                                                                    |
| 38639 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38640 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38641 |           | If <i>arg</i> is 0, the calling process shall be unregistered and shall not receive further SIGPOLL signals for the stream associated with <i>fildev</i> .                                                                                                                                                                                                                                                                                                 |
| 38642 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38643 |           | Processes that wish to receive SIGPOLL signals shall ensure that they explicitly register to receive them using I_SETSIG. If several processes register to receive this signal for the same event on the same STREAM, each process shall be signaled when the event occurs.                                                                                                                                                                                |
| 38644 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38645 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38646 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38647 |           | The <i>ioctl()</i> function with the I_SETSIG command shall fail if:                                                                                                                                                                                                                                                                                                                                                                                       |
| 38648 | [EINVAL]  | The value of <i>arg</i> is invalid.                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 38649 | [EINVAL]  | The value of <i>arg</i> is 0 and the calling process is not registered to receive the SIGPOLL signal.                                                                                                                                                                                                                                                                                                                                                      |
| 38650 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38651 | [EAGAIN]  | There were insufficient resources to store the signal request.                                                                                                                                                                                                                                                                                                                                                                                             |
| 38652 | I_GETSIG  | Returns the events for which the calling process is currently registered to be sent a SIGPOLL signal. The events are returned as a bitmask in an <b>int</b> pointed to by <i>arg</i> , where the events are those specified in the description of I_SETSIG above.                                                                                                                                                                                          |
| 38653 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38654 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38655 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38656 |           | The <i>ioctl()</i> function with the I_GETSIG command shall fail if:                                                                                                                                                                                                                                                                                                                                                                                       |
| 38657 | [EINVAL]  | Process is not registered to receive the SIGPOLL signal.                                                                                                                                                                                                                                                                                                                                                                                                   |
| 38658 | I_FIND    | Compares the names of all modules currently present in the STREAM to the name pointed to by <i>arg</i> , and returns 1 if the named module is present in the STREAM, or returns 0 if the named module is not present.                                                                                                                                                                                                                                      |
| 38659 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38660 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38661 |           | The <i>ioctl()</i> function with the I_FIND command shall fail if:                                                                                                                                                                                                                                                                                                                                                                                         |
| 38662 | [EINVAL]  | <i>arg</i> does not contain a valid module name.                                                                                                                                                                                                                                                                                                                                                                                                           |
| 38663 | I_PEEK    | Retrieves the information in the first message on the STREAM head read queue without taking the message off the queue. It is analogous to <i>getmsg()</i> except that this command does not remove the message from the queue. The <i>arg</i> argument points to a <b>strpeek</b> structure.                                                                                                                                                               |
| 38664 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38665 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38666 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38667 |           | The application shall ensure that the <i>maxlen</i> member in the <b>ctlbuf</b> and <b>databuf</b> structures is set to the number of bytes of control information and/or data information, respectively, to retrieve. The <i>flags</i> member may be marked RS_HIPRI or 0, as described by <i>getmsg()</i> . If the process sets <i>flags</i> to RS_HIPRI, for example, I_PEEK shall only look for a high-priority message on the STREAM head read queue. |
| 38668 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38669 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38670 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38671 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38672 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38673 |           | I_PEEK returns 1 if a message was retrieved, and returns 0 if no message was found on the STREAM head read queue, or if the RS_HIPRI flag was set in <i>flags</i> and a high-priority message was not present on the STREAM head read queue. It does not wait for a message to arrive. On return, <b>ctlbuf</b> specifies information in the control buffer, <b>databuf</b> specifies information in the data                                              |
| 38674 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38675 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38676 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 38677 |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

|       |            |                                                                                                      |
|-------|------------|------------------------------------------------------------------------------------------------------|
| 38678 |            | buffer, and <i>flags</i> contains the value RS_HIPRI or 0.                                           |
| 38679 | I_SRDOPT   | Sets the read mode using the value of the argument <i>arg</i> . Read modes are                       |
| 38680 |            | described in <i>read()</i> . Valid <i>arg</i> flags are:                                             |
| 38681 |            | RNORM           Byte-stream mode, the default.                                                       |
| 38682 |            | RMSGD           Message-discard mode.                                                                |
| 38683 |            | RMSGN           Message-nondiscard mode.                                                             |
| 38684 |            | The bitwise-inclusive OR of RMSGD and RMSGN shall return [EINVAL]. The                               |
| 38685 |            | bitwise-inclusive OR of RNORM and either RMSGD or RMSGN shall result in                              |
| 38686 |            | the other flag overriding RNORM which is the default.                                                |
| 38687 |            | In addition, treatment of control messages by the STREAM head may be                                 |
| 38688 |            | changed by setting any of the following flags in <i>arg</i> :                                        |
| 38689 |            | RPROTNORM   Fail <i>read()</i> with [EBADMSG] if a message containing a                              |
| 38690 |            | control part is at the front of the STREAM head read queue.                                          |
| 38691 |            | RPROTDAT     Deliver the control part of a message as data when a process                            |
| 38692 |            | issues a <i>read()</i> .                                                                             |
| 38693 |            | RPROTDIS     Discard the control part of a message, delivering any data                              |
| 38694 |            | portion, when a process issues a <i>read()</i> .                                                     |
| 38695 |            | The <i>ioctl()</i> function with the I_SRDOPT command shall fail if:                                 |
| 38696 |            | [EINVAL]      The <i>arg</i> argument is not valid.                                                  |
| 38697 | I_GRDOPT   | Returns the current read mode setting, as described above, in an <b>int</b> pointed to               |
| 38698 |            | by the argument <i>arg</i> . Read modes are described in <i>read()</i> .                             |
| 38699 | I_NREAD    | Counts the number of data bytes in the data part of the first message on the                         |
| 38700 |            | STREAM head read queue and places this value in the <b>int</b> pointed to by <i>arg</i> .            |
| 38701 |            | The return value for the command shall be the number of messages on the                              |
| 38702 |            | STREAM head read queue. For example, if 0 is returned in <i>arg</i> , but the <i>ioctl()</i>         |
| 38703 |            | return value is greater than 0, this indicates that a zero-length message is next                    |
| 38704 |            | on the queue.                                                                                        |
| 38705 | I_FDINSERT | Creates a message from specified buffer(s), adds information about another                           |
| 38706 |            | STREAM, and sends the message downstream. The message contains a                                     |
| 38707 |            | control part and an optional data part. The data and control parts to be sent                        |
| 38708 |            | are distinguished by placement in separate buffers, as described below. The                          |
| 38709 |            | <i>arg</i> argument points to a <b>strfdinsert</b> structure.                                        |
| 38710 |            | The application shall ensure that the <i>len</i> member in the <b>ctlbuf</b> <b>strbuf</b> structure |
| 38711 |            | is set to the size of a <b>t_uscalar_t</b> plus the number of bytes of control                       |
| 38712 |            | information to be sent with the message. The <i>fildev</i> member specifies the file                 |
| 38713 |            | descriptor of the other STREAM, and the <i>offset</i> member, which must be                          |
| 38714 |            | suitably aligned for use as a <b>t_uscalar_t</b> , specifies the offset from the start of            |
| 38715 |            | the control buffer where I_FDINSERT shall store a <b>t_uscalar_t</b> whose                           |
| 38716 |            | interpretation is specific to the STREAM end. The application shall ensure that                      |
| 38717 |            | the <i>len</i> member in the <b>databuf</b> <b>strbuf</b> structure is set to the number of bytes of |
| 38718 |            | data information to be sent with the message, or to 0 if no data part is to be                       |
| 38719 |            | sent.                                                                                                |
| 38720 |            | The <i>flags</i> member specifies the type of message to be created. A normal                        |
| 38721 |            | message is created if <i>flags</i> is set to 0, and a high-priority message is created if            |

|       |                     |                                                                                        |
|-------|---------------------|----------------------------------------------------------------------------------------|
| 38722 |                     | <i>flags</i> is set to RS_HIPRI. For non-priority messages, I_FDINSERT shall block if  |
| 38723 |                     | the STREAM write queue is full due to internal flow control conditions. For            |
| 38724 |                     | priority messages, I_FDINSERT does not block on this condition. For non-               |
| 38725 |                     | priority messages, I_FDINSERT does not block when the write queue is full              |
| 38726 |                     | and O_NONBLOCK is set. Instead, it fails and sets <i>errno</i> to [EAGAIN].            |
| 38727 |                     | I_FDINSERT also blocks, unless prevented by lack of internal resources,                |
| 38728 |                     | waiting for the availability of message blocks in the STREAM, regardless of            |
| 38729 |                     | priority or whether O_NONBLOCK has been specified. No partial message is               |
| 38730 |                     | sent.                                                                                  |
| 38731 |                     | The <i>ioctl()</i> function with the I_FDINSERT command shall fail if:                 |
| 38732 | [EAGAIN]            | A non-priority message is specified, the O_NONBLOCK                                    |
| 38733 |                     | flag is set, and the STREAM write queue is full due to                                 |
| 38734 |                     | internal flow control conditions.                                                      |
| 38735 | [EAGAIN] or [ENOSR] |                                                                                        |
| 38736 |                     | Buffers cannot be allocated for the message that is to be                              |
| 38737 |                     | created.                                                                               |
| 38738 | [EINVAL]            | One of the following:                                                                  |
| 38739 |                     | — The <i>fildev</i> member of the <b>strfdinsert</b> structure is not a                |
| 38740 |                     | valid, open STREAM file descriptor.                                                    |
| 38741 |                     | — The size of a <b>t_uscalar_t</b> plus <i>offset</i> is greater than the              |
| 38742 |                     | <i>len</i> member for the buffer specified through <b>ctlbuf</b> .                     |
| 38743 |                     | — The <i>offset</i> member does not specify a properly-aligned                         |
| 38744 |                     | location in the data buffer.                                                           |
| 38745 |                     | — An undefined value is stored in <i>flags</i> .                                       |
| 38746 | [ENXIO]             | Hangup received on the STREAM identified by either the                                 |
| 38747 |                     | <i>fildev</i> argument or the <i>fildev</i> member of the <b>strfdinsert</b>           |
| 38748 |                     | structure.                                                                             |
| 38749 | [ERANGE]            | The <i>len</i> member for the buffer specified through <b>databuf</b>                  |
| 38750 |                     | does not fall within the range specified by the maximum                                |
| 38751 |                     | and minimum packet sizes of the topmost STREAM                                         |
| 38752 |                     | module; or the <i>len</i> member for the buffer specified through                      |
| 38753 |                     | <b>databuf</b> is larger than the maximum configured size of the                       |
| 38754 |                     | data part of a message; or the <i>len</i> member for the buffer                        |
| 38755 |                     | specified through <b>ctlbuf</b> is larger than the maximum                             |
| 38756 |                     | configured size of the control part of a message.                                      |
| 38757 | I_STR               | Constructs an internal STREAMS <i>ioctl()</i> message from the data pointed to by      |
| 38758 |                     | <i>arg</i> , and sends that message downstream.                                        |
| 38759 |                     | This mechanism is provided to send <i>ioctl()</i> requests to downstream modules       |
| 38760 |                     | and drivers. It allows information to be sent with <i>ioctl()</i> , and returns to the |
| 38761 |                     | process any information sent upstream by the downstream recipient. I_STR               |
| 38762 |                     | shall block until the system responds with either a positive or negative               |
| 38763 |                     | acknowledgement message, or until the request times out after some period of           |
| 38764 |                     | time. If the request times out, it shall fail with <i>errno</i> set to [ETIME].        |
| 38765 |                     | At most, one I_STR can be active on a STREAM. Further I_STR calls shall                |
| 38766 |                     | block until the active I_STR completes at the STREAM head. The default                 |

|       |          |                                                                                         |
|-------|----------|-----------------------------------------------------------------------------------------|
| 38767 |          | timeout interval for these requests is 15 seconds. The O_NONBLOCK flag has              |
| 38768 |          | no effect on this call.                                                                 |
| 38769 |          | To send requests downstream, the application shall ensure that <i>arg</i> points to a   |
| 38770 |          | <b>strioc</b> structure.                                                                |
| 38771 |          | The <i>ic_cmd</i> member is the internal <i>ioctl()</i> command intended for a          |
| 38772 |          | downstream module or driver and <i>ic_timeout</i> is the number of seconds              |
| 38773 |          | (−1=infinite, 0=use implementation-defined timeout interval, >0=as specified)           |
| 38774 |          | an I_STR request shall wait for acknowledgement before timing out. <i>ic_len</i> is     |
| 38775 |          | the number of bytes in the data argument, and <i>ic_dp</i> is a pointer to the data     |
| 38776 |          | argument. The <i>ic_len</i> member has two uses: on input, it contains the length of    |
| 38777 |          | the data argument passed in, and on return from the command, it contains the            |
| 38778 |          | number of bytes being returned to the process (the buffer pointed to by <i>ic_dp</i>    |
| 38779 |          | should be large enough to contain the maximum amount of data that any                   |
| 38780 |          | module or the driver in the STREAM can return).                                         |
| 38781 |          | The STREAM head shall convert the information pointed to by the <b>strioc</b>           |
| 38782 |          | structure to an internal <i>ioctl()</i> command message and send it downstream.         |
| 38783 |          | The <i>ioctl()</i> function with the I_STR command shall fail if:                       |
| 38784 |          | [EAGAIN] or [ENOSR]                                                                     |
| 38785 |          | Unable to allocate buffers for the <i>ioctl()</i> message.                              |
| 38786 |          | [EINVAL] The <i>ic_len</i> member is less than 0 or larger than the                     |
| 38787 |          | maximum configured size of the data part of a message, or                               |
| 38788 |          | <i>ic_timeout</i> is less than −1.                                                      |
| 38789 |          | [ENXIO] Hangup received on <i>fil</i> .                                                 |
| 38790 |          | [ETIME] A downstream <i>ioctl()</i> timed out before acknowledgement                    |
| 38791 |          | was received.                                                                           |
| 38792 |          | An I_STR can also fail while waiting for an acknowledgement if a message                |
| 38793 |          | indicating an error or a hangup is received at the STREAM head. In addition,            |
| 38794 |          | an error code can be returned in the positive or negative acknowledgement               |
| 38795 |          | message, in the event the <i>ioctl()</i> command sent downstream fails. For these       |
| 38796 |          | cases, I_STR shall fail with <i>errno</i> set to the value in the message.              |
| 38797 | I_SWROPT | Sets the write mode using the value of the argument <i>arg</i> . Valid bit settings for |
| 38798 |          | <i>arg</i> are:                                                                         |
| 38799 |          | SNDZERO Send a zero-length message downstream when a <i>write()</i> of 0                |
| 38800 |          | bytes occurs. To not send a zero-length message when a                                  |
| 38801 |          | <i>write()</i> of 0 bytes occurs, the application shall ensure that                     |
| 38802 |          | this bit is not set in <i>arg</i> (for example, <i>arg</i> would be set to 0).          |
| 38803 |          | The <i>ioctl()</i> function with the I_SWROPT command shall fail if:                    |
| 38804 |          | [EINVAL] <i>arg</i> is not the above value.                                             |
| 38805 | I_GWROPT | Returns the current write mode setting, as described above, in the <b>int</b> that is   |
| 38806 |          | pointed to by the argument <i>arg</i> .                                                 |
| 38807 | I_SENDFD | Creates a new reference to the open file description associated with the file           |
| 38808 |          | descriptor <i>arg</i> , and writes a message on the STREAMS-based pipe <i>fil</i>       |
| 38809 |          | containing this reference, together with the user ID and group ID of the calling        |
| 38810 |          | process.                                                                                |

|       |          |                                                                                                    |
|-------|----------|----------------------------------------------------------------------------------------------------|
| 38811 |          | The <i>ioctl()</i> function with the I_SENDFD command shall fail if:                               |
| 38812 |          | [EAGAIN] The sending STREAM is unable to allocate a message block                                  |
| 38813 |          | to contain the file pointer; or the read queue of the receiving                                    |
| 38814 |          | STREAM head is full and cannot accept the message sent by                                          |
| 38815 |          | I_SENDFD.                                                                                          |
| 38816 |          | [EBADF] The <i>arg</i> argument is not a valid, open file descriptor.                              |
| 38817 |          | [EINVAL] The <i>fildev</i> argument is not connected to a STREAM pipe.                             |
| 38818 |          | [ENXIO] Hangup received on <i>fildev</i> .                                                         |
| 38819 |          | The <i>ioctl()</i> function with the I_SENDFD command may fail if:                                 |
| 38820 |          | [EINVAL] The <i>arg</i> argument is equal to the <i>fildev</i> argument.                           |
| 38821 | I_RECVFD | Retrieves the reference to an open file description from a message written to a                    |
| 38822 |          | STREAMS-based pipe using the I_SENDFD command, and allocates a new file                            |
| 38823 |          | descriptor in the calling process that refers to this open file description. The                   |
| 38824 |          | <i>arg</i> argument is a pointer to a <b>strrecvfd</b> data structure as defined in                |
| 38825 |          | <b>&lt;stropts.h&gt;</b> .                                                                         |
| 38826 |          | The <i>fd</i> member is a file descriptor. The <i>uid</i> and <i>gid</i> members are the effective |
| 38827 |          | user ID and effective group ID, respectively, of the sending process.                              |
| 38828 |          | If O_NONBLOCK is not set, I_RECVFD shall block until a message is present                          |
| 38829 |          | at the STREAM head. If O_NONBLOCK is set, I_RECVFD shall fail with <i>errno</i>                    |
| 38830 |          | set to [EAGAIN] if no message is present at the STREAM head.                                       |
| 38831 |          | If the message at the STREAM head is a message sent by an I_SENDFD, a new                          |
| 38832 |          | file descriptor shall be allocated for the open file descriptor referenced in the                  |
| 38833 |          | message. The new file descriptor is placed in the <i>fd</i> member of the <b>strrecvfd</b>         |
| 38834 |          | structure pointed to by <i>arg</i> .                                                               |
| 38835 |          | The <i>ioctl()</i> function with the I_RECVFD command shall fail if:                               |
| 38836 |          | [EAGAIN] A message is not present at the STREAM head read queue                                    |
| 38837 |          | and the O_NONBLOCK flag is set.                                                                    |
| 38838 |          | [EBADMSG] The message at the STREAM head read queue is not a                                       |
| 38839 |          | message containing a passed file descriptor.                                                       |
| 38840 |          | [EMFILE] All file descriptors available to the process are currently                               |
| 38841 |          | open.                                                                                              |
| 38842 |          | [ENXIO] Hangup received on <i>fildev</i> .                                                         |
| 38843 | I_LIST   | Allows the process to list all the module names on the STREAM, up to and                           |
| 38844 |          | including the topmost driver name. If <i>arg</i> is a null pointer, the return value               |
| 38845 |          | shall be the number of modules, including the driver, that are on the STREAM                       |
| 38846 |          | pointed to by <i>fildev</i> . This lets the process allocate enough space for the module           |
| 38847 |          | names. Otherwise, it should point to a <b>str_list</b> structure.                                  |
| 38848 |          | The <i>sl_nmods</i> member indicates the number of entries the process has                         |
| 38849 |          | allocated in the array. Upon return, the <i>sl_modlist</i> member of the <b>str_list</b>           |
| 38850 |          | structure shall contain the list of module names, and the number of entries                        |
| 38851 |          | that have been filled into the <i>sl_modlist</i> array is found in the <i>sl_nmods</i> member      |
| 38852 |          | (the number includes the number of modules including the driver). The return                       |
| 38853 |          | value from <i>ioctl()</i> shall be 0. The entries are filled in starting at the top of the         |

|       |                    |                                                                                           |
|-------|--------------------|-------------------------------------------------------------------------------------------|
| 38854 |                    | STREAM and continuing downstream until either the end of the STREAM is                    |
| 38855 |                    | reached, or the number of requested modules ( <i>sl_nmods</i> ) is satisfied.             |
| 38856 |                    | The <i>ioctl()</i> function with the <i>I_LIST</i> command shall fail if:                 |
| 38857 |                    | [EINVAL]           The <i>sl_nmods</i> member is less than 1.                             |
| 38858 |                    | [EAGAIN] or [ENOSR]                                                                       |
| 38859 |                    | Unable to allocate buffers.                                                               |
| 38860 | <i>I_ATMARK</i>    | Allows the process to see if the message at the head of the STREAM head read              |
| 38861 |                    | queue is marked by some module downstream. The <i>arg</i> argument determines             |
| 38862 |                    | how the checking is done when there may be multiple marked messages on                    |
| 38863 |                    | the STREAM head read queue. It may take on the following values:                          |
| 38864 |                    | ANYMARK           Check if the message is marked.                                         |
| 38865 |                    | LASTMARK          Check if the message is the last one marked on the queue.               |
| 38866 |                    | The bitwise-inclusive OR of the flags ANYMARK and LASTMARK is                             |
| 38867 |                    | permitted.                                                                                |
| 38868 |                    | The return value shall be 1 if the mark condition is satisfied; otherwise, the            |
| 38869 |                    | value shall be 0.                                                                         |
| 38870 |                    | The <i>ioctl()</i> function with the <i>I_ATMARK</i> command shall fail if:               |
| 38871 |                    | [EINVAL]           Invalid <i>arg</i> value.                                              |
| 38872 | <i>I_CKBAND</i>    | Checks if the message of a given priority band exists on the STREAM head                  |
| 38873 |                    | read queue. This shall return 1 if a message of the given priority exists, 0 if no        |
| 38874 |                    | such message exists, or –1 on error. <i>arg</i> should be of type <b>int</b> .            |
| 38875 |                    | The <i>ioctl()</i> function with the <i>I_CKBAND</i> command shall fail if:               |
| 38876 |                    | [EINVAL]           Invalid <i>arg</i> value.                                              |
| 38877 | <i>I_GETBAND</i>   | Returns the priority band of the first message on the STREAM head read                    |
| 38878 |                    | queue in the integer referenced by <i>arg</i> .                                           |
| 38879 |                    | The <i>ioctl()</i> function with the <i>I_GETBAND</i> command shall fail if:              |
| 38880 |                    | [ENODATA]          No message on the STREAM head read queue.                              |
| 38881 | <i>I_CANPUT</i>    | Checks if a certain band is writable. <i>arg</i> is set to the priority band in question. |
| 38882 |                    | The return value shall be 0 if the band is flow-controlled, 1 if the band is              |
| 38883 |                    | writable, or –1 on error.                                                                 |
| 38884 |                    | The <i>ioctl()</i> function with the <i>I_CANPUT</i> command shall fail if:               |
| 38885 |                    | [EINVAL]           Invalid <i>arg</i> value.                                              |
| 38886 | <i>I_SETCLTIME</i> | This request allows the process to set the time the STREAM head shall delay               |
| 38887 |                    | when a STREAM is closing and there is data on the write queues. Before                    |
| 38888 |                    | closing each module or driver, if there is data on its write queue, the STREAM            |
| 38889 |                    | head shall delay for the specified amount of time to allow the data to drain. If,         |
| 38890 |                    | after the delay, data is still present, it shall be flushed. The <i>arg</i> argument is a |
| 38891 |                    | pointer to an integer specifying the number of milliseconds to delay, rounded             |
| 38892 |                    | up to the nearest valid value. If <i>I_SETCLTIME</i> is not performed on a STREAM,        |
| 38893 |                    | an implementation-defined default timeout interval is used.                               |



38894 The *ioctl()* function with the *I\_SETCLTIME* command shall fail if:

38895 [EINVAL] Invalid *arg* value.

38896 *I\_GETCLTIME* Returns the close time delay in the integer pointed to by *arg*.

### 38897 Multiplexed STREAMS Configurations

38898 The following commands are used for connecting and disconnecting multiplexed STREAMS  
38899 configurations. These commands use an implementation-defined default timeout interval.

38900 *I\_LINK* Connects two STREAMs, where *fildev* is the file descriptor of the STREAM  
38901 connected to the multiplexing driver, and *arg* is the file descriptor of the  
38902 STREAM connected to another driver. The STREAM designated by *arg* is  
38903 connected below the multiplexing driver. *I\_LINK* requires the multiplexing  
38904 driver to send an acknowledgement message to the STREAM head regarding  
38905 the connection. This call shall return a multiplexer ID number (an identifier  
38906 used to disconnect the multiplexer; see *I\_UNLINK*) on success, and *-1* on  
38907 failure.

38908 The *ioctl()* function with the *I\_LINK* command shall fail if:

38909 [ENXIO] Hangup received on *fildev*.

38910 [ETIME] Timeout before acknowledgement message was received at  
38911 STREAM head.

38912 [EAGAIN] or [ENOSR]

38913 Unable to allocate STREAMS storage to perform the  
38914 *I\_LINK*.

38915 [EBADF] The *arg* argument is not a valid, open file descriptor.

38916 [EINVAL] The *fildev* argument does not support multiplexing; or *arg* is  
38917 not a STREAM or is already connected downstream from a  
38918 multiplexer; or the specified *I\_LINK* operation would  
38919 connect the STREAM head in more than one place in the  
38920 multiplexed STREAM.

38921 An *I\_LINK* can also fail while waiting for the multiplexing driver to  
38922 acknowledge the request, if a message indicating an error or a hangup is  
38923 received at the STREAM head of *fildev*. In addition, an error code can be  
38924 returned in the positive or negative acknowledgement message. For these  
38925 cases, *I\_LINK* fails with *errno* set to the value in the message.

38926 *I\_UNLINK* Disconnects the two STREAMs specified by *fildev* and *arg*. *fildev* is the file  
38927 descriptor of the STREAM connected to the multiplexing driver. The *arg*  
38928 argument is the multiplexer ID number that was returned by the *I\_LINK*  
38929 *ioctl()* command when a STREAM was connected downstream from the  
38930 multiplexing driver. If *arg* is *MUXID\_ALL*, then all STREAMs that were  
38931 connected to *fildev* shall be disconnected. As in *I\_LINK*, this command requires  
38932 acknowledgement.

38933 The *ioctl()* function with the *I\_UNLINK* command shall fail if:

38934 [ENXIO] Hangup received on *fildev*.

|       |           |                     |                                                                                             |
|-------|-----------|---------------------|---------------------------------------------------------------------------------------------|
| 38935 |           | [ETIME]             | Timeout before acknowledgement message was received at                                      |
| 38936 |           |                     | STREAM head.                                                                                |
| 38937 |           | [EAGAIN] or [ENOSR] |                                                                                             |
| 38938 |           |                     | Unable to allocate buffers for the acknowledgement                                          |
| 38939 |           |                     | message.                                                                                    |
| 38940 |           | [EINVAL]            | Invalid multiplexer ID number.                                                              |
| 38941 |           |                     | An I_UNLINK can also fail while waiting for the multiplexing driver to                      |
| 38942 |           |                     | acknowledge the request if a message indicating an error or a hangup is                     |
| 38943 |           |                     | received at the STREAM head of <i>fildev</i> . In addition, an error code can be            |
| 38944 |           |                     | returned in the positive or negative acknowledgement message. For these                     |
| 38945 |           |                     | cases, I_UNLINK shall fail with <i>errno</i> set to the value in the message.               |
| 38946 | I_PLINK   |                     | Creates a <i>persistent connection</i> between two STREAMs, where <i>fildev</i> is the file |
| 38947 |           |                     | descriptor of the STREAM connected to the multiplexing driver, and <i>arg</i> is the        |
| 38948 |           |                     | file descriptor of the STREAM connected to another driver. This call shall                  |
| 38949 |           |                     | create a persistent connection which can exist even if the file descriptor <i>fildev</i>    |
| 38950 |           |                     | associated with the upper STREAM to the multiplexing driver is closed. The                  |
| 38951 |           |                     | STREAM designated by <i>arg</i> gets connected via a persistent connection below            |
| 38952 |           |                     | the multiplexing driver. I_PLINK requires the multiplexing driver to send an                |
| 38953 |           |                     | acknowledgement message to the STREAM head. This call shall return a                        |
| 38954 |           |                     | multiplexer ID number (an identifier that may be used to disconnect the                     |
| 38955 |           |                     | multiplexer; see I_PUNLINK) on success, and -1 on failure.                                  |
| 38956 |           |                     | The <i>ioctl()</i> function with the I_PLINK command shall fail if:                         |
| 38957 |           | [ENXIO]             | Hangup received on <i>fildev</i> .                                                          |
| 38958 |           | [ETIME]             | Timeout before acknowledgement message was received at                                      |
| 38959 |           |                     | STREAM head.                                                                                |
| 38960 |           | [EAGAIN] or [ENOSR] |                                                                                             |
| 38961 |           |                     | Unable to allocate STREAMS storage to perform the                                           |
| 38962 |           |                     | I_PLINK.                                                                                    |
| 38963 |           | [EBADF]             | The <i>arg</i> argument is not a valid, open file descriptor.                               |
| 38964 |           | [EINVAL]            | The <i>fildev</i> argument does not support multiplexing; or <i>arg</i> is                  |
| 38965 |           |                     | not a STREAM or is already connected downstream from a                                      |
| 38966 |           |                     | multiplexer; or the specified I_PLINK operation would                                       |
| 38967 |           |                     | connect the STREAM head in more than one place in the                                       |
| 38968 |           |                     | multiplexed STREAM.                                                                         |
| 38969 |           |                     | An I_PLINK can also fail while waiting for the multiplexing driver to                       |
| 38970 |           |                     | acknowledge the request, if a message indicating an error or a hangup is                    |
| 38971 |           |                     | received at the STREAM head of <i>fildev</i> . In addition, an error code can be            |
| 38972 |           |                     | returned in the positive or negative acknowledgement message. For these                     |
| 38973 |           |                     | cases, I_PLINK shall fail with <i>errno</i> set to the value in the message.                |
| 38974 | I_PUNLINK |                     | Disconnects the two STREAMs specified by <i>fildev</i> and <i>arg</i> from a persistent     |
| 38975 |           |                     | connection. The <i>fildev</i> argument is the file descriptor of the STREAM                 |
| 38976 |           |                     | connected to the multiplexing driver. The <i>arg</i> argument is the multiplexer ID         |
| 38977 |           |                     | number that was returned by the I_PLINK <i>ioctl()</i> command when a STREAM                |
| 38978 |           |                     | was connected downstream from the multiplexing driver. If <i>arg</i> is                     |
| 38979 |           |                     | MUXID_ALL, then all STREAMs which are persistent connections to <i>fildev</i>               |
| 38980 |           |                     | shall be disconnected. As in I_PLINK, this command requires the multiplexing                |



38981 driver to acknowledge the request.

38982 The *ioctl()* function with the *I\_PUNLINK* command shall fail if:

38983 [ENXIO] Hangup received on *fildev*.

38984 [ETIME] Timeout before acknowledgement message was received at

38985 STREAM head.

38986 [EAGAIN] or [ENOSR]

38987 Unable to allocate buffers for the acknowledgement

38988 message.

38989 [EINVAL] Invalid multiplexer ID number.

38990 An *I\_PUNLINK* can also fail while waiting for the multiplexing driver to

38991 acknowledge the request if a message indicating an error or a hangup is

38992 received at the STREAM head of *fildev*. In addition, an error code can be

38993 returned in the positive or negative acknowledgement message. For these

38994 cases, *I\_PUNLINK* shall fail with *errno* set to the value in the message.

#### 38995 RETURN VALUE

38996 Upon successful completion, *ioctl()* shall return a value other than *-1* that depends upon the

38997 STREAMS device control function. Otherwise, it shall return *-1* and set *errno* to indicate the

38998 error.

#### 38999 ERRORS

39000 Under the following general conditions, *ioctl()* shall fail if:

39001 [EBADF] The *fildev* argument is not a valid open file descriptor.

39002 [EINTR] A signal was caught during the *ioctl()* operation.

39003 [EINVAL] The STREAM or multiplexer referenced by *fildev* is linked (directly or

39004 indirectly) downstream from a multiplexer.

39005 If an underlying device driver detects an error, then *ioctl()* shall fail if:

39006 [EINVAL] The *request* or *arg* argument is not valid for this device.

39007 [EIO] Some physical I/O error has occurred.

39008 [ENOTTY] The file associated with the *fildev* argument is not a STREAMS device that

39009 accepts control functions.

39010 [ENXIO] The *request* and *arg* arguments are valid for this device driver, but the service

39011 requested cannot be performed on this particular sub-device.

39012 [ENODEV] The *fildev* argument refers to a valid STREAMS device, but the corresponding

39013 device driver does not support the *ioctl()* function.

39014 If a STREAM is connected downstream from a multiplexer, any *ioctl()* command except

39015 *I\_UNLINK* and *I\_PUNLINK* shall set *errno* to [EINVAL].

39016 **EXAMPLES**

39017 None.

39018 **APPLICATION USAGE**

39019 The implementation-defined timeout interval for STREAMS has historically been 15 seconds.

39020 **RATIONALE**

39021 None.

39022 **FUTURE DIRECTIONS**

39023 The *ioctl()* function may be removed in a future version.

39024 **SEE ALSO**

39025 Section 2.6 (on page 500), *close()*, *fcntl()*, *getmsg()*, *open()*, *pipe()*, *poll()*, *putmsg()*, *read()*,  
39026 *sigaction()*, *write()*

39027 XBD <stropts.h>

39028 **CHANGE HISTORY**

39029 First released in Issue 4, Version 2.

39030 **Issue 5**

39031 Moved from X/OPEN UNIX extension to BASE.

39032 **Issue 6**

39033 The Open Group Corrigendum U028/4 is applied, correcting text in the I\_FDINSERT [EINVAL]  
39034 case to refer to *ctlbuf*.

39035 This function is marked as part of the XSI STREAMS Option Group.

39036 The normative text is updated to avoid use of the term “must” for application requirements.

39037 **Issue 7**

39038 Austin Group Interpretation 1003.1-2001 #155 is applied, adding a “may fail” [EINVAL] error  
39039 condition for the I\_SENDFD command.

39040 SD5-XSH-ERN-100 is applied, correcting the definition of the [ENOTTY] error condition.

39041 The *ioctl()* function is marked obsolescent.

39042 **NAME**

39043 isalnum, isalnum\_l — test for an alphanumeric character

39044 **SYNOPSIS**

39045 #include &lt;ctype.h&gt;

39046 int isalnum(int c);

39047 CX int isalnum\_l(int c, locale\_t locale);

39048 **DESCRIPTION**

39049 CX For *isalnum()*: The functionality described on this reference page is aligned with the ISO C  
 39050 standard. Any conflict between the requirements described here and the ISO C standard is  
 39051 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39052 CX The *isalnum()* and *isalnum\_l()* functions shall test whether *c* is a character of class **alpha** or  
 39053 CX **digit** in the current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7  
 39054 (on page 135).

39055 The *c* argument is an **int**, the value of which the application shall ensure is representable as an  
 39056 **unsigned char** or equal to the value of the macro EOF. If the argument has any other value, the  
 39057 behavior is undefined.

39058 CX The behavior is undefined if the *locale* argument to *isalnum\_l()* is the special locale object  
 39059 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

39060 **RETURN VALUE**

39061 CX The *isalnum()* and *isalnum\_l()* functions shall return non-zero if *c* is an alphanumeric character;  
 39062 otherwise, they shall return 0.

39063 **ERRORS**

39064 No errors are defined.

39065 **EXAMPLES**

39066 None.

39067 **APPLICATION USAGE**

39068 To ensure applications portability, especially across natural languages, only these functions and  
 39069 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 39070 classification.

39071 **RATIONALE**

39072 None.

39073 **FUTURE DIRECTIONS**

39074 None.

39075 **SEE ALSO**

39076 *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*, *isupper()*,  
 39077 *isxdigit()*, *setlocale()*, *uselocale()*

39078 XBD Chapter 7 (on page 135), &lt;ctype.h&gt;, &lt;stdio.h&gt;

39079 **CHANGE HISTORY**

39080 First released in Issue 1. Derived from Issue 1 of the SVID.

39081 **Issue 6**

39082 The normative text is updated to avoid use of the term “must” for application requirements.

39083 **Issue 7**

39084 The *isalnum\_l()* function is added from The Open Group Technical Standard, 2006, Extended  
39085 API Set Part 4.

39086 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0274 [302], XSH/TC1-2008/0275 [283],  
39087 and XSH/TC1-2008/0276 [283] are applied.

39088 **NAME**

39089 isalpha, isalpha\_l — test for an alphabetic character

39090 **SYNOPSIS**

```
39091     #include <ctype.h>
39092     int isalpha(int c);
39093 CX    int isalpha_l(int c, locale_t locale);
```

39094 **DESCRIPTION**

39095 CX For *isalpha()*: The functionality described on this reference page is aligned with the ISO C  
 39096 standard. Any conflict between the requirements described here and the ISO C standard is  
 39097 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39098 CX The *isalpha()* and *isalpha\_l()* functions shall test whether *c* is a character of class **alpha** in the  
 39099 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page  
 39100 135).

39101 The *c* argument is an **int**, the value of which the application shall ensure is representable as an  
 39102 **unsigned char** or equal to the value of the macro EOF. If the argument has any other value, the  
 39103 behavior is undefined.

39104 CX The behavior is undefined if the *locale* argument to *isalpha\_l()* is the special locale object  
 39105 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

39106 **RETURN VALUE**

39107 CX The *isalpha()* and *isalpha\_l()* functions shall return non-zero if *c* is an alphabetic character;  
 39108 otherwise, they shall return 0.

39109 **ERRORS**

39110 No errors are defined.

39111 **EXAMPLES**

39112 None.

39113 **APPLICATION USAGE**

39114 To ensure applications portability, especially across natural languages, only these functions and  
 39115 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 39116 classification.

39117 **RATIONALE**

39118 None.

39119 **FUTURE DIRECTIONS**

39120 None.

39121 **SEE ALSO**

39122 *isalnum()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*, *isupper()*,  
 39123 *isxdigit()*, *setlocale()*, *uselocale()*

39124 XBD Chapter 7 (on page 135), [<ctype.h>](#), [<locale.h>](#), [<stdio.h>](#)39125 **CHANGE HISTORY**

39126 First released in Issue 1. Derived from Issue 1 of the SVID.

39127 **Issue 6**

39128 The normative text is updated to avoid use of the term “must” for application requirements.

39129 **Issue 7**

39130 The *isalpha\_l()* function is added from The Open Group Technical Standard, 2006, Extended API  
39131 Set Part 4.

39132 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0277 [302], XSH/TC1-2008/0278 [283],  
39133 and XSH/TC1-2008/0279 [283] are applied.

39134 **NAME**

39135 isascii — test for a 7-bit US-ASCII character

39136 **SYNOPSIS**

```
39137 OB XSI #include <ctype.h>
39138 int isascii(int c);
```

39139 **DESCRIPTION**39140 The *isascii()* function shall test whether *c* is a 7-bit US-ASCII character code.39141 The *isascii()* function is defined on all integer values.39142 **RETURN VALUE**

39143 The *isascii()* function shall return non-zero if *c* is a 7-bit US-ASCII character code between 0 and  
 39144 octal 0177 inclusive; otherwise, it shall return 0.

39145 **ERRORS**

39146 No errors are defined.

39147 **EXAMPLES**

39148 None.

39149 **APPLICATION USAGE**39150 The *isascii()* function cannot be used portably in a localized application.39151 **RATIONALE**

39152 None.

39153 **FUTURE DIRECTIONS**39154 The *isascii()* function may be removed in a future version.39155 **SEE ALSO**39156 XBD [<ctype.h>](#)39157 **CHANGE HISTORY**

39158 First released in Issue 1. Derived from Issue 1 of the SVID.

39159 **Issue 7**39160 The *isascii()* function is marked obsolescent.

39161 **NAME**39162 isastream — test a file descriptor (**STREAMS**)39163 **SYNOPSIS**

```
39164 OB XSR #include <stropts.h>
39165 int isastream(int fildes);
```

39166 **DESCRIPTION**

39167 The *isastream()* function shall test whether *fildes*, an open file descriptor, is associated with a  
39168 STREAMS-based file.

39169 **RETURN VALUE**

39170 Upon successful completion, *isastream()* shall return 1 if *fildes* refers to a STREAMS-based file  
39171 and 0 if not. Otherwise, *isastream()* shall return -1 and set *errno* to indicate the error.

39172 **ERRORS**

39173 The *isastream()* function shall fail if:

39174 [EBADF] The *fildes* argument is not a valid open file descriptor.

39175 **EXAMPLES**

39176 None.

39177 **APPLICATION USAGE**

39178 None.

39179 **RATIONALE**

39180 None.

39181 **FUTURE DIRECTIONS**

39182 The *isastream()* function may be removed in a future version.

39183 **SEE ALSO**

39184 XBD [<stropts.h>](#)

39185 **CHANGE HISTORY**

39186 First released in Issue 4, Version 2.

39187 **Issue 5**

39188 Moved from X/OPEN UNIX extension to BASE.

39189 **Issue 7**

39190 The *isastream()* function is marked obsolescent.



39191 **NAME**

39192 isatty — test for a terminal device

39193 **SYNOPSIS**

39194 #include &lt;unistd.h&gt;

39195 int isatty(int *fildes*);39196 **DESCRIPTION**39197 The *isatty()* function shall test whether *fildes*, an open file descriptor, is associated with a  
39198 terminal device.39199 **RETURN VALUE**39200 The *isatty()* function shall return 1 if *fildes* is associated with a terminal; otherwise, it shall return  
39201 0 and may set *errno* to indicate the error.39202 **ERRORS**39203 The *isatty()* function may fail if:39204 [EBADF] The *fildes* argument is not a valid open file descriptor.39205 [ENOTTY] The file associated with the *fildes* argument is not a terminal.39206 **EXAMPLES**

39207 None.

39208 **APPLICATION USAGE**39209 The *isatty()* function does not necessarily indicate that a human being is available for interaction  
39210 via *fildes*. It is quite possible that non-terminal devices are connected to the communications  
39211 line.39212 **RATIONALE**

39213 None.

39214 **FUTURE DIRECTIONS**

39215 None.

39216 **SEE ALSO**

39217 XBD &lt;unistd.h&gt;

39218 **CHANGE HISTORY**

39219 First released in Issue 1. Derived from Issue 1 of the SVID.

39220 **Issue 6**39221 The following new requirements on POSIX implementations derive from alignment with the  
39222 Single UNIX Specification:

- 39223 • The optional setting of *errno* to indicate an error is added.
- 39224 • The [EBADF] and [ENOTTY] optional error conditions are added.

39225 **Issue 7**

39226 SD5-XSH-ERN-100 is applied, correcting the definition of the [ENOTTY] error condition.

39227 **NAME**

39228 isblank, isblank\_l — test for a blank character

39229 **SYNOPSIS**

39230 #include &lt;ctype.h&gt;

39231 int isblank(int c);

39232 CX int isblank\_l(int c, locale\_t locale);

39233 **DESCRIPTION**39234 CX For *isblank()*: The functionality described on this reference page is aligned with the ISO C  
39235 standard. Any conflict between the requirements described here and the ISO C standard is  
39236 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.39237 CX The *isblank()* and *isblank\_l()* functions shall test whether *c* is a character of class **blank** in the  
39238 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page  
39239 135).39240 The *c* argument is a type **int**, the value of which the application shall ensure is a character  
39241 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has  
39242 any other value, the behavior is undefined.39243 CX The behavior is undefined if the *locale* argument to *isblank\_l()* is the special locale object  
39244 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.39245 **RETURN VALUE**39246 CX The *isblank()* and *isblank\_l()* functions shall return non-zero if *c* is a <blank>; otherwise, they  
39247 shall return 0.39248 **ERRORS**

39249 No errors are defined.

39250 **EXAMPLES**

39251 None.

39252 **APPLICATION USAGE**39253 To ensure applications portability, especially across natural languages, only these functions and  
39254 the functions in the reference pages listed in the SEE ALSO section should be used for character  
39255 classification.39256 **RATIONALE**

39257 None.

39258 **FUTURE DIRECTIONS**

39259 None.

39260 **SEE ALSO**39261 *isalnum()*, *isalpha()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*, *isupper()*,  
39262 *isxdigit()*, *setlocale()*, *uselocale()*

39263 XBD Chapter 7 (on page 135), &lt;ctype.h&gt;, &lt;locale.h&gt;

39264 **CHANGE HISTORY**

39265 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

39266 **Issue 7**39267 The *isblank\_l()* function is added from The Open Group Technical Standard, 2006, Extended API  
39268 Set Part 4.

39269  
39270

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0280 [302], XSH/TC1-2008/0281 [283], and XSH/TC1-2008/0282 [283] are applied.

39271 **NAME**

39272 isctrl, isctrl\_l — test for a control character

39273 **SYNOPSIS**

39274 #include &lt;ctype.h&gt;

39275 int isctrl(int c);

39276 CX int isctrl\_l(int c, locale\_t locale);

39277 **DESCRIPTION**39278 CX For *isctrl()*: The functionality described on this reference page is aligned with the ISO C  
39279 standard. Any conflict between the requirements described here and the ISO C standard is  
39280 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.39281 CX The *isctrl()* and *isctrl\_l()* functions shall test whether *c* is a character of class **cntrl** in the  
39282 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page  
39283 135).39284 The *c* argument is a type **int**, the value of which the application shall ensure is a character  
39285 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has  
39286 any other value, the behavior is undefined.39287 CX The behavior is undefined if the *locale* argument to *isctrl\_l()* is the special locale object  
39288 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.39289 **RETURN VALUE**39290 CX The *isctrl()* and *isctrl\_l()* functions shall return non-zero if *c* is a control character; otherwise,  
39291 they shall return 0.39292 **ERRORS**

39293 No errors are defined.

39294 **EXAMPLES**

39295 None.

39296 **APPLICATION USAGE**39297 To ensure applications portability, especially across natural languages, only these functions and  
39298 the functions in the reference pages listed in the SEE ALSO section should be used for character  
39299 classification.39300 **RATIONALE**

39301 None.

39302 **FUTURE DIRECTIONS**

39303 None.

39304 **SEE ALSO**39305 *isalnum()*, *isalpha()*, *isblank()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*,  
39306 *isupper()*, *isxdigit()*, *setlocale()*, *uselocale()*

39307 XBD Chapter 7 (on page 135), &lt;ctype.h&gt;, &lt;locale.h&gt;

39308 **CHANGE HISTORY**

39309 First released in Issue 1. Derived from Issue 1 of the SVID.

39310 **Issue 6**

39311 The normative text is updated to avoid use of the term “must” for application requirements.

39312 **Issue 7**

39313

39314

The *isctrl\_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

39315

39316

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0283 [302], XSH/TC1-2008/0284 [283], and XSH/TC1-2008/0285 [283] are applied.

39317 **NAME**

39318       isdigit, isdigit\_l — test for a decimal digit

39319 **SYNOPSIS**

```
39320       #include <ctype.h>
39321       int isdigit(int c);
39322 CX     int isdigit_l(int c, locale_t locale);
```

39323 **DESCRIPTION**

39324 CX     For *isdigit()*: The functionality described on this reference page is aligned with the ISO C  
39325 standard. Any conflict between the requirements described here and the ISO C standard is  
39326 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39327 CX     The *isdigit()* and *isdigit\_l()* functions shall test whether *c* is a character of class **digit** in the  
39328 CX     current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page  
39329 135).

39330       The *c* argument is an **int**, the value of which the application shall ensure is a character  
39331 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has  
39332 any other value, the behavior is undefined.

39333 CX     The behavior is undefined if the *locale* argument to *isdigit\_l()* is the special locale object  
39334 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

39335 **RETURN VALUE**

39336 CX     The *isdigit()* and *isdigit\_l()* functions shall return non-zero if *c* is a decimal digit; otherwise,  
39337 they shall return 0.

39338 **ERRORS**

39339       No errors are defined.

39340 **EXAMPLES**

39341       None.

39342 **APPLICATION USAGE**

39343       To ensure applications portability, especially across natural languages, only these functions and  
39344 the functions in the reference pages listed in the SEE ALSO section should be used for character  
39345 classification.

39346 **RATIONALE**

39347       None.

39348 **FUTURE DIRECTIONS**

39349       None.

39350 **SEE ALSO**

39351       *isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*,  
39352 *isupper()*, *isxdigit()*

39353       XBD Chapter 7 (on page 135), *<ctype.h>*, *<locale.h>*39354 **CHANGE HISTORY**

39355       First released in Issue 1. Derived from Issue 1 of the SVID.

39356 **Issue 6**

39357       The normative text is updated to avoid use of the term “must” for application requirements.

39358 **Issue 7**

39359

39360

The *isdigit\_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

39361

39362

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0286 [302], XSH/TC1-2008/0287 [283], and XSH/TC1-2008/0288 [283] are applied.

39363 **NAME**

39364 isfinite — test for finite value

39365 **SYNOPSIS**

39366 #include &lt;math.h&gt;

39367 int isfinite(real-floating x);

39368 **DESCRIPTION**

39369 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
39370 conflict between the requirements described here and the ISO C standard is unintentional. This  
39371 volume of POSIX.1-2008 defers to the ISO C standard.

39372 The *isfinite()* macro shall determine whether its argument has a finite value (zero, subnormal, or  
39373 normal, and not infinite or NaN). First, an argument represented in a format wider than its  
39374 semantic type is converted to its semantic type. Then determination is based on the type of the  
39375 argument.

39376 **RETURN VALUE**39377 The *isfinite()* macro shall return a non-zero value if and only if its argument has a finite value.39378 **ERRORS**

39379 No errors are defined.

39380 **EXAMPLES**

39381 None.

39382 **APPLICATION USAGE**

39383 None.

39384 **RATIONALE**

39385 None.

39386 **FUTURE DIRECTIONS**

39387 None.

39388 **SEE ALSO**39389 *fpclassify()*, *isinf()*, *isnan()*, *isnormal()*, *signbit()*

39390 XBD &lt;math.h&gt;

39391 **CHANGE HISTORY**

39392 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.



39393 **NAME**

39394 isgraph, isgraph\_l — test for a visible character

39395 **SYNOPSIS**

```
39396     #include <ctype.h>
39397     int isgraph(int c);
39398 CX   int isgraph_l(int c, locale_t locale);
```

39399 **DESCRIPTION**

39400 CX For *isgraph()*: The functionality described on this reference page is aligned with the ISO C  
 39401 standard. Any conflict between the requirements described here and the ISO C standard is  
 39402 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39403 CX The *isgraph()* and *isgraph\_l()* functions shall test whether *c* is a character of class **graph** in the  
 39404 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page  
 39405 135).

39406 The *c* argument is an **int**, the value of which the application shall ensure is a character  
 39407 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has  
 39408 any other value, the behavior is undefined.

39409 CX The behavior is undefined if the *locale* argument to *isgraph\_l()* is the special locale object  
 39410 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

39411 **RETURN VALUE**

39412 CX The *isgraph()* and *isgraph\_l()* functions shall return non-zero if *c* is a character with a visible  
 39413 representation; otherwise, they shall return 0.

39414 **ERRORS**

39415 No errors are defined.

39416 **EXAMPLES**

39417 None.

39418 **APPLICATION USAGE**

39419 To ensure applications portability, especially across natural languages, only these functions and  
 39420 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 39421 classification.

39422 **RATIONALE**

39423 None.

39424 **FUTURE DIRECTIONS**

39425 None.

39426 **SEE ALSO**

39427 *isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*, *isupper()*,  
 39428 *isxdigit()*, *setlocale()*, *uselocale()*

39429 XBD Chapter 7 (on page 135), *<ctype.h>*, *<locale.h>*39430 **CHANGE HISTORY**

39431 First released in Issue 1. Derived from Issue 1 of the SVID.

39432 **Issue 6**

39433 The normative text is updated to avoid use of the term “must” for application requirements.

39434 **Issue 7**

39435

39436

The *isgraph\_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

39437

39438

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0289 [302], XSH/TC1-2008/0290 [283], and XSH/TC1-2008/0291 [283] are applied.

39439 **NAME**39440 `isgreater` — test if  $x$  greater than  $y$ 39441 **SYNOPSIS**39442 `#include <math.h>`39443 `int isgreater(real-floating  $x$ , real-floating  $y$ );`39444 **DESCRIPTION**

39445 **CX** The functionality described on this reference page is aligned with the ISO C standard. Any  
 39446 conflict between the requirements described here and the ISO C standard is unintentional. This  
 39447 volume of POSIX.1-2008 defers to the ISO C standard.

39448 The `isgreater()` macro shall determine whether its first argument is greater than its second  
 39449 argument. The value of `isgreater( $x$ ,  $y$ )` shall be equal to  $(x) > (y)$ ; however, unlike  $(x) > (y)$ ,  
 39450 `isgreater( $x$ ,  $y$ )` shall not raise the invalid floating-point exception when  $x$  and  $y$  are unordered.

39451 **RETURN VALUE**39452 Upon successful completion, the `isgreater()` macro shall return the value of  $(x) > (y)$ .39453 If  $x$  or  $y$  is NaN, 0 shall be returned.39454 **ERRORS**

39455 No errors are defined.

39456 **EXAMPLES**

39457 None.

39458 **APPLICATION USAGE**

39459 The relational and equality operators support the usual mathematical relationships between  
 39460 numeric values. For any ordered pair of numeric values, exactly one of the relationships (less,  
 39461 greater, and equal) is true. Relational operators may raise the invalid floating-point exception  
 39462 when argument values are NaNs. For a NaN and a numeric value, or for two NaNs, just the  
 39463 unordered relationship is true. This macro is a quiet (non-floating-point exception raising)  
 39464 version of a relational operator. It facilitates writing efficient code that accounts for NaNs  
 39465 without suffering the invalid floating-point exception. In the SYNOPSIS section, **real-floating**  
 39466 indicates that the argument shall be an expression of **real-floating** type.

39467 **RATIONALE**

39468 None.

39469 **FUTURE DIRECTIONS**

39470 None.

39471 **SEE ALSO**39472 `isgreaterequal()`, `isless()`, `islessequal()`, `islessgreater()`, `isunordered()`39473 XBD `<math.h>`39474 **CHANGE HISTORY**

39475 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

39476 **NAME**39477       isgreaterequal — test if *x* is greater than or equal to *y*39478 **SYNOPSIS**

39479       #include &lt;math.h&gt;

39480       int isgreaterequal(real-floating *x*, real-floating *y*);39481 **DESCRIPTION**

39482 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 39483 conflict between the requirements described here and the ISO C standard is unintentional. This  
 39484 volume of POSIX.1-2008 defers to the ISO C standard.

39485       The *isgreaterequal()* macro shall determine whether its first argument is greater than or equal to  
 39486 its second argument. The value of *isgreaterequal(x, y)* shall be equal to  $(x) \geq (y)$ ; however, unlike  
 39487  $(x) \geq (y)$ , *isgreaterequal(x, y)* shall not raise the invalid floating-point exception when *x* and *y* are  
 39488 unordered.

39489 **RETURN VALUE**39490       Upon successful completion, the *isgreaterequal()* macro shall return the value of  $(x) \geq (y)$ .39491       If *x* or *y* is NaN, 0 shall be returned.39492 **ERRORS**

39493       No errors are defined.

39494 **EXAMPLES**

39495       None.

39496 **APPLICATION USAGE**

39497       The relational and equality operators support the usual mathematical relationships between  
 39498 numeric values. For any ordered pair of numeric values, exactly one of the relationships (less,  
 39499 greater, and equal) is true. Relational operators may raise the invalid floating-point exception  
 39500 when argument values are NaNs. For a NaN and a numeric value, or for two NaNs, just the  
 39501 unordered relationship is true. This macro is a quiet (non-floating-point exception raising)  
 39502 version of a relational operator. It facilitates writing efficient code that accounts for NaNs  
 39503 without suffering the invalid floating-point exception. In the SYNOPSIS section, **real-floating**  
 39504 indicates that the argument shall be an expression of **real-floating** type.

39505 **RATIONALE**

39506       None.

39507 **FUTURE DIRECTIONS**

39508       None.

39509 **SEE ALSO**39510       *isgreater()*, *isless()*, *islessequal()*, *islessgreater()*, *isunordered()*

39511       XBD &lt;math.h&gt;

39512 **CHANGE HISTORY**

39513       First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

39514 **NAME**

39515       isinf — test for infinity

39516 **SYNOPSIS**

39517       #include &lt;math.h&gt;

39518       int isinf(real-floating x);

39519 **DESCRIPTION**

39520 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 39521 conflict between the requirements described here and the ISO C standard is unintentional. This  
 39522 volume of POSIX.1-2008 defers to the ISO C standard.

39523       The *isinf()* macro shall determine whether its argument value is an infinity (positive or  
 39524 negative). First, an argument represented in a format wider than its semantic type is converted  
 39525 to its semantic type. Then determination is based on the type of the argument.

39526 **RETURN VALUE**39527       The *isinf()* macro shall return a non-zero value if and only if its argument has an infinite value.39528 **ERRORS**

39529       No errors are defined.

39530 **EXAMPLES**

39531       None.

39532 **APPLICATION USAGE**

39533       None.

39534 **RATIONALE**

39535       None.

39536 **FUTURE DIRECTIONS**

39537       None.

39538 **SEE ALSO**39539       *fpclassify()*, *isfinite()*, *isnan()*, *isnormal()*, *signbit()*

39540       XBD &lt;math.h&gt;

39541 **CHANGE HISTORY**

39542       First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

39543 **NAME**39544 isless — test if  $x$  is less than  $y$ 39545 **SYNOPSIS**

39546 #include &lt;math.h&gt;

39547 int isless(real-floating  $x$ , real-floating  $y$ );39548 **DESCRIPTION**

39549 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
39550 conflict between the requirements described here and the ISO C standard is unintentional. This  
39551 volume of POSIX.1-2008 defers to the ISO C standard.

39552 The *isless()* macro shall determine whether its first argument is less than its second argument.  
39553 The value of *isless*( $x$ ,  $y$ ) shall be equal to  $(x) < (y)$ ; however, unlike  $(x) < (y)$ , *isless*( $x$ ,  $y$ ) shall not  
39554 raise the invalid floating-point exception when  $x$  and  $y$  are unordered.

39555 **RETURN VALUE**39556 Upon successful completion, the *isless()* macro shall return the value of  $(x) < (y)$ .39557 If  $x$  or  $y$  is NaN, 0 shall be returned.39558 **ERRORS**

39559 No errors are defined.

39560 **EXAMPLES**

39561 None.

39562 **APPLICATION USAGE**

39563 The relational and equality operators support the usual mathematical relationships between  
39564 numeric values. For any ordered pair of numeric values, exactly one of the relationships (less,  
39565 greater, and equal) is true. Relational operators may raise the invalid floating-point exception  
39566 when argument values are NaNs. For a NaN and a numeric value, or for two NaNs, just the  
39567 unordered relationship is true. This macro is a quiet (non-floating-point exception raising)  
39568 version of a relational operator. It facilitates writing efficient code that accounts for NaNs  
39569 without suffering the invalid floating-point exception. In the SYNOPSIS section, **real-floating**  
39570 indicates that the argument shall be an expression of **real-floating** type.

39571 **RATIONALE**

39572 None.

39573 **FUTURE DIRECTIONS**

39574 None.

39575 **SEE ALSO**39576 *isgreater()*, *isgreaterequal()*, *islessequal()*, *islessgreater()*, *isunordered()*

39577 XBD &lt;math.h&gt;

39578 **CHANGE HISTORY**

39579 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

39580 **NAME**39581 islessequal — test if *x* is less than or equal to *y*39582 **SYNOPSIS**

39583 #include &lt;math.h&gt;

39584 int islessequal(real-floating *x*, real-floating *y*);39585 **DESCRIPTION**

39586 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
39587 conflict between the requirements described here and the ISO C standard is unintentional. This  
39588 volume of POSIX.1-2008 defers to the ISO C standard.

39589 The *islessequal()* macro shall determine whether its first argument is less than or equal to its  
39590 second argument. The value of *islessequal(x, y)* shall be equal to  $(x) \leq (y)$ ; however, unlike  
39591  $(x) \leq (y)$ , *islessequal(x, y)* shall not raise the invalid floating-point exception when *x* and *y* are  
39592 unordered.

39593 **RETURN VALUE**39594 Upon successful completion, the *islessequal()* macro shall return the value of  $(x) \leq (y)$ .39595 If *x* or *y* is NaN, 0 shall be returned.39596 **ERRORS**

39597 No errors are defined.

39598 **EXAMPLES**

39599 None.

39600 **APPLICATION USAGE**

39601 The relational and equality operators support the usual mathematical relationships between  
39602 numeric values. For any ordered pair of numeric values, exactly one of the relationships (less,  
39603 greater, and equal) is true. Relational operators may raise the invalid floating-point exception  
39604 when argument values are NaNs. For a NaN and a numeric value, or for two NaNs, just the  
39605 unordered relationship is true. This macro is a quiet (non-floating-point exception raising)  
39606 version of a relational operator. It facilitates writing efficient code that accounts for NaNs  
39607 without suffering the invalid floating-point exception. In the SYNOPSIS section, **real-floating**  
39608 indicates that the argument shall be an expression of **real-floating** type.

39609 **RATIONALE**

39610 None.

39611 **FUTURE DIRECTIONS**

39612 None.

39613 **SEE ALSO**39614 *isgreater(), isgreaterequal(), isless(), islessgreater(), isunordered()*

39615 XBD &lt;math.h&gt;

39616 **CHANGE HISTORY**

39617 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

## NAME

islessgreater — test if *x* is less than or greater than *y*

## SYNOPSIS

```
#include <math.h>
```

```
int islessgreater(real-floating x, real-floating y);
```

## DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *islessgreater()* macro shall determine whether its first argument is less than or greater than its second argument. The *islessgreater(x, y)* macro is similar to  $(x) < (y) \parallel (x) > (y)$ ; however, *islessgreater(x, y)* shall not raise the invalid floating-point exception when *x* and *y* are unordered (nor shall it evaluate *x* and *y* twice).

## RETURN VALUE

Upon successful completion, the *islessgreater()* macro shall return the value of  $(x) < (y) \parallel (x) > (y)$ .

If *x* or *y* is NaN, 0 shall be returned.

## ERRORS

No errors are defined.

## EXAMPLES

None.

## APPLICATION USAGE

The relational and equality operators support the usual mathematical relationships between numeric values. For any ordered pair of numeric values, exactly one of the relationships (less, greater, and equal) is true. Relational operators may raise the invalid floating-point exception when argument values are NaNs. For a NaN and a numeric value, or for two NaNs, just the unordered relationship is true. This macro is a quiet (non-floating-point exception raising) version of a relational operator. It facilitates writing efficient code that accounts for NaNs without suffering the invalid floating-point exception. In the SYNOPSIS section, **real-floating** indicates that the argument shall be an expression of **real-floating** type.

## RATIONALE

None.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*isgreater()*, *isgreaterequal()*, *isless()*, *islessequal()*, *isunordered()*

XBD **<math.h>**

## CHANGE HISTORY

First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.



39657 **NAME**

39658 islower, islower\_l — test for a lowercase letter

39659 **SYNOPSIS**

```
39660     #include <ctype.h>
39661     int islower(int c);
39662 CX    int islower_l(int c, locale_t locale);
```

39663 **DESCRIPTION**

39664 CX For *islower()*: The functionality described on this reference page is aligned with the ISO C  
 39665 standard. Any conflict between the requirements described here and the ISO C standard is  
 39666 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39667 CX The *islower()* and *islower\_l()* functions shall test whether *c* is a character of class **lower** in the  
 39668 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page  
 39669 135).

39670 The *c* argument is an **int**, the value of which the application shall ensure is a character  
 39671 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has  
 39672 any other value, the behavior is undefined.

39673 CX The behavior is undefined if the *locale* argument to *islower\_l()* is the special locale object  
 39674 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

39675 **RETURN VALUE**

39676 CX The *islower()* and *islower\_l()* functions shall return non-zero if *c* is a lowercase letter; otherwise,  
 39677 they shall return 0.

39678 **ERRORS**

39679 No errors are defined.

39680 **EXAMPLES**39681 **Testing for a Lowercase Letter**

39682 Two examples follow, the first using *islower()*, the second using multiple concurrent locales and  
 39683 *islower\_l()*.

39684 The examples test whether the value is a lowercase letter, based on the current locale, then use it  
 39685 as part of a key value.

```
39686     /* Example 1 -- using islower() */
39687     #include <ctype.h>
39688     #include <stdlib.h>
39689     #include <locale.h>
39690     ...
39691     char *keystr;
39692     int elementlen, len;
39693     unsigned char c;
39694     ...
39695     setlocale(LC_ALL, "");
39696     ...
39697     len = 0;
39698     while (len < elementlen) {
39699         c = (unsigned char) (rand() % 256);
39700     ...
```

```

39701         if (islower(c))
39702             keystr[len++] = c;
39703     }
39704     ...
39705     /* Example 2 -- using islower_l() */
39706     #include <ctype.h>
39707     #include <stdlib.h>
39708     #include <locale.h>
39709     ...
39710     char *keystr;
39711     int elementlen, len;
39712     unsigned char c;
39713     ...
39714     locale_t loc = newlocale (LC_ALL_MASK, "", (locale_t) 0);
39715     ...
39716     len = 0;
39717     while (len < elementlen) {
39718         c = (unsigned char) (rand() % 256);
39719         ...
39720         if (islower_l(c, loc))
39721             keystr[len++] = c;
39722     }
39723     ...

```

#### APPLICATION USAGE

To ensure applications portability, especially across natural languages, only these functions and the functions in the reference pages listed in the SEE ALSO section should be used for character classification.

#### RATIONALE

None.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *isprint()*, *ispunct()*, *isspace()*, *isupper()*, *isxdigit()*, *setlocale()*, *uselocale()*

XBD Chapter 7 (on page 135), **<ctype.h>**, **<locale.h>**

#### CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

##### Issue 6

The normative text is updated to avoid use of the term “must” for application requirements and an example is added.

##### Issue 7

The *islower\_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0292 [302], XSH/TC1-2008/0293 [283], XSH/TC1-2008/0294 [283], XSH/TC1-2008/0295 [302], and XSH/TC1-2008/0296 [304] are applied.



**NAME**

isnan — test for a NaN

**SYNOPSIS**

```
#include <math.h>
```

```
int isnan(real-floating x);
```

**DESCRIPTION**

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *isnan()* macro shall determine whether its argument value is a NaN. First, an argument represented in a format wider than its semantic type is converted to its semantic type. Then determination is based on the type of the argument.

**RETURN VALUE**

The *isnan()* macro shall return a non-zero value if and only if its argument has a NaN value.

**ERRORS**

No errors are defined.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*fpclassify()*, *isfinite()*, *isinf()*, *isnormal()*, *signbit()*

XBD **<math.h>**

**CHANGE HISTORY**

First released in Issue 3.

**Issue 5**

The DESCRIPTION is updated to indicate the return value when NaN is not supported. This text was previously published in the APPLICATION USAGE section.

**Issue 6**

Re-written for alignment with the ISO/IEC 9899:1999 standard.

39781 **NAME**

39782       isnormal — test for a normal value

39783 **SYNOPSIS**

39784       #include &lt;math.h&gt;

39785       int isnormal(real-floating x);

39786 **DESCRIPTION**

39787 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 39788 conflict between the requirements described here and the ISO C standard is unintentional. This  
 39789 volume of POSIX.1-2008 defers to the ISO C standard.

39790       The *isnormal()* macro shall determine whether its argument value is normal (neither zero,  
 39791 subnormal, infinite, nor NaN). First, an argument represented in a format wider than its  
 39792 semantic type is converted to its semantic type. Then determination is based on the type of the  
 39793 argument.

39794 **RETURN VALUE**

39795       The *isnormal()* macro shall return a non-zero value if and only if its argument has a normal  
 39796 value.

39797 **ERRORS**

39798       No errors are defined.

39799 **EXAMPLES**

39800       None.

39801 **APPLICATION USAGE**

39802       None.

39803 **RATIONALE**

39804       None.

39805 **FUTURE DIRECTIONS**

39806       None.

39807 **SEE ALSO**39808       *fpclassify()*, *isfinite()*, *isinf()*, *isnan()*, *signbit()*

39809       XBD &lt;math.h&gt;

39810 **CHANGE HISTORY**

39811       First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

39812 **NAME**

39813 isprint, isprint\_l — test for a printable character

39814 **SYNOPSIS**

```
39815 #include <ctype.h>
39816 int isprint(int c);
39817 CX int isprint_l(int c, locale_t locale);
```

39818 **DESCRIPTION**

39819 CX For *isprint()*: The functionality described on this reference page is aligned with the ISO C  
39820 standard. Any conflict between the requirements described here and the ISO C standard is  
39821 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39822 CX The *isprint()* and *isprint\_l()* functions shall test whether *c* is a character of class **print** in the  
39823 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page  
39824 135).

39825 The *c* argument is an **int**, the value of which the application shall ensure is a character  
39826 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has  
39827 any other value, the behavior is undefined.

39828 CX The behavior is undefined if the *locale* argument to *isprint\_l()* is the special locale object  
39829 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

39830 **RETURN VALUE**

39831 CX The *isprint()* and *isprint\_l()* functions shall return non-zero if *c* is a printable character;  
39832 otherwise, they shall return 0.

39833 **ERRORS**

39834 No errors are defined.

39835 **EXAMPLES**

39836 None.

39837 **APPLICATION USAGE**

39838 To ensure applications portability, especially across natural languages, only these functions and  
39839 the functions in the reference pages listed in the SEE ALSO section should be used for character  
39840 classification.

39841 **RATIONALE**

39842 None.

39843 **FUTURE DIRECTIONS**

39844 None.

39845 **SEE ALSO**

39846 *isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *ispunct()*, *isspace()*, *isupper()*,  
39847 *isxdigit()*, *setlocale()*, *uselocale()*

39848 XBD Chapter 7 (on page 135), [<ctype.h>](#), [<locale.h>](#)39849 **CHANGE HISTORY**

39850 First released in Issue 1. Derived from Issue 1 of the SVID.

39851 **Issue 6**

39852 The normative text is updated to avoid use of the term “must” for application requirements.

39853 **Issue 7**

39854 The *isprint\_l()* function is added from The Open Group Technical Standard, 2006, Extended API  
39855 Set Part 4.

39856 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0297 [302], XSH/TC1-2008/0298 [283],  
39857 and XSH/TC1-2008/0299 [283] are applied.

39858 **NAME**

39859       ispunct, ispunct\_l — test for a punctuation character

39860 **SYNOPSIS**

39861       #include &lt;ctype.h&gt;

39862       int ispunct(int c);

39863 CX      int ispunct\_l(int c, locale\_t locale);

39864 **DESCRIPTION**39865 CX      For *ispunct()*: The functionality described on this reference page is aligned with the ISO C  
39866 standard. Any conflict between the requirements described here and the ISO C standard is  
39867 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.39868 CX      The *ispunct()* and *ispunct\_l()* functions shall test whether *c* is a character of class **punct** in the  
39869 CX      current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page  
39870 135).39871      The *c* argument is an **int**, the value of which the application shall ensure is a character  
39872 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has  
39873 any other value, the behavior is undefined.39874 CX      The behavior is undefined if the *locale* argument to *ispunct\_l()* is the special locale object  
39875 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.39876 **RETURN VALUE**39877 CX      The *ispunct()* and *ispunct\_l()* functions shall return non-zero if *c* is a punctuation character;  
39878 otherwise, they shall return 0.39879 **ERRORS**

39880      No errors are defined.

39881 **EXAMPLES**

39882      None.

39883 **APPLICATION USAGE**39884      To ensure applications portability, especially across natural languages, only these functions and  
39885 the functions in the reference pages listed in the SEE ALSO section should be used for character  
39886 classification.39887 **RATIONALE**

39888      None.

39889 **FUTURE DIRECTIONS**

39890      None.

39891 **SEE ALSO**39892      *isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *isspace()*, *isupper()*,  
39893 *isxdigit()*, *setlocale()*, *uselocale()*

39894      XBD Chapter 7 (on page 135), &lt;ctype.h&gt;, &lt;locale.h&gt;

39895 **CHANGE HISTORY**

39896      First released in Issue 1. Derived from Issue 1 of the SVID.

39897 **Issue 6**

39898      The normative text is updated to avoid use of the term “must” for application requirements.



39899 **Issue 7**

39900 The *ispunct\_l()* function is added from The Open Group Technical Standard, 2006, Extended API  
39901 Set Part 4.

39902 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0300 [302], XSH/TC1-2008/0301 [283],  
39903 and XSH/TC1-2008/0302 [283] are applied.

39904 **NAME**

39905       isspace, isspace\_l — test for a white-space character

39906 **SYNOPSIS**

39907       #include &lt;ctype.h&gt;

39908       int isspace(int c);

39909 CX       int isspace\_l(int c, locale\_t locale);

39910 **DESCRIPTION**39911 CX       For *isspace()*: The functionality described on this reference page is aligned with the ISO C  
39912 standard. Any conflict between the requirements described here and the ISO C standard is  
39913 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.39914 CX       The *isspace()* and *isspace\_l()* functions shall test whether *c* is a character of class **space** in the  
39915 CX       current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page  
39916 135).39917       The *c* argument is an **int**, the value of which the application shall ensure is a character  
39918 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has  
39919 any other value, the behavior is undefined.39920 CX       The behavior is undefined if the *locale* argument to *isspace\_l()* is the special locale object  
39921 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.39922 **RETURN VALUE**39923 CX       The *isspace()* and *isspace\_l()* functions shall return non-zero if *c* is a white-space character;  
39924 otherwise, they shall return 0.39925 **ERRORS**

39926       No errors are defined.

39927 **EXAMPLES**

39928       None.

39929 **APPLICATION USAGE**39930       To ensure applications portability, especially across natural languages, only these functions and  
39931 the functions in the reference pages listed in the SEE ALSO section should be used for character  
39932 classification.39933 **RATIONALE**

39934       None.

39935 **FUTURE DIRECTIONS**

39936       None.

39937 **SEE ALSO**39938       *isalnum()*, *isalpha()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isupper()*,  
39939 *isxdigit()*, *setlocale()*, *uselocale()*

39940       XBD Chapter 7 (on page 135), &lt;ctype.h&gt;, &lt;locale.h&gt;

39941 **CHANGE HISTORY**

39942       First released in Issue 1. Derived from Issue 1 of the SVID.

39943 **Issue 6**

39944       The normative text is updated to avoid use of the term “must” for application requirements.

39945 **Issue 7**

39946

39947

The *isspace\_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

39948

39949

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0303 [302], XSH/TC1-2008/0304 [283], and XSH/TC1-2008/0305 [283] are applied.

## 39950 NAME

39951 isunordered — test if arguments are unordered

## 39952 SYNOPSIS

39953 #include <math.h>

39954 int isunordered(real-floating x, real-floating y);

## 39955 DESCRIPTION

39956 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 39957 conflict between the requirements described here and the ISO C standard is unintentional. This  
 39958 volume of POSIX.1-2008 defers to the ISO C standard.

39959 The *isunordered()* macro shall determine whether its arguments are unordered.

## 39960 RETURN VALUE

39961 Upon successful completion, the *isunordered()* macro shall return 1 if its arguments are  
 39962 unordered, and 0 otherwise.

39963 If *x* or *y* is NaN, 1 shall be returned.

## 39964 ERRORS

39965 No errors are defined.

## 39966 EXAMPLES

39967 None.

## 39968 APPLICATION USAGE

39969 The relational and equality operators support the usual mathematical relationships between  
 39970 numeric values. For any ordered pair of numeric values, exactly one of the relationships (less,  
 39971 greater, and equal) is true. Relational operators may raise the invalid floating-point exception  
 39972 when argument values are NaNs. For a NaN and a numeric value, or for two NaNs, just the  
 39973 unordered relationship is true. This macro is a quiet (non-floating-point exception raising)  
 39974 version of a relational operator. It facilitates writing efficient code that accounts for NaNs  
 39975 without suffering the invalid floating-point exception. In the SYNOPSIS section, **real-floating**  
 39976 indicates that the argument shall be an expression of **real-floating** type.

## 39977 RATIONALE

39978 None.

## 39979 FUTURE DIRECTIONS

39980 None.

## 39981 SEE ALSO

39982 *isgreater()*, *isgreaterequal()*, *isless()*, *islessequal()*, *islessgreater()*

39983 XBD <math.h>

## 39984 CHANGE HISTORY

39985 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

39986 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/50 is applied, correcting the RETURN  
 39987 VALUE section when *x* or *y* is NaN.

39988 **NAME**

39989 isupper, isupper\_l — test for an uppercase letter

39990 **SYNOPSIS**

```
39991     #include <ctype.h>
39992     int isupper(int c);
39993 CX    int isupper_l(int c, locale_t locale);
```

39994 **DESCRIPTION**

39995 CX For *isupper()*: The functionality described on this reference page is aligned with the ISO C  
 39996 standard. Any conflict between the requirements described here and the ISO C standard is  
 39997 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

39998 CX The *isupper()* and *isupper\_l()* functions shall test whether *c* is a character of class **upper** in the  
 39999 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page  
 40000 135).

40001 The *c* argument is an **int**, the value of which the application shall ensure is a character  
 40002 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has  
 40003 any other value, the behavior is undefined.

40004 CX The behavior is undefined if the *locale* argument to *isupper\_l()* is the special locale object  
 40005 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

40006 **RETURN VALUE**

40007 CX The *isupper()* and *isupper\_l()* functions shall return non-zero if *c* is an uppercase letter;  
 40008 otherwise, they shall return 0.

40009 **ERRORS**

40010 No errors are defined.

40011 **EXAMPLES**

40012 None.

40013 **APPLICATION USAGE**

40014 To ensure applications portability, especially across natural languages, only these functions and  
 40015 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 40016 classification.

40017 **RATIONALE**

40018 None.

40019 **FUTURE DIRECTIONS**

40020 None.

40021 **SEE ALSO**

40022 *isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*,  
 40023 *isxdigit()*, *setlocale()*, *uselocale()*

40024 XBD Chapter 7 (on page 135), *<ctype.h>*, *<locale.h>*40025 **CHANGE HISTORY**

40026 First released in Issue 1. Derived from Issue 1 of the SVID.

40027 **Issue 6**

40028 The normative text is updated to avoid use of the term “must” for application requirements.

40029 **Issue 7**

40030 The *isupper\_l()* function is added from The Open Group Technical Standard, 2006, Extended API  
40031 Set Part 4.

40032 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0306 [302], XSH/TC1-2008/0307 [283],  
40033 and XSH/TC1-2008/0308 [283] are applied.

40034 **NAME**

40035       iswalnum, iswalnum\_l — test for an alphanumeric wide-character code

40036 **SYNOPSIS**

40037       #include &lt;wctype.h&gt;

40038       int iswalnum(wint\_t wc);

40039 CX     int iswalnum\_l(wint\_t wc, locale\_t locale);

40040 **DESCRIPTION**40041 CX     For *iswalnum()*: The functionality described on this reference page is aligned with the ISO C  
40042 standard. Any conflict between the requirements described here and the ISO C standard is  
40043 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.40044 CX     The *iswalnum()* and *iswalnum\_l()* functions shall test whether *wc* is a wide-character code  
40045 CX     representing a character of class **alpha** or **digit** in the current locale, or in the locale represented  
40046 by *locale*, respectively; see XBD Chapter 7 (on page 135).40047       The *wc* argument is a **wint\_t**, the value of which the application shall ensure is a wide-character  
40048 code corresponding to a valid character in the locale used by the function, or equal to the value  
40049 of the macro WEOF. If the argument has any other value, the behavior is undefined.40050 CX     The behavior is undefined if the *locale* argument to *iswalnum\_l()* is the special locale object  
40051 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.40052 **RETURN VALUE**40053 CX     The *iswalnum()* and *iswalnum\_l()* functions shall return non-zero if *wc* is an alphanumeric  
40054 wide-character code; otherwise, they shall return 0.40055 **ERRORS**

40056       No errors are defined.

40057 **EXAMPLES**

40058       None.

40059 **APPLICATION USAGE**40060       To ensure applications portability, especially across natural languages, only these functions and  
40061 the functions in the reference pages listed in the SEE ALSO section should be used for character  
40062 classification.40063 **RATIONALE**

40064       None.

40065 **FUTURE DIRECTIONS**

40066       None.

40067 **SEE ALSO**40068       *iswalpha()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*,  
40069 *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

40070       XBD Chapter 7 (on page 135), &lt;locale.h&gt;, &lt;stdio.h&gt;, &lt;wctype.h&gt;

40071 **CHANGE HISTORY**

40072       First released as a World-wide Portability Interface in Issue 4.

40073 **Issue 5**40074       The following change has been made in this version for alignment with  
40075 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40076 • The SYNOPSIS has been changed to indicate that this function and associated data types  
40077 are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.

40078 **Issue 6**

40079 The normative text is updated to avoid use of the term “must” for application requirements.

40080 **Issue 7**

40081 The `iswalnum_l()` function is added from The Open Group Technical Standard, 2006, Extended  
40082 API Set Part 4.

40083 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0309 [302], XSH/TC1-2008/0310 [283],  
40084 and XSH/TC1-2008/0311 [283] are applied.



40085 **NAME**

40086 iswalpha, iswalpha\_l — test for an alphabetic wide-character code

40087 **SYNOPSIS**

40088 #include &lt;wctype.h&gt;

40089 int iswalpha(wint\_t wc);

40090 CX int iswalpha\_l(wint\_t wc, locale\_t locale);

40091 **DESCRIPTION**

40092 CX For *iswalpha()*: The functionality described on this reference page is aligned with the ISO C  
 40093 standard. Any conflict between the requirements described here and the ISO C standard is  
 40094 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40095 CX The *iswalpha()* and *iswalpha\_l()* functions shall test whether *wc* is a wide-character code  
 40096 CX representing a character of class **alpha** in the current locale, or in the locale represented by *locale*,  
 40097 respectively; see XBD Chapter 7 (on page 135).

40098 The *wc* argument is a **wint\_t**, the value of which the application shall ensure is a wide-character  
 40099 code corresponding to a valid character in the locale used by the function, or equal to the value  
 40100 of the macro WEOF. If the argument has any other value, the behavior is undefined.

40101 CX The behavior is undefined if the *locale* argument to *iswalpha\_l()* is the special locale object  
 40102 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

40103 **RETURN VALUE**

40104 CX The *iswalpha()* and *iswalpha\_l()* functions shall return non-zero if *wc* is an alphabetic wide-  
 40105 character code; otherwise, they shall return 0.

40106 **ERRORS**

40107 No errors are defined.

40108 **EXAMPLES**

40109 None.

40110 **APPLICATION USAGE**

40111 To ensure applications portability, especially across natural languages, only these functions and  
 40112 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 40113 classification.

40114 **RATIONALE**

40115 None.

40116 **FUTURE DIRECTIONS**

40117 None.

40118 **SEE ALSO**

40119 *iswalnum()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*,  
 40120 *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

40121 XBD Chapter 7 (on page 135), &lt;locale.h&gt;, &lt;wctype.h&gt;

40122 **CHANGE HISTORY**

40123 First released in Issue 4.

40124 **Issue 5**

40125 The following change has been made in this version for alignment with  
 40126 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40127                   • The SYNOPSIS has been changed to indicate that this function and associated data types  
40128                   are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.
- 40129 **Issue 6**
- 40130                   The normative text is updated to avoid use of the term “must” for application requirements.
- 40131 **Issue 7**
- 40132                   The *iswalpha\_l()* function is added from The Open Group Technical Standard, 2006, Extended  
40133                   API Set Part 4.
- 40134                   POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0312 [302], XSH/TC1-2008/0313 [283],  
40135                   and XSH/TC1-2008/0314 [283] are applied.

40136 **NAME**

40137 iswblank, iswblank\_l — test for a blank wide-character code

40138 **SYNOPSIS**

40139 #include &lt;wctype.h&gt;

40140 int iswblank(wint\_t wc);

40141 CX int iswblank\_l(wint\_t wc, locale\_t locale);

40142 **DESCRIPTION**

40143 CX For *iswblank()*: The functionality described on this reference page is aligned with the ISO C  
 40144 standard. Any conflict between the requirements described here and the ISO C standard is  
 40145 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40146 CX The *iswblank()* and *iswblank\_l()* functions shall test whether *wc* is a wide-character code  
 40147 CX representing a character of class **blank** in the current locale, or in the locale represented by  
 40148 *locale*, respectively; see XBD Chapter 7 (on page 135).

40149 The *wc* argument is a **wint\_t**, the value of which the application shall ensure is a wide-character  
 40150 code corresponding to a valid character in the locale used by the function, or equal to the value  
 40151 of the macro WEOF. If the argument has any other value, the behavior is undefined.

40152 CX The behavior is undefined if the *locale* argument to *iswblank\_l()* is the special locale object  
 40153 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

40154 **RETURN VALUE**

40155 CX The *iswblank()* and *iswblank\_l()* functions shall return non-zero if *wc* is a blank wide-character  
 40156 code; otherwise, they shall return 0.

40157 **ERRORS**

40158 No errors are defined.

40159 **EXAMPLES**

40160 None.

40161 **APPLICATION USAGE**

40162 To ensure applications portability, especially across natural languages, only these functions and  
 40163 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 40164 classification.

40165 **RATIONALE**

40166 None.

40167 **FUTURE DIRECTIONS**

40168 None.

40169 **SEE ALSO**

40170 *iswalnum()*, *iswalpha()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*,  
 40171 *iswpunct()*, *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

40172 XBD Chapter 7 (on page 135), &lt;locale.h&gt;, &lt;wctype.h&gt;

40173 **CHANGE HISTORY**

40174 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

40175 **Issue 7**

40176 The *iswblank\_l()* function is added from The Open Group Technical Standard, 2006, Extended  
 40177 API Set Part 4.

40178  
40179

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0315 [302], XSH/TC1-2008/0316 [283], and XSH/TC1-2008/0317 [283] are applied.

40180 **NAME**

40181 iswcntrl, iswcntrl\_l — test for a control wide-character code

40182 **SYNOPSIS**

40183 #include &lt;wctype.h&gt;

40184 int iswcntrl(wint\_t wc);

40185 CX int iswcntrl\_l(wint\_t wc, locale\_t locale);

40186 **DESCRIPTION**

40187 CX For *iswcntrl()*: The functionality described on this reference page is aligned with the ISO C  
 40188 standard. Any conflict between the requirements described here and the ISO C standard is  
 40189 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40190 CX The *iswcntrl()* and *iswcntrl\_l()* functions shall test whether *wc* is a wide-character code  
 40191 CX representing a character of class **cntrl** in the current locale, or in the locale represented by *locale*,  
 40192 respectively; see XBD Chapter 7 (on page 135).

40193 The *wc* argument is a **wint\_t**, the value of which the application shall ensure is a wide-character  
 40194 code corresponding to a valid character in the locale used by the function, or equal to the value  
 40195 of the macro WEOF. If the argument has any other value, the behavior is undefined.

40196 CX The behavior is undefined if the *locale* argument to *iswcntrl\_l()* is the special locale object  
 40197 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

40198 **RETURN VALUE**

40199 CX The *iswcntrl()* and *iswcntrl\_l()* functions shall return non-zero if *wc* is a control wide-character  
 40200 code; otherwise, they shall return 0.

40201 **ERRORS**

40202 No errors are defined.

40203 **EXAMPLES**

40204 None.

40205 **APPLICATION USAGE**

40206 To ensure applications portability, especially across natural languages, only these functions and  
 40207 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 40208 classification.

40209 **RATIONALE**

40210 None.

40211 **FUTURE DIRECTIONS**

40212 None.

40213 **SEE ALSO**

40214 *iswalnum()*, *iswalpha()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*,  
 40215 *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

40216 XBD Chapter 7 (on page 135), &lt;locale.h&gt;, &lt;wctype.h&gt;

40217 **CHANGE HISTORY**

40218 First released in Issue 4.

40219 **Issue 5**

40220 The following change has been made in this version for alignment with  
 40221 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40222                   • The SYNOPSIS has been changed to indicate that this function and associated data types  
40223                   are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.
- 40224 **Issue 6**
- 40225                   The normative text is updated to avoid use of the term “must” for application requirements.
- 40226 **Issue 7**
- 40227                   The `iswcntrl_l()` function is added from The Open Group Technical Standard, 2006, Extended  
40228                   API Set Part 4.
- 40229                   POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0318 [302], XSH/TC1-2008/0319 [283],  
40230                   and XSH/TC1-2008/0320 [283] are applied.

**NAME**

iswctype, iswctype\_l — test character for a specified class

**SYNOPSIS**

```
#include <wctype.h>

int iswctype(wint_t wc, wctype_t charclass);
int iswctype_l(wint_t wc, wctype_t charclass,
               locale_t locale);
```

**DESCRIPTION**

For *iswctype()*: The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *iswctype()* and *iswctype\_l()* functions shall determine whether the wide-character code *wc* has the character class *charclass*, returning true or false. The *iswctype()* and *iswctype\_l()* functions are defined on WEOF and wide-character codes corresponding to the valid character encodings in the current locale, or in the locale represented by *locale*, respectively. If the *wc* argument is not in the domain of the function, the result is undefined. If the value of *charclass* is invalid (that is, not obtained by a call to *wctype()* or *charclass* is invalidated by a subsequent call to *setlocale()* that has affected category *LC\_CTYPE*) the result is unspecified.

The behavior is undefined if the *locale* argument to *iswctype\_l()* is the special locale object *LC\_GLOBAL\_LOCALE* or is not a valid locale object handle.

**RETURN VALUE**

The *iswctype()* and *iswctype\_l()* functions shall return non-zero (true) if and only if *wc* has the property described by *charclass*. If *charclass* is (**wctype\_t**)0, these functions shall return 0.

**ERRORS**

No errors are defined.

**EXAMPLES****Testing for a Valid Character**

```
#include <wctype.h>
...
int yes_or_no;
wint_t wc;
wctype_t valid_class;
...
if ((valid_class=wctype("vowel")) == (wctype_t)0)
    /* Invalid character class. */
yes_or_no=iswctype(wc,valid_class);
```

**APPLICATION USAGE**

The twelve strings "alnum", "alpha", "blank", "cntrl", "digit", "graph", "lower", "print", "punct", "space", "upper", and "xdigit" are reserved for the standard character classes. In the table below, the functions in the left column are equivalent to the functions in the right column.

|                                         |                                                          |
|-----------------------------------------|----------------------------------------------------------|
| iswalnum( <i>wc</i> )                   | iswctype( <i>wc</i> , wctype("alnum"))                   |
| iswalnum_l( <i>wc</i> , <i>locale</i> ) | iswctype_l( <i>wc</i> , wctype("alnum"), <i>locale</i> ) |
| iswalpha( <i>wc</i> )                   | iswctype( <i>wc</i> , wctype("alpha"))                   |
| iswalpha_l( <i>wc</i> , <i>locale</i> ) | iswctype_l( <i>wc</i> , wctype("alpha"), <i>locale</i> ) |

```

40276      iswblank(wc)           iswctype(wc, wctype("blank"))
40277      iswblank_l(wc, locale) iswctype_l(wc, wctype("blank"), locale)
40278      iswcntrl(wc)           iswctype(wc, wctype("cntrl"))
40279      iswcntrl_l(wc, locale) iswctype_l(wc, wctype("cntrl"), locale)
40280      iswdigit(wc)           iswctype(wc, wctype("digit"))
40281      iswdigit_l(wc, locale) iswctype_l(wc, wctype("digit"), locale)
40282      iswgraph(wc)           iswctype(wc, wctype("graph"))
40283      iswgraph_l(wc, locale) iswctype_l(wc, wctype("graph"), locale)
40284      iswlower(wc)           iswctype(wc, wctype("lower"))
40285      iswlower_l(wc, locale) iswctype_l(wc, wctype("lower"), locale)
40286      iswprint(wc)           iswctype(wc, wctype("print"))
40287      iswprint_l(wc, locale) iswctype_l(wc, wctype("print"), locale)
40288      iswpunct(wc)           iswctype(wc, wctype("punct"))
40289      iswpunct_l(wc, locale) iswctype_l(wc, wctype("punct"), locale)
40290      iswspace(wc)           iswctype(wc, wctype("space"))
40291      iswspace_l(wc, locale) iswctype_l(wc, wctype("space"), locale)
40292      iswupper(wc)           iswctype(wc, wctype("upper"))
40293      iswupper_l(wc, locale) iswctype_l(wc, wctype("upper"), locale)
40294      iswxdigit(wc)          iswctype(wc, wctype("xdigit"))
40295      iswxdigit_l(wc, locale) iswctype_l(wc, wctype("xdigit"), locale)

```

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*iswalnum()*, *iswalpha()*, *iswcntrl()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*, *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*, *wctype()*

XBD **<locale.h>**, **<wctype.h>****CHANGE HISTORY**

First released as World-wide Portability Interfaces in Issue 4.

**Issue 5**

The following change has been made in this version for alignment with ISO/IEC 9899:1990/Amendment 1:1995 (E):

- The SYNOPSIS has been changed to indicate that this function and associated data types are now made visible by inclusion of the **<wctype.h>** header rather than **<wchar.h>**.

**Issue 6**The behavior of *charclass* = (**wctype\_t**)0 is now described.

An example is added.

A new function, *iswblank()*, is added to the list in the APPLICATION USAGE.**Issue 7**

The *iswctype\_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0321 [283] and XSH/TC1-2008/0322 [283] are applied.



40320 **NAME**

40321 iswdigit, iswdigit\_l — test for a decimal digit wide-character code

40322 **SYNOPSIS**

40323 #include &lt;wctype.h&gt;

40324 int iswdigit(wint\_t wc);

40325 CX int iswdigit\_l(wint\_t wc, locale\_t locale);

40326 **DESCRIPTION**

40327 CX For *iswdigit()*: The functionality described on this reference page is aligned with the ISO C  
 40328 standard. Any conflict between the requirements described here and the ISO C standard is  
 40329 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40330 CX The *iswdigit()* and *iswdigit\_l()* functions shall test whether *wc* is a wide-character code  
 40331 CX representing a character of class **digit** in the current locale, or in the locale represented by *locale*,  
 40332 respectively; see XBD Chapter 7 (on page 135).

40333 The *wc* argument is a **wint\_t**, the value of which the application shall ensure is a wide-character  
 40334 code corresponding to a valid character in the locale used by the function, or equal to the value  
 40335 of the macro WEOF. If the argument has any other value, the behavior is undefined.

40336 CX The behavior is undefined if the *locale* argument to *iswdigit\_l()* is the special locale object  
 40337 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

40338 **RETURN VALUE**

40339 CX The *iswdigit()* and *iswdigit\_l()* functions shall return non-zero if *wc* is a decimal digit wide-  
 40340 character code; otherwise, they shall return 0.

40341 **ERRORS**

40342 No errors are defined.

40343 **EXAMPLES**

40344 None.

40345 **APPLICATION USAGE**

40346 To ensure applications portability, especially across natural languages, only these functions and  
 40347 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 40348 classification.

40349 **RATIONALE**

40350 None.

40351 **FUTURE DIRECTIONS**

40352 None.

40353 **SEE ALSO**

40354 *iswalnum()*, *iswalpha()*, *iswcntrl()*, *iswctype()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*,  
 40355 *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

40356 XBD Chapter 7 (on page 135), &lt;locale.h&gt;, &lt;wctype.h&gt;

40357 **CHANGE HISTORY**

40358 First released in Issue 4.

40359 **Issue 5**

40360 The following change has been made in this version for alignment with  
 40361 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40362 • The SYNOPSIS has been changed to indicate that this function and associated data types  
40363 are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.

40364 **Issue 6**

40365 The normative text is updated to avoid use of the term “must” for application requirements.

40366 **Issue 7**

40367 The `iswdigit_l()` function is added from The Open Group Technical Standard, 2006, Extended  
40368 API Set Part 4.

40369 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0323 [302], XSH/TC1-2008/0324 [283],  
40370 and XSH/TC1-2008/0325 [283] are applied.

40371 **NAME**

40372 iswgraph, iswgraph\_l — test for a visible wide-character code

40373 **SYNOPSIS**

40374 #include &lt;wctype.h&gt;

40375 int iswgraph(wint\_t wc);

40376 CX int iswgraph\_l(wint\_t wc, locale\_t locale);

40377 **DESCRIPTION**

40378 CX For *iswgraph()*: The functionality described on this reference page is aligned with the ISO C  
 40379 standard. Any conflict between the requirements described here and the ISO C standard is  
 40380 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40381 CX The *iswgraph()* and *iswgraph\_l()* functions shall test whether *wc* is a wide-character code  
 40382 CX representing a character of class **graph** in the current locale, or in the locale represented by  
 40383 *locale*, respectively; see XBD Chapter 7 (on page 135).

40384 The *wc* argument is a **wint\_t**, the value of which the application shall ensure is a wide-character  
 40385 code corresponding to a valid character in the locale used by the function, or equal to the value  
 40386 of the macro WEOF. If the argument has any other value, the behavior is undefined.

40387 CX The behavior is undefined if the *locale* argument to *iswgraph\_l()* is the special locale object  
 40388 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

40389 **RETURN VALUE**

40390 CX The *iswgraph()* and *iswgraph\_l()* functions shall return non-zero if *wc* is a wide-character code  
 40391 with a visible representation; otherwise, they shall return 0.

40392 **ERRORS**

40393 No errors are defined.

40394 **EXAMPLES**

40395 None.

40396 **APPLICATION USAGE**

40397 To ensure applications portability, especially across natural languages, only these functions and  
 40398 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 40399 classification.

40400 **RATIONALE**

40401 None.

40402 **FUTURE DIRECTIONS**

40403 None.

40404 **SEE ALSO**

40405 *iswalnum()*, *iswalphabeta()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswlower()*, *iswprint()*, *iswpunct()*,  
 40406 *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

40407 XBD Chapter 7 (on page 135), &lt;locale.h&gt;, &lt;wctype.h&gt;

40408 **CHANGE HISTORY**

40409 First released in Issue 4.

40410 **Issue 5**

40411 The following change has been made in this version for alignment with  
 40412 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40413 • The SYNOPSIS has been changed to indicate that this function and associated data types  
40414 are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.

40415 **Issue 6**

40416 The normative text is updated to avoid use of the term “must” for application requirements.

40417 **Issue 7**

40418 The `iswgraph_l()` function is added from The Open Group Technical Standard, 2006, Extended  
40419 API Set Part 4.

40420 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0326 [302], XSH/TC1-2008/0327 [283],  
40421 and XSH/TC1-2008/0328 [283] are applied.

40422 **NAME**

40423 iswlower, iswlower\_l — test for a lowercase letter wide-character code

40424 **SYNOPSIS**

40425 #include &lt;wctype.h&gt;

40426 int iswlower(wint\_t wc);

40427 CX int iswlower\_l(wint\_t wc, locale\_t locale);

40428 **DESCRIPTION**

40429 CX For *iswlower()*: The functionality described on this reference page is aligned with the ISO C  
 40430 standard. Any conflict between the requirements described here and the ISO C standard is  
 40431 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40432 CX The *iswlower()* and *iswlower\_l()* functions shall test whether *wc* is a wide-character code  
 40433 CX representing a character of class **lower** in the current locale, or in the locale represented by  
 40434 *locale*, respectively; see XBD Chapter 7 (on page 135).

40435 The *wc* argument is a **wint\_t**, the value of which the application shall ensure is a wide-character  
 40436 code corresponding to a valid character in the locale used by the function, or equal to the value  
 40437 of the macro WEOF. If the argument has any other value, the behavior is undefined.

40438 CX The behavior is undefined if the *locale* argument to *iswlower\_l()* is the special locale object  
 40439 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

40440 **RETURN VALUE**

40441 CX The *iswlower()* and *iswlower\_l()* functions shall return non-zero if *wc* is a lowercase letter wide-  
 40442 character code; otherwise, they shall return 0.

40443 **ERRORS**

40444 No errors are defined.

40445 **EXAMPLES**

40446 None.

40447 **APPLICATION USAGE**

40448 To ensure applications portability, especially across natural languages, only these functions and  
 40449 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 40450 classification.

40451 **RATIONALE**

40452 None.

40453 **FUTURE DIRECTIONS**

40454 None.

40455 **SEE ALSO**

40456 *iswalnum()*, *iswalphabet()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswprint()*, *iswpunct()*,  
 40457 *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()* (on page 2196) 1

40458 XBD Chapter 7 (on page 135), &lt;locale.h&gt;, &lt;wctype.h&gt;

40459 **CHANGE HISTORY**

40460 First released in Issue 4.

40461 **Issue 5**

40462 The following change has been made in this version for alignment with  
 40463 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40464
- 40465
- The SYNOPSIS has been changed to indicate that this function and associated data types are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.

40466 **Issue 6**

40467 The normative text is updated to avoid use of the term “must” for application requirements.

40468 **Issue 7**

40469 The `iswlower_l()` function is added from The Open Group Technical Standard, 2006, Extended  
40470 API Set Part 4.

40471 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0329 [302], XSH/TC1-2008/0330 [283],  
40472 and XSH/TC1-2008/0331 [283] are applied.

40473 **NAME**

40474 iswprint, iswprint\_l — test for a printable wide-character code

40475 **SYNOPSIS**

40476 #include &lt;wctype.h&gt;

40477 int iswprint(wint\_t wc);

40478 CX int iswprint\_l(wint\_t wc, locale\_t locale);

40479 **DESCRIPTION**

40480 CX For *iswprint()*: The functionality described on this reference page is aligned with the ISO C  
 40481 standard. Any conflict between the requirements described here and the ISO C standard is  
 40482 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40483 CX The *iswprint()* and *iswprint\_l()* functions shall test whether *wc* is a wide-character code  
 40484 CX representing a character of class **print** in the current locale, or in the locale represented by *locale*,  
 40485 respectively; see XBD Chapter 7 (on page 135).

40486 The *wc* argument is a **wint\_t**, the value of which the application shall ensure is a wide-character  
 40487 code corresponding to a valid character in the locale used by the function, or equal to the value  
 40488 of the macro WEOF. If the argument has any other value, the behavior is undefined.

40489 CX The behavior is undefined if the *locale* argument to *iswprint\_l()* is the special locale object  
 40490 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

40491 **RETURN VALUE**

40492 CX The *iswprint()* and *iswprint\_l()* functions shall return non-zero if *wc* is a printable wide-  
 40493 character code; otherwise, they shall return 0.

40494 **ERRORS**

40495 No errors are defined.

40496 **EXAMPLES**

40497 None.

40498 **APPLICATION USAGE**

40499 To ensure applications portability, especially across natural languages, only these functions and  
 40500 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 40501 classification.

40502 **RATIONALE**

40503 None.

40504 **FUTURE DIRECTIONS**

40505 None.

40506 **SEE ALSO**

40507 *iswalnum()*, *iswalpha()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswpunct()*,  
 40508 *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

40509 XBD Chapter 7 (on page 135), &lt;locale.h&gt;, &lt;wctype.h&gt;

40510 **CHANGE HISTORY**

40511 First released in Issue 4.

40512 **Issue 5**

40513 The following change has been made in this version for alignment with  
 40514 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40515 • The SYNOPSIS has been changed to indicate that this function and associated data types  
40516 are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.

40517 **Issue 6**

40518 The normative text is updated to avoid use of the term “must” for application requirements.

40519 **Issue 7**

40520 The `iswprint_l()` function is added from The Open Group Technical Standard, 2006, Extended  
40521 API Set Part 4.

40522 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0332 [302], XSH/TC1-2008/0333 [283],  
40523 and XSH/TC1-2008/0334 [283] are applied.



40524 **NAME**

40525 iswpunct, iswpunct\_l — test for a punctuation wide-character code

40526 **SYNOPSIS**

40527 #include &lt;wctype.h&gt;

40528 int iswpunct(wint\_t wc);

40529 CX int iswpunct\_l(wint\_t wc, locale\_t locale);

40530 **DESCRIPTION**

40531 CX For *iswpunct()*: The functionality described on this reference page is aligned with the ISO C  
 40532 standard. Any conflict between the requirements described here and the ISO C standard is  
 40533 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40534 CX The *iswpunct()* and *iswpunct\_l()* functions shall test whether *wc* is a wide-character code  
 40535 CX representing a character of class **punct** in the current locale, or in the locale represented by  
 40536 *locale*, respectively; see XBD Chapter 7 (on page 135).

40537 The *wc* argument is a **wint\_t**, the value of which the application shall ensure is a wide-character  
 40538 code corresponding to a valid character in the locale used by the function, or equal to the value  
 40539 of the macro WEOF. If the argument has any other value, the behavior is undefined.

40540 CX The behavior is undefined if the *locale* argument to *iswpunct\_l()* is the special locale object  
 40541 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

40542 **RETURN VALUE**

40543 CX The *iswpunct()* and *iswpunct\_l()* functions shall return non-zero if *wc* is a punctuation wide-  
 40544 character code; otherwise, they shall return 0.

40545 **ERRORS**

40546 No errors are defined.

40547 **EXAMPLES**

40548 None.

40549 **APPLICATION USAGE**

40550 To ensure applications portability, especially across natural languages, only these functions and  
 40551 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 40552 classification.

40553 **RATIONALE**

40554 None.

40555 **FUTURE DIRECTIONS**

40556 None.

40557 **SEE ALSO**

40558 *iswalnum()*, *iswalphabeta()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*,  
 40559 *iswspace()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

40560 XBD Chapter 7 (on page 135), &lt;locale.h&gt;, &lt;wctype.h&gt;

40561 **CHANGE HISTORY**

40562 First released in Issue 4.

40563 **Issue 5**

40564 The following change has been made in this version for alignment with  
 40565 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40566 • The SYNOPSIS has been changed to indicate that this function and associated data types  
40567 are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.

40568 **Issue 6**

40569 The normative text is updated to avoid use of the term “must” for application requirements.

40570 **Issue 7**

40571 The `iswpunct_l()` function is added from The Open Group Technical Standard, 2006, Extended  
40572 API Set Part 4.

40573 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0335 [302], XSH/TC1-2008/0336 [283],  
40574 and XSH/TC1-2008/0337 [283] are applied.

40575 **NAME**

40576 iswspace, iswspace\_l — test for a white-space wide-character code

40577 **SYNOPSIS**

40578 #include &lt;wctype.h&gt;

40579 int iswspace(wint\_t wc);

40580 CX int iswspace\_l(wint\_t wc, locale\_t locale);

40581 **DESCRIPTION**

40582 CX For *iswspace()*: The functionality described on this reference page is aligned with the ISO C  
 40583 standard. Any conflict between the requirements described here and the ISO C standard is  
 40584 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40585 CX The *iswspace()* and *iswspace\_l()* functions shall test whether *wc* is a wide-character code  
 40586 CX representing a character of class **space** in the current locale, or in the locale represented by *locale*,  
 40587 respectively; see XBD Chapter 7 (on page 135).

40588 The *wc* argument is a **wint\_t**, the value of which the application shall ensure is a wide-character  
 40589 code corresponding to a valid character in the locale used by the function, or equal to the value  
 40590 of the macro WEOF. If the argument has any other value, the behavior is undefined.

40591 CX The behavior is undefined if the *locale* argument to *iswspace\_l()* is the special locale object  
 40592 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

40593 **RETURN VALUE**

40594 CX The *iswspace()* and *iswspace\_l()* functions shall return non-zero if *wc* is a white-space wide-  
 40595 character code; otherwise, they shall return 0.

40596 **ERRORS**

40597 No errors are defined.

40598 **EXAMPLES**

40599 None.

40600 **APPLICATION USAGE**

40601 To ensure applications portability, especially across natural languages, only these functions and  
 40602 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 40603 classification.

40604 **RATIONALE**

40605 None.

40606 **FUTURE DIRECTIONS**

40607 None.

40608 **SEE ALSO**

40609 *iswalnum()*, *iswalphabeta()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*,  
 40610 *iswpunct()*, *iswupper()*, *iswxdigit()*, *setlocale()*, *uselocale()*

40611 XBD Chapter 7 (on page 135), &lt;locale.h&gt;, &lt;wctype.h&gt;

40612 **CHANGE HISTORY**

40613 First released in Issue 4.

40614 **Issue 5**

40615 The following change has been made in this version for alignment with  
 40616 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40617                   • The SYNOPSIS has been changed to indicate that this function and associated data types  
40618                   are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.

40619 **Issue 6**

40620                   The normative text is updated to avoid use of the term “must” for application requirements.

40621 **Issue 7**

40622                   The `iswspace_l()` function is added from The Open Group Technical Standard, 2006, Extended  
40623                   API Set Part 4.

40624                   POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0338 [302], XSH/TC1-2008/0339 [283],  
40625                   and XSH/TC1-2008/0340 [283] are applied.

40626 **NAME**

40627 iswupper, iswupper\_l — test for an uppercase letter wide-character code

40628 **SYNOPSIS**

40629 #include &lt;wctype.h&gt;

40630 int iswupper(wint\_t wc);

40631 CX int iswupper\_l(wint\_t wc, locale\_t locale);

40632 **DESCRIPTION**

40633 CX For *iswupper()*: The functionality described on this reference page is aligned with the ISO C  
 40634 standard. Any conflict between the requirements described here and the ISO C standard is  
 40635 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40636 CX The *iswupper()* and *iswupper\_l()* functions shall test whether *wc* is a wide-character code  
 40637 CX representing a character of class **upper** in the current locale, or in the locale represented by  
 40638 *locale*, respectively; see XBD Chapter 7 (on page 135).

40639 The *wc* argument is a **wint\_t**, the value of which the application shall ensure is a wide-character  
 40640 code corresponding to a valid character in the locale used by the function, or equal to the value  
 40641 of the macro WEOF. If the argument has any other value, the behavior is undefined.

40642 CX The behavior is undefined if the *locale* argument to *iswupper\_l()* is the special locale object  
 40643 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

40644 **RETURN VALUE**

40645 CX The *iswupper()* and *iswupper\_l()* functions shall return non-zero if *wc* is an uppercase letter  
 40646 wide-character code; otherwise, they shall return 0.

40647 **ERRORS**

40648 No errors are defined.

40649 **EXAMPLES**

40650 None.

40651 **APPLICATION USAGE**

40652 To ensure applications portability, especially across natural languages, only these functions and  
 40653 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 40654 classification.

40655 **RATIONALE**

40656 None.

40657 **FUTURE DIRECTIONS**

40658 None.

40659 **SEE ALSO**

40660 *iswalnum()*, *iswalphabeta()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*,  
 40661 *iswpunct()*, *iswspace()*, *iswxdigit()*, *setlocale()*, *uselocale()*

40662 XBD Chapter 7 (on page 135), &lt;locale.h&gt;, &lt;wctype.h&gt;

40663 **CHANGE HISTORY**

40664 First released in Issue 4.

40665 **Issue 5**

40666 The following change has been made in this version for alignment with  
 40667 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40668                   • The SYNOPSIS has been changed to indicate that this function and associated data types  
40669                   are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.

40670 **Issue 6**

40671                   The normative text is updated to avoid use of the term “must” for application requirements.

40672 **Issue 7**

40673                   The *iswupper\_l()* function is added from The Open Group Technical Standard, 2006, Extended  
40674                   API Set Part 4.

40675                   POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0341 [302], XSH/TC1-2008/0342 [283],  
40676                   and XSH/TC1-2008/0343 [283] are applied.

40677 **NAME**

40678 iswxdigit, iswxdigit\_l — test for a hexadecimal digit wide-character code

40679 **SYNOPSIS**

40680 #include &lt;wctype.h&gt;

40681 int iswxdigit(wint\_t wc);

40682 CX int iswxdigit\_l(wint\_t wc, locale\_t locale);

40683 **DESCRIPTION**

40684 CX For *iswxdigit()*: The functionality described on this reference page is aligned with the ISO C  
 40685 standard. Any conflict between the requirements described here and the ISO C standard is  
 40686 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40687 CX The *iswxdigit()* and *iswxdigit\_l()* functions shall test whether *wc* is a wide-character code  
 40688 CX representing a character of class **xdigit** in the current locale, or in the locale represented by  
 40689 *locale*, respectively; see XBD Chapter 7 (on page 135).

40690 The *wc* argument is a **wint\_t**, the value of which the application shall ensure is a wide-character  
 40691 code corresponding to a valid character in the locale used by the function, or equal to the value  
 40692 of the macro WEOF. If the argument has any other value, the behavior is undefined.

40693 CX The behavior is undefined if the *locale* argument to *iswxdigit\_l()* is the special locale object  
 40694 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

40695 **RETURN VALUE**

40696 CX The *iswxdigit()* and *iswxdigit\_l()* functions shall return non-zero if *wc* is a hexadecimal digit  
 40697 wide-character code; otherwise, they shall return 0.

40698 **ERRORS**

40699 No errors are defined.

40700 **EXAMPLES**

40701 None.

40702 **APPLICATION USAGE**

40703 To ensure applications portability, especially across natural languages, only these functions and  
 40704 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 40705 classification.

40706 **RATIONALE**

40707 None.

40708 **FUTURE DIRECTIONS**

40709 None.

40710 **SEE ALSO**

40711 *iswalnum()*, *iswalpha()*, *iswcntrl()*, *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*,  
 40712 *iswpunct()*, *iswspace()*, *iswupper()*, *setlocale()*, *uselocale()*

40713 XBD Chapter 7 (on page 135), &lt;locale.h&gt;, &lt;wctype.h&gt;

40714 **CHANGE HISTORY**

40715 First released in Issue 4.

40716 **Issue 5**

40717 The following change has been made in this version for alignment with  
 40718 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 40719                   • The SYNOPSIS has been changed to indicate that this function and associated data types  
40720                   are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.
- 40721 **Issue 6**
- 40722                   The normative text is updated to avoid use of the term “must” for application requirements.
- 40723 **Issue 7**
- 40724                   The `iswxdigit_l()` function is added from The Open Group Technical Standard, 2006, Extended  
40725                   API Set Part 4.
- 40726                   POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0344 [302], XSH/TC1-2008/0345 [283],  
40727                   and XSH/TC1-2008/0346 [283] are applied.



40728 **NAME**

40729 isxdigit, isxdigit\_l — test for a hexadecimal digit

40730 **SYNOPSIS**

```
40731     #include <ctype.h>
40732     int isxdigit(int c);
40733 CX    int isxdigit_l(int c, locale_t locale);
```

40734 **DESCRIPTION**

40735 CX For *isxdigit()*: The functionality described on this reference page is aligned with the ISO C  
 40736 standard. Any conflict between the requirements described here and the ISO C standard is  
 40737 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

40738 CX The *isxdigit()* and *isxdigit\_l()* functions shall test whether *c* is a character of class **xdigit** in the  
 40739 CX current locale, or in the locale represented by *locale*, respectively; see XBD Chapter 7 (on page  
 40740 135).

40741 The *c* argument is an **int**, the value of which the application shall ensure is a character  
 40742 representable as an **unsigned char** or equal to the value of the macro EOF. If the argument has  
 40743 any other value, the behavior is undefined.

40744 CX The behavior is undefined if the *locale* argument to *isxdigit\_l()* is the special locale object  
 40745 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

40746 **RETURN VALUE**

40747 CX The *isxdigit()* and *isxdigit\_l()* functions shall return non-zero if *c* is a hexadecimal digit;  
 40748 otherwise, they shall return 0.

40749 **ERRORS**

40750 No errors are defined.

40751 **EXAMPLES**

40752 None.

40753 **APPLICATION USAGE**

40754 To ensure applications portability, especially across natural languages, only these functions and  
 40755 the functions in the reference pages listed in the SEE ALSO section should be used for character  
 40756 classification.

40757 **RATIONALE**

40758 None.

40759 **FUTURE DIRECTIONS**

40760 None.

40761 **SEE ALSO**

40762 *isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*, *islower()*, *isprint()*, *ispunct()*, *isspace()*,  
 40763 *isupper()*

40764 XBD Chapter 7 (on page 135), **<ctype.h>**40765 **CHANGE HISTORY**

40766 First released in Issue 1. Derived from Issue 1 of the SVID.

40767 **Issue 6**

40768 The normative text is updated to avoid use of the term “must” for application requirements.

40769 **Issue 7**40770  
40771

The *isxdigit\_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

40772  
40773

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0347 [302], XSH/TC1-2008/0348 [283], and XSH/TC1-2008/0349 [283] are applied.

40774 **NAME**

40775       j0, j1, jn — Bessel functions of the first kind

40776 **SYNOPSIS**

```

40777 XSI      #include <math.h>
40778          double j0(double x);
40779          double j1(double x);
40780          double jn(int n, double x);

```

40781 **DESCRIPTION**

40782       The *j0()*, *j1()*, and *jn()* functions shall compute Bessel functions of *x* of the first kind of orders 0,  
 40783       1, and *n*, respectively.

40784       An application wishing to check for error situations should set *errno* to zero and call  
 40785       *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 40786       *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 40787       zero, an error has occurred.

40788 **RETURN VALUE**

40789       Upon successful completion, these functions shall return the relevant Bessel value of *x* of the  
 40790       first kind.

40791       If the *x* argument is too large in magnitude, or the correct result would cause underflow, 0 shall  
 40792       be returned and a range error may occur.

40793 MXX       If *x* is NaN, a NaN shall be returned.

40794 **ERRORS**

40795       These functions may fail if:

40796       Range Error       The value of *x* was too large in magnitude, or an underflow occurred.

40797       If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 40798       then *errno* shall be set to [ERANGE]. If the integer expression  
 40799       (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 40800       floating-point exception shall be raised.

40801       No other errors shall occur.

40802 **EXAMPLES**

40803       None.

40804 **APPLICATION USAGE**

40805       On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 40806       MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

40807 **RATIONALE**

40808       None.

40809 **FUTURE DIRECTIONS**

40810       None.

40811 **SEE ALSO**

40812       *feclearexcept()*, *fetestexcept()*, *isnan()*, *y0()*

40813       XBD Section 4.20 (on page 117), *<math.h>*

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

The DESCRIPTION is updated to indicate how an application should check for an error. This text was previously published in the APPLICATION USAGE section.

**Issue 6**

The may fail [EDOM] error is removed for the case for NaN.

The RETURN VALUE and ERRORS sections are reworked for alignment of the error handling with the ISO/IEC 9899:1999 standard.

**Issue 7**

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0350 [68] is applied.

40825 **NAME**

40826       jrand48 — generate a uniformly distributed pseudo-random long signed integer

40827 **SYNOPSIS**

```
40828 XSI      #include <stdlib.h>
40829          long jrand48(unsigned short xsubi[3]);
```

40830 **DESCRIPTION**40831       Refer to *drand48()*.

**NAME**

kill — send a signal to a process or a group of processes

**SYNOPSIS**

```
CX      #include <signal.h>
int kill(pid_t pid, int sig);
```

**DESCRIPTION**

The `kill()` function shall send a signal to a process or a group of processes specified by `pid`. The signal to be sent is specified by `sig` and is either one from the list given in `<signal.h>` or 0. If `sig` is 0 (the null signal), error checking is performed but no signal is actually sent. The null signal can be used to check the validity of `pid`.

For a process to have permission to send a signal to a process designated by `pid`, unless the sending process has appropriate privileges, the real or effective user ID of the sending process shall match the real or saved set-user-ID of the receiving process.

If `pid` is greater than 0, `sig` shall be sent to the process whose process ID is equal to `pid`.

If `pid` is 0, `sig` shall be sent to all processes (excluding an unspecified set of system processes) whose process group ID is equal to the process group ID of the sender, and for which the process has permission to send a signal.

If `pid` is -1, `sig` shall be sent to all processes (excluding an unspecified set of system processes) for which the process has permission to send that signal.

If `pid` is negative, but not -1, `sig` shall be sent to all processes (excluding an unspecified set of system processes) whose process group ID is equal to the absolute value of `pid`, and for which the process has permission to send a signal.

If the value of `pid` causes `sig` to be generated for the sending process, and if `sig` is not blocked for the calling thread and if no other thread has `sig` unblocked or is waiting in a `sigwait()` function for `sig`, either `sig` or at least one pending unblocked signal shall be delivered to the sending thread before `kill()` returns.

The user ID tests described above shall not be applied when sending SIGCONT to a process that is a member of the same session as the sending process.

An implementation that provides extended security controls may impose further implementation-defined restrictions on the sending of signals, including the null signal. In particular, the system may deny the existence of some or all of the processes specified by `pid`.

The `kill()` function is successful if the process has permission to send `sig` to any of the processes specified by `pid`. If `kill()` fails, no signal shall be sent.

**RETURN VALUE**

Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and `errno` set to indicate the error.

**ERRORS**

The `kill()` function shall fail if:

- |          |                                                                                        |
|----------|----------------------------------------------------------------------------------------|
| [EINVAL] | The value of the <code>sig</code> argument is an invalid or unsupported signal number. |
| [EPERM]  | The process does not have permission to send the signal to any receiving process.      |

40873 [ESRCH] No process or process group can be found corresponding to that specified by  
 40874 *pid*.

#### 40875 EXAMPLES

40876 None.

#### 40877 APPLICATION USAGE

40878 None.

#### 40879 RATIONALE

40880 The semantics for permission checking for *kill()* differed between System V and most other  
 40881 implementations, such as Version 7 or 4.3 BSD. The semantics chosen for this volume of  
 40882 POSIX.1-2008 agree with System V. Specifically, a set-user-ID process cannot protect itself  
 40883 against signals (or at least not against SIGKILL) unless it changes its real user ID. This choice  
 40884 allows the user who starts an application to send it signals even if it changes its effective user ID.  
 40885 The other semantics give more power to an application that wants to protect itself from the user  
 40886 who ran it.

40887 Some implementations provide semantic extensions to the *kill()* function when the absolute  
 40888 value of *pid* is greater than some maximum, or otherwise special, value. Negative values are a  
 40889 flag to *kill()*. Since most implementations return [ESRCH] in this case, this behavior is not  
 40890 included in this volume of POSIX.1-2008, although a conforming implementation could provide  
 40891 such an extension.

40892 The unspecified processes to which a signal cannot be sent may include the scheduler or *init*.

40893 There was initially strong sentiment to specify that, if *pid* specifies that a signal be sent to the  
 40894 calling process and that signal is not blocked, that signal would be delivered before *kill()*  
 40895 returns. This would permit a process to call *kill()* and be guaranteed that the call never return.  
 40896 However, historical implementations that provide only the *signal()* function make only the  
 40897 weaker guarantee in this volume of POSIX.1-2008, because they only deliver one signal each  
 40898 time a process enters the kernel. Modifications to such implementations to support the  
 40899 *sigaction()* function generally require entry to the kernel following return from a signal-catching  
 40900 function, in order to restore the signal mask. Such modifications have the effect of satisfying the  
 40901 stronger requirement, at least when *sigaction()* is used, but not necessarily when *signal()* is used.  
 40902 The standard developers considered making the stronger requirement except when *signal()* is  
 40903 used, but felt this would be unnecessarily complex. Implementors are encouraged to meet the  
 40904 stronger requirement whenever possible. In practice, the weaker requirement is the same, except  
 40905 in the rare case when two signals arrive during a very short window. This reasoning also applies  
 40906 to a similar requirement for *sigprocmask()*.

40907 In 4.2 BSD, the SIGCONT signal can be sent to any descendant process regardless of user-ID  
 40908 security checks. This allows a job control shell to continue a job even if processes in the job have  
 40909 altered their user IDs (as in the *su* command). In keeping with the addition of the concept of  
 40910 sessions, similar functionality is provided by allowing the SIGCONT signal to be sent to any  
 40911 process in the same session regardless of user ID security checks. This is less restrictive than BSD  
 40912 in the sense that ancestor processes (in the same session) can now be the recipient. It is more  
 40913 restrictive than BSD in the sense that descendant processes that form new sessions are now  
 40914 subject to the user ID checks. A similar relaxation of security is not necessary for the other job  
 40915 control signals since those signals are typically sent by the terminal driver in recognition of  
 40916 special characters being typed; the terminal driver bypasses all security checks.

40917 In secure implementations, a process may be restricted from sending a signal to a process having  
 40918 a different security label. In order to prevent the existence or nonexistence of a process from  
 40919 being used as a covert channel, such processes should appear nonexistent to the sender; that is,  
 40920 [ESRCH] should be returned, rather than [EPERM], if *pid* refers only to such processes.

Historical implementations varied on the result of a `kill()` with `pid` indicating a zombie process. Some indicated success on such a call (subject to permission checking), while others gave an error of [ESRCH]. Since the definition of process lifetime in this volume of POSIX.1-2008 covers zombie processes, the [ESRCH] error as described is inappropriate in this case and implementations that give this error do not conform. This means that an application cannot have a parent process check for termination of a particular child by sending it the null signal with `kill()`, but must instead use `waitpid()` or `waitid()`.

There is some belief that the name `kill()` is misleading, since the function is not always intended to cause process termination. However, the name is common to all historical implementations, and any change would be in conflict with the goal of minimal changes to existing application code.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

`getpid()`, `raise()`, `setsid()`, `sigaction()`, `sigqueue()`, `wait()`

XBD `<signal.h>`, `<sys/types.h>`

#### CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

##### Issue 5

The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

##### Issue 6

In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the second paragraph is reworded to indicate that the saved set-user-ID of the calling process is checked in place of its effective user ID. This is a FIPS requirement.
- The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The behavior when `pid` is `-1` is now specified. It was previously explicitly unspecified in the POSIX.1-1988 standard.

The normative text is updated to avoid use of the term “must” for application requirements.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/51 is applied, correcting the RATIONALE section.



40956 **NAME**

40957 killpg — send a signal to a process group

40958 **SYNOPSIS**

```
40959 XSI      #include <signal.h>
40960         int killpg(pid_t pgrp, int sig);
```

40961 **DESCRIPTION**40962 The *killpg()* function shall send the signal specified by *sig* to the process group specified by *pgrp*.

40963 If *pgrp* is greater than 1, *killpg(pgrp, sig)* shall be equivalent to *kill(-pgrp, sig)*. If *pgrp* is less than or  
 40964 equal to 1, the behavior of *killpg()* is undefined.

40965 **RETURN VALUE**40966 Refer to *kill()*.40967 **ERRORS**40968 Refer to *kill()*.40969 **EXAMPLES**40970 **Sending a Signal to All Other Members of a Process Group**

40971 The following example shows how the calling process could send a signal to all other members  
 40972 of its process group. To prevent itself from receiving the signal it first makes itself immune to the  
 40973 signal by ignoring it.

```
40974 #include <signal.h>
40975 #include <unistd.h>
40976 ...
40977     if (signal(SIGUSR1, SIG_IGN) == SIG_ERR)
40978         /* Handle error */;
40979     if (killpg(getpgrp(), SIGUSR1) == -1)
40980         /* Handle error */;
```

40981 **APPLICATION USAGE**

40982 None.

40983 **RATIONALE**

40984 None.

40985 **FUTURE DIRECTIONS**

40986 None.

40987 **SEE ALSO**40988 *getpgid()*, *getpid()*, *kill()*, *raise()*40989 XBD *<signal.h>*40990 **CHANGE HISTORY**

40991 First released in Issue 4, Version 2.

40992 **Issue 5**

40993 Moved from X/OPEN UNIX extension to BASE.

40994 **Issue 6**

40995 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/52 is applied, adding the example to the  
40996 EXAMPLES section.

40997 **NAME**

40998 l64a — convert a 32-bit integer to a radix-64 ASCII string

40999 **SYNOPSIS**

```
41000 XSI      #include <stdlib.h>  
41001      char *l64a(long value);
```

41002 **DESCRIPTION**41003 Refer to *a64l()*.

41004 **NAME**

41005 labs, llabs — return a long integer absolute value

41006 **SYNOPSIS**

41007 #include &lt;stdlib.h&gt;

41008 long labs(long i);

41009 long long llabs(long long i);

41010 **DESCRIPTION**

41011 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
41012 conflict between the requirements described here and the ISO C standard is unintentional. This  
41013 volume of POSIX.1-2008 defers to the ISO C standard.

41014 The *labs()* function shall compute the absolute value of the **long** integer operand *i*. The *llabs()*  
41015 function shall compute the absolute value of the **long long** integer operand *i*. If the result  
41016 cannot be represented, the behavior is undefined.

41017 **RETURN VALUE**41018 The *labs()* function shall return the absolute value of the **long** integer operand.41019 The *llabs()* function shall return the absolute value of the **long long** integer operand.41020 **ERRORS**

41021 No errors are defined.

41022 **EXAMPLES**

41023 None.

41024 **APPLICATION USAGE**

41025 None.

41026 **RATIONALE**

41027 None.

41028 **FUTURE DIRECTIONS**

41029 None.

41030 **SEE ALSO**41031 *abs()*

41032 XBD &lt;stdlib.h&gt;

41033 **CHANGE HISTORY**

41034 First released in Issue 4. Derived from the ISO C standard.

41035 **Issue 6**41036 The *llabs()* function is added for alignment with the ISO/IEC 9899:1999 standard.41037 **Issue 7**

41038 SD5-XSH-ERN-152 is applied, correcting the RETURN VALUE section.

41039 **NAME**

41040 lchown — change the owner and group of a symbolic link

41041 **SYNOPSIS**

41042 #include &lt;unistd.h&gt;

41043 int lchown(const char \*path, uid\_t owner, gid\_t group);

41044 **DESCRIPTION**

41045 The *lchown()* function shall be equivalent to *chown()*, except in the case where the named file is a  
 41046 symbolic link. In this case, *lchown()* shall change the ownership of the symbolic link file itself,  
 41047 while *chown()* changes the ownership of the file or directory to which the symbolic link refers.

41048 **RETURN VALUE**

41049 Upon successful completion, *lchown()* shall return 0. Otherwise, it shall return –1 and set *errno* to  
 41050 indicate an error.

41051 **ERRORS**41052 The *lchown()* function shall fail if:

- |       |                |                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 41053 | [EACCES]       | Search permission is denied on a component of the path prefix of <i>path</i> .                                                                                                                                                                                                                                                                                                                        |
| 41054 | [EINVAL]       | The owner or group ID is not a value supported by the implementation.                                                                                                                                                                                                                                                                                                                                 |
| 41055 | [ELOOP]        | A loop exists in symbolic links encountered during resolution of the <i>path</i> argument.                                                                                                                                                                                                                                                                                                            |
| 41056 |                |                                                                                                                                                                                                                                                                                                                                                                                                       |
| 41057 | [ENAMETOOLONG] |                                                                                                                                                                                                                                                                                                                                                                                                       |
| 41058 |                | The length of a component of a pathname is longer than {NAME_MAX}.                                                                                                                                                                                                                                                                                                                                    |
| 41059 | [ENOENT]       | A component of <i>path</i> does not name an existing file or <i>path</i> is an empty string.                                                                                                                                                                                                                                                                                                          |
| 41060 | [ENOTDIR]      | A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>path</i> argument contains at least one non- <i>&lt;slash&gt;</i> character and ends with one or more trailing <i>&lt;slash&gt;</i> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory. |
| 41061 |                |                                                                                                                                                                                                                                                                                                                                                                                                       |
| 41062 |                |                                                                                                                                                                                                                                                                                                                                                                                                       |
| 41063 |                |                                                                                                                                                                                                                                                                                                                                                                                                       |
| 41064 |                |                                                                                                                                                                                                                                                                                                                                                                                                       |
| 41065 | [EPERM]        | The effective user ID does not match the owner of the file and the process does not have appropriate privileges.                                                                                                                                                                                                                                                                                      |
| 41066 |                |                                                                                                                                                                                                                                                                                                                                                                                                       |
| 41067 | [EROFS]        | The file resides on a read-only file system.                                                                                                                                                                                                                                                                                                                                                          |
| 41068 |                | The <i>lchown()</i> function may fail if:                                                                                                                                                                                                                                                                                                                                                             |
| 41069 | [EIO]          | An I/O error occurred while reading or writing to the file system.                                                                                                                                                                                                                                                                                                                                    |
| 41070 | [EINTR]        | A signal was caught during execution of the function.                                                                                                                                                                                                                                                                                                                                                 |
| 41071 | [ELOOP]        | More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the <i>path</i> argument.                                                                                                                                                                                                                                                                                                |
| 41072 |                |                                                                                                                                                                                                                                                                                                                                                                                                       |
| 41073 | [ENAMETOOLONG] |                                                                                                                                                                                                                                                                                                                                                                                                       |
| 41074 |                | The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.                                                                                                                                                                                                                                         |
| 41075 |                |                                                                                                                                                                                                                                                                                                                                                                                                       |
| 41076 |                |                                                                                                                                                                                                                                                                                                                                                                                                       |

41077 **EXAMPLES**41078 **Changing the Current Owner of a File**

41079 The following example shows how to change the ownership of the symbolic link named  
41080 **/modules/pass1** to the user ID associated with “jones” and the group ID associated with “cnd”.

41081 The numeric value for the user ID is obtained by using the *getpwnam()* function. The numeric  
41082 value for the group ID is obtained by using the *getgrnam()* function.

```
41083 #include <sys/types.h>
41084 #include <unistd.h>
41085 #include <pwd.h>
41086 #include <grp.h>

41087 struct passwd *pwd;
41088 struct group *grp;
41089 char          *path = "/modules/pass1";
41090 ...
41091 pwd = getpwnam("jones");
41092 grp = getgrnam("cnd");
41093 lchown(path, pwd->pw_uid, grp->gr_gid);
```

41094 **APPLICATION USAGE**

41095 On implementations which support symbolic links as directory entries rather than files, *lchown()*  
41096 may fail.

41097 **RATIONALE**

41098 None.

41099 **FUTURE DIRECTIONS**

41100 None.

41101 **SEE ALSO**

41102 *chown()*, *symlink()*

41103 XBD **<unistd.h>**

41104 **CHANGE HISTORY**

41105 First released in Issue 4, Version 2.

41106 **Issue 5**

41107 Moved from X/OPEN UNIX extension to BASE.

41108 **Issue 6**

41109 The wording of the mandatory [ELOOP] error condition is updated, and a second optional  
41110 [ELOOP] error condition is added.

41111 The Open Group Base Resolution bwg2001-013 is applied, adding wording to the  
41112 APPLICATION USAGE.

41113 **Issue 7**

41114 Austin Group Interpretation 1003.1-2001 #143 is applied.

41115 The *lchown()* function is moved from the XSI option to the Base.

41116 The [EOPNOTSUPP] error is removed.

41117 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a  
41118 pathname exists but is not a directory or a symbolic link to a directory.

41119

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0351 [324] is applied.

**41120 NAME**

41121 lcong48 — seed a uniformly distributed pseudo-random signed long integer generator

**41122 SYNOPSIS**

```
41123 XSI #include <stdlib.h>  
41124 void lcong48(unsigned short param[7]);
```

**41125 DESCRIPTION**

41126 Refer to *drand48()*.



41127 **NAME**

41128 ldexp, ldexpf, ldexpl — load exponent of a floating-point number

41129 **SYNOPSIS**

```
41130 #include <math.h>
41131 double ldexp(double x, int exp);
41132 float ldexpf(float x, int exp);
41133 long double ldexpl(long double x, int exp);
```

41134 **DESCRIPTION**

41135 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 41136 conflict between the requirements described here and the ISO C standard is unintentional. This  
 41137 volume of POSIX.1-2008 defers to the ISO C standard.

41138 These functions shall compute the quantity  $x * 2^{exp}$ .

41139 An application wishing to check for error situations should set *errno* to zero and call  
 41140 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 41141 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 41142 zero, an error has occurred.

41143 **RETURN VALUE**

41144 Upon successful completion, these functions shall return *x* multiplied by 2, raised to the power  
 41145 *exp*.

41146 If these functions would cause overflow, a range error shall occur and *ldexp*(), *ldexpf*(), and  
 41147 *ldexpl*() shall return  $\pm\text{HUGE\_VAL}$ ,  $\pm\text{HUGE\_VALF}$ , and  $\pm\text{HUGE\_VALL}$  (according to the sign of  
 41148 *x*), respectively.

41149 MXX If the correct value would cause underflow, and is not representable, a range error may occur,  
 41150 MXX and *ldexp*(), *ldexpf*(), and *ldexpl*() shall return 0.0, or (if IEC 60559 Floating-Point is not  
 41151 supported) an implementation-defined value no greater in magnitude than DBL\_MIN,  
 41152 FLT\_MIN, and LDBL\_MIN, respectively.

41153 MX If *x* is NaN, a NaN shall be returned.

41154 If *x* is  $\pm 0$  or  $\pm\text{Inf}$ , *x* shall be returned.

41155 If *exp* is 0, *x* shall be returned.

41156 MXX If the correct value would cause underflow, and is representable, a range error may occur and  
 41157 the correct value shall be returned.

41158 **ERRORS**

41159 These functions shall fail if:

41160 Range Error The result overflows.

41161 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 41162 then *errno* shall be set to [ERANGE]. If the integer expression  
 41163 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the overflow  
 41164 floating-point exception shall be raised.

41165 These functions may fail if:

41166 Range Error The result underflows.

41167 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 41168 then *errno* shall be set to [ERANGE]. If the integer expression  
 41169 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow

41170 floating-point exception shall be raised.

41171 **EXAMPLES**

41172 None.

41173 **APPLICATION USAGE**

41174 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
41175 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

41176 **RATIONALE**

41177 None.

41178 **FUTURE DIRECTIONS**

41179 None.

41180 **SEE ALSO**

41181 *feclearexcept()*, *fetestexcept()*, *frexp()*, *isnan()*

41182 XBD Section 4.20 (on page 117), **<math.h>**

41183 **CHANGE HISTORY**

41184 First released in Issue 1. Derived from Issue 1 of the SVID.

41185 **Issue 5**

41186 The DESCRIPTION is updated to indicate how an application should check for an error. This  
41187 text was previously published in the APPLICATION USAGE section.

41188 **Issue 6**

41189 The *ldexpf()* and *ldexpl()* functions are added for alignment with the ISO/IEC 9899:1999  
41190 standard.

41191 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
41192 revised to align with the ISO/IEC 9899:1999 standard.

41193 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
41194 marked.

41195 **Issue 7**

41196 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0352 [68] and XSH/TC1-2008/0353  
41197 [68] are applied.

41198 **NAME**

41199       ldiv, lldiv — compute quotient and remainder of a long division

41200 **SYNOPSIS**

41201       #include &lt;stdlib.h&gt;

41202       ldiv\_t ldiv(long *numer*, long *denom*);41203       lldiv\_t lldiv(long long *numer*, long long *denom*);41204 **DESCRIPTION**

41205 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 41206 conflict between the requirements described here and the ISO C standard is unintentional. This  
 41207 volume of POSIX.1-2008 defers to the ISO C standard.

41208       These functions shall compute the quotient and remainder of the division of the numerator  
 41209 *numer* by the denominator *denom*. If the division is inexact, the resulting quotient is the **long**  
 41210 integer (for the *ldiv*() function) or **long long** integer (for the *lldiv*() function) of lesser magnitude  
 41211 that is the nearest to the algebraic quotient. If the result cannot be represented, the behavior is  
 41212 undefined; otherwise, *quot* \* *denom* + *rem* shall equal *numer*.

41213 **RETURN VALUE**

41214       The *ldiv*() function shall return a structure of type **ldiv\_t**, comprising both the quotient and the  
 41215 remainder. The structure shall include the following members, in any order:

```
41216       long   quot;       /* Quotient */
41217       long   rem;       /* Remainder */
```

41218       The *lldiv*() function shall return a structure of type **lldiv\_t**, comprising both the quotient and the  
 41219 remainder. The structure shall include the following members, in any order:

```
41220       long long   quot;       /* Quotient */
41221       long long   rem;       /* Remainder */
```

41222 **ERRORS**

41223       No errors are defined.

41224 **EXAMPLES**

41225       None.

41226 **APPLICATION USAGE**

41227       None.

41228 **RATIONALE**

41229       None.

41230 **FUTURE DIRECTIONS**

41231       None.

41232 **SEE ALSO**41233       

*div*()

41234       XBD &lt;stdlib.h&gt;

41235 **CHANGE HISTORY**

41236       First released in Issue 4. Derived from the ISO C standard.

41237 **Issue 6**41238       The *lldiv*() function is added for alignment with the ISO/IEC 9899:1999 standard.

41239 **NAME**

41240 lfind — find entry in a linear search table

41241 **SYNOPSIS**

```
41242 XSI #include <search.h>
41243 void *lfind(const void *key, const void *base, size_t *nelp,
41244            size_t width, int (*compar)(const void *, const void *));
```

41245 **DESCRIPTION**41246 Refer to *lsearch()*.

41247 **NAME**

41248 lgamma, lgammaf, lgammal, signgam — log gamma function

41249 **SYNOPSIS**

```
41250 #include <math.h>
41251 double lgamma(double x);
41252 float lgammaf(float x);
41253 long double lgammal(long double x);
41254 XSI extern int signgam;
```

41255 **DESCRIPTION**

41256 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 41257 conflict between the requirements described here and the ISO C standard is unintentional. This  
 41258 volume of POSIX.1-2008 defers to the ISO C standard.

41259 These functions shall compute  $\log_e |\Gamma(x)|$  where  $\Gamma(x)$  is defined as  $\int_0^{\infty} e^{-t} t^{x-1} dt$ . The argument  $x$   
 41260 need not be a non-positive integer ( $\Gamma(x)$  is defined over the reals, except the non-positive  
 41261 integers).

41262 XSI If  $x$  is NaN,  $-\text{Inf}$ , or a negative integer, the value of *signgam* is unspecified.

41263 CX These functions need not be thread-safe.

41264 An application wishing to check for error situations should set *errno* to zero and call  
 41265 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 41266 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 41267 zero, an error has occurred.

41268 **RETURN VALUE**

41269 Upon successful completion, these functions shall return the logarithmic gamma of  $x$ .

41270 If  $x$  is a non-positive integer, a pole error shall occur and *lgamma*(), *lgammaf*(), and *lgammal*()  
 41271 shall return +HUGE\_VAL, +HUGE\_VALF, and +HUGE\_VALL, respectively.

41272 If the correct value would cause overflow, a range error shall occur and *lgamma*(), *lgammaf*(),  
 41273 and *lgammal*() shall return  $\pm\text{HUGE\_VAL}$ ,  $\pm\text{HUGE\_VALF}$ , and  $\pm\text{HUGE\_VALL}$  (having the same  
 41274 sign as the correct value), respectively.

41275 MX If  $x$  is NaN, a NaN shall be returned.

41276 If  $x$  is 1 or 2, +0 shall be returned.

41277 If  $x$  is  $\pm\text{Inf}$ , +Inf shall be returned.

41278 **ERRORS**

41279 These functions shall fail if:

41280 Pole Error The  $x$  argument is a negative integer or zero.

41281 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 41282 then *errno* shall be set to [ERANGE]. If the integer expression  
 41283 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the divide-by-zero  
 41284 floating-point exception shall be raised.

41285 Range Error The result overflows.

41286 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 41287 then *errno* shall be set to [ERANGE]. If the integer expression

41288 (math\_errhandling & MATH\_ERREXCEPT) is non-zero, then the overflow  
41289 floating-point exception shall be raised.

#### 41290 EXAMPLES

41291 None.

#### 41292 APPLICATION USAGE

41293 On error, the expressions (math\_errhandling & MATH\_ERRNO) and (math\_errhandling &  
41294 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

#### 41295 RATIONALE

41296 None.

#### 41297 FUTURE DIRECTIONS

41298 None.

#### 41299 SEE ALSO

41300 [exp\(\)](#), [feclearexcept\(\)](#), [fetestexcept\(\)](#), [isnan\(\)](#)

41301 XBD [Section 4.20](#) (on page 117), [<math.h>](#)

#### 41302 CHANGE HISTORY

41303 First released in Issue 3.

#### 41304 Issue 5

41305 The DESCRIPTION is updated to indicate how an application should check for an error. This  
41306 text was previously published in the APPLICATION USAGE section.

41307 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

#### 41308 Issue 6

41309 The [lgamma\(\)](#) function is no longer marked as an extension.

41310 The [lgammaf\(\)](#) and [lgammal\(\)](#) functions are added for alignment with the ISO/IEC 9899:1999  
41311 standard.

41312 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
41313 revised to align with the ISO/IEC 9899:1999 standard.

41314 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
41315 marked.

41316 Functionality relating to the XSI option is marked.

#### 41317 Issue 7

41318 Austin Group Interpretation 1003.1-2001 #156 is applied.

41319 The DESCRIPTION is clarified regarding the value of *signgam* when *x* is Nan, -Inf, or a negative  
41320 integer.

41321 **NAME**

41322 link, linkat — link one file to another file |

41323 **SYNOPSIS**

41324 #include &lt;unistd.h&gt;

41325 int link(const char \*path1, const char \*path2);

41326 OH #include &lt;fcntl.h&gt; +

41327 +

```
41328 int linkat(int fd1, const char *path1, int fd2,
41329           const char *path2, int flag);
```

41330 **DESCRIPTION**41331 The *link()* function shall create a new link (directory entry) for the existing file, *path1*.

41332 The *path1* argument points to a pathname naming an existing file. The *path2* argument points to  
 41333 a pathname naming the new directory entry to be created. The *link()* function shall atomically  
 41334 create a new link for the existing file and the link count of the file shall be incremented by one.

41335 If *path1* names a directory, *link()* shall fail unless the process has appropriate privileges and the  
 41336 implementation supports using *link()* on directories.

41337 If *path1* names a symbolic link, it is implementation-defined whether *link()* follows the symbolic  
 41338 link, or creates a new link to the symbolic link itself.

41339 Upon successful completion, *link()* shall mark for update the last file status change timestamp of  
 41340 the file. Also, the last data modification and last file status change timestamps of the directory  
 41341 that contains the new entry shall be marked for update.

41342 If *link()* fails, no link shall be created and the link count of the file shall remain unchanged.

41343 The implementation may require that the calling process has permission to access the existing  
 41344 file.

41345 The *linkat()* function shall be equivalent to the *link()* function except that symbolic links shall be  
 41346 handled as specified by the value of *flag* (see below) and except in the case where either *path1* or  
 41347 *path2* or both are relative paths. In this case a relative path *path1* is interpreted relative to the  
 41348 directory associated with the file descriptor *fd1* instead of the current working directory and  
 41349 similarly for *path2* and the file descriptor *fd2*. If the access mode of the open file description |  
 41350 associated with the file descriptor is not O\_SEARCH, the function shall check whether directory |  
 41351 searches are permitted using the current permissions of the directory underlying the file |  
 41352 descriptor. If the access mode is O\_SEARCH, the function shall not perform the check.

41353 Values for *flag* are constructed by a bitwise-inclusive OR of flags from the following list, defined  
 41354 in <fcntl.h>:

41355 AT\_SYMLINK\_FOLLOW

41356 If *path1* names a symbolic link, a new link for the target of the symbolic link is created.

41357 If *linkat()* is passed the special value AT\_FDCWD in the *fd1* or *fd2* parameter, the current  
 41358 working directory shall be used for the respective *path* argument. If both *fd1* and *fd2* have value  
 41359 AT\_FDCWD, the behavior shall be identical to a call to *link()*, except that symbolic links shall be  
 41360 handled as specified by the value of *flag*.

41361 If the AT\_SYMLINK\_FOLLOW flag is clear in the *flag* argument and the *path1* argument names a  
 41362 symbolic link, a new link is created for the symbolic link *path1* and not its target.

41363 **RETURN VALUE**

41364       Upon successful completion, these functions shall return 0. Otherwise, these functions shall  
41365       return -1 and set *errno* to indicate the error.

41366 **ERRORS**

41367       These functions shall fail if:

- |                                                                      |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------------------------------------------------------------|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 41368<br>41369<br>41370<br>41371                                     | [EACCES]              | A component of either path prefix denies search permission, or the requested link requires writing in a directory that denies write permission, or the calling process does not have permission to access the existing file and this is required by the implementation.                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 41372<br>41373                                                       | [EEXIST]              | The <i>path2</i> argument resolves to an existing directory entry or refers to a symbolic link.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 41374<br>41375                                                       | [ELOOP]               | A loop exists in symbolic links encountered during resolution of the <i>path1</i> or <i>path2</i> argument.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 41376                                                                | [EMLINK]              | The number of links to the file named by <i>path1</i> would exceed {LINK_MAX}.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 41377<br>41378                                                       | [ENAMETOOLONG]        | The length of a component of a pathname is longer than {NAME_MAX}.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 41379<br>41380                                                       | [ENOENT]              | A component of either path prefix does not exist; the file named by <i>path1</i> does not exist; or <i>path1</i> or <i>path2</i> points to an empty string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 41381<br>41382<br>41383<br>41384<br>41385                            | [ENOENT] or [ENOTDIR] | The <i>path1</i> argument names an existing non-directory file, and the <i>path2</i> +<br>argument contains at least one non- <code>&lt;slash&gt;</code> character and ends with one or +<br>more trailing <code>&lt;slash&gt;</code> characters. If <i>path2</i> without the trailing <code>&lt;slash&gt;</code> +<br>characters would name an existing file, an [ENOENT] error shall not occur.                                                                                                                                                                                                                                                                                          |
| 41386                                                                | [ENOSPC]              | The directory to contain the link cannot be extended.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 41387<br>41388<br>41389<br>41390<br>41391<br>41392<br>41393<br>41394 | [ENOTDIR]             | A component of either path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>path1</i> argument contains at least one non- <code>&lt;slash&gt;</code> character and ends with one or more trailing <code>&lt;slash&gt;</code> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>path1</i> argument names an existing non-directory file and the <i>path2</i> argument names a nonexistent file, contains at least one non- <code>&lt;slash&gt;</code> character, and ends with one or more trailing <code>&lt;slash&gt;</code> characters. |
| 41395<br>41396<br>41397                                              | [EPERM]               | The file named by <i>path1</i> is a directory and either the calling process does not have appropriate privileges or the implementation prohibits using <i>link()</i> on directories.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 41398                                                                | [EROFS]               | The requested link requires writing in a directory on a read-only file system.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 41399<br>41400                                                       | [EXDEV]               | The link named by <i>path2</i> and the file named by <i>path1</i> are on different file systems and the implementation does not support links between file systems.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 41401                                                                | OB XSR [EXDEV]        | <i>path1</i> refers to a named STREAM.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

41402       The *linkat()* function shall fail if:

- |                         |          |                                                                                                                                                                                                                                            |
|-------------------------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 41403<br>41404<br>41405 | [EACCES] | The access mode of the open file description associated with <i>fd1</i> or <i>fd2</i> is not +<br>O_SEARCH and the permissions of the directory underlying <i>fd1</i> or <i>fd2</i> , +<br>respectively, do not permit directory searches. |
|-------------------------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



41406 [EBADF] The *path1* or *path2* argument does not specify an absolute path and the *fd1* or  
 41407 *fd2* argument, respectively, is neither AT\_FDCWD nor a valid file descriptor  
 41408 open for reading or searching.

41409 [ENOTDIR] The *path1* or *path2* argument is not an absolute path and *fd1* or *fd2*,  
 41410 respectively, is a file descriptor associated with a non-directory file.

41411 These functions may fail if:

41412 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
 41413 resolution of the *path1* or *path2* argument.

41414 [ENAMETOOLONG]  
 41415 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
 41416 symbolic link produced an intermediate result with a length that exceeds  
 41417 {PATH\_MAX}.

41418 The *linkat()* function may fail if:

41419 [EINVAL] The value of the *flag* argument is not valid.

## 41420 EXAMPLES

### 41421 Creating a Link to a File

41422 The following example shows how to create a link to a file named **/home/cnd/mod1** by creating  
 41423 a new directory entry named **/modules/pass1**.

```
41424 #include <unistd.h>
41425 char *path1 = "/home/cnd/mod1";
41426 char *path2 = "/modules/pass1";
41427 int status;
41428 ...
41429 status = link (path1, path2);
```

### 41430 Creating a Link to a File Within a Program

41431 In the following program example, the *link()* function links the **/etc/passwd** file (defined as  
 41432 **PASSWDFILE**) to a file named **/etc/opasswd** (defined as **SAVEFILE**), which is used to save the  
 41433 current password file. Then, after removing the current password file (defined as  
 41434 **PASSWDFILE**), the new password file is saved as the current password file using the *link()*  
 41435 function again.

```
41436 #include <unistd.h>
41437 #define LOCKFILE "/etc/ptmp"
41438 #define PASSWDFILE "/etc/passwd"
41439 #define SAVEFILE "/etc/opasswd"
41440 ...
41441 /* Save current password file */
41442 link (PASSWDFILE, SAVEFILE);
41443
41444 /* Remove current password file. */
41445 unlink (PASSWDFILE);
41446
41447 /* Save new password file as current password file. */
41448 link (LOCKFILE, PASSWDFILE);
```

**APPLICATION USAGE**

Some implementations do allow links between file systems.

If *path1* refers to a symbolic link, application developers should use *linkat()* with appropriate flags to select whether or not the symbolic link should be resolved.

**RATIONALE**

Linking to a directory is restricted to the superuser in most historical implementations because this capability may produce loops in the file hierarchy or otherwise corrupt the file system. This volume of POSIX.1-2008 continues that philosophy by prohibiting *link()* and *unlink()* from doing this. Other functions could do it if the implementor designed such an extension.

Some historical implementations allow linking of files on different file systems. Wording was added to explicitly allow this optional behavior.

The exception for cross-file system links is intended to apply only to links that are programmatically indistinguishable from “hard” links.

The purpose of the *linkat()* function is to link files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *link()*, resulting in unspecified behavior. By opening a file descriptor for the directory of both the existing file and the target location and using the *linkat()* function it can be guaranteed that the both filenames are in the desired directories.

The AT\_SYMLINK\_FOLLOW flag allows for implementing both common behaviors of the *link()* function. The POSIX specification requires that if *path1* is a symbolic link, a new link for the target of the symbolic link is created. Many systems by default or as an alternative provide a mechanism to avoid the implicit symbolic link lookup and create a new link for the symbolic link itself.

Earlier versions of this standard specified only the *link()* function, and required it to behave like *linkat()* with the AT\_SYMLINK\_FOLLOW flag. However, historical practice from SVR4 and Linux kernels had *link()* behaving like *linkat()* with no flags, and many systems that attempted to provide a conforming *link()* function did so in a way that was rarely used, and when it was used did not conform to the standard (e.g., by not being atomic, or by dereferencing the symbolic link incorrectly). Since applications could not rely on *link()* following links in practice, the *linkat()* function was added taking a flag to specify the desired behavior for the application.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*rename()*, *symlink()*, *unlink()*

XBD <fcntl.h>, <unistd.h>

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 6**

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- An explanation is added of the action when *path2* refers to a symbolic link.
- The [ELOOP] optional error condition is added.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #143 is applied.

SD5-XSH-ERN-93 is applied, adding RATIONALE.

The *linkat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Functionality relating to XSI STREAMS is marked obsolescent.

Changes are made related to support for finegrained timestamps.

The [EOPNOTSUPP] error is removed.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0354 [326], XSH/TC1-2008/0355 [461], XSH/TC1-2008/0356 [326], XSH/TC1-2008/0357 [324], XSH/TC1-2008/0358 [147,429], XSH/TC1-2008/0359 [277], XSH/TC1-2008/0360 [278], and XSH/TC1-2008/0361 [278] are applied.

**NAME**

lio\_listio — list directed I/O

**SYNOPSIS**

```
#include <aio.h>
```

```
int lio_listio(int mode, struct aiocb *restrict const list[restrict],  
              int nent, struct sigevent *restrict sig);
```

**DESCRIPTION**

The *lio\_listio()* function shall initiate a list of I/O requests with a single function call.

The *mode* argument takes one of the values LIO\_WAIT or LIO\_NOWAIT declared in **<aio.h>** and determines whether the function returns when the I/O operations have been completed, or as soon as the operations have been queued. If the *mode* argument is LIO\_WAIT, the function shall wait until all I/O is complete and the *sig* argument shall be ignored.

If the *mode* argument is LIO\_NOWAIT, the function shall return immediately, and asynchronous notification shall occur, according to the *sig* argument, when all the I/O operations complete. If *sig* is NULL, then no asynchronous notification shall occur. If *sig* is not NULL, asynchronous notification occurs as specified in [Section 2.4.1](#) (on page 488) when all the requests in *list* have completed.

The I/O requests enumerated by *list* are submitted in an unspecified order.

The *list* argument is an array of pointers to **aiocb** structures. The array contains *nent* elements. The array may contain NULL elements, which shall be ignored.

If the buffer pointed to by *list* or the **aiocb** structures pointed to by the elements of the array *list* become illegal addresses before all asynchronous I/O completed and, if necessary, the notification is sent, then the behavior is undefined. If the buffers pointed to by the *aio\_buf* member of the **aiocb** structure pointed to by the elements of the array *list* become illegal addresses prior to the asynchronous I/O associated with that **aiocb** structure being completed, the behavior is undefined.

The *aio\_lio\_opcode* field of each **aiocb** structure specifies the operation to be performed. The supported operations are LIO\_READ, LIO\_WRITE, and LIO\_NOP; these symbols are defined in **<aio.h>**. The LIO\_NOP operation causes the list entry to be ignored. If the *aio\_lio\_opcode* element is equal to LIO\_READ, then an I/O operation is submitted as if by a call to *aio\_read()* with the *aio\_cbp* equal to the address of the **aiocb** structure. If the *aio\_lio\_opcode* element is equal to LIO\_WRITE, then an I/O operation is submitted as if by a call to *aio\_write()* with the *aio\_cbp* equal to the address of the **aiocb** structure.

The *aio\_fildes* member specifies the file descriptor on which the operation is to be performed.

The *aio\_buf* member specifies the address of the buffer to or from which the data is transferred.

The *aio\_nbytes* member specifies the number of bytes of data to be transferred.

The members of the **aiocb** structure further describe the I/O operation to be performed, in a manner identical to that of the corresponding **aiocb** structure when used by the *aio\_read()* and *aio\_write()* functions.

The *nent* argument specifies how many elements are members of the list; that is, the length of the array.

The behavior of this function is altered according to the definitions of synchronized I/O data integrity completion and synchronized I/O file integrity completion if synchronized I/O is enabled on the file associated with *aio\_fildes*.

For regular files, no data transfer shall occur past the offset maximum established in the open file description associated with *aio*cbp->*aio\_fildes*.

If *sig*->*sigev\_notify* is SIGEV\_THREAD and *sig*->*sigev\_notify\_attributes* is a non-null pointer and the block pointed to by this pointer becomes an illegal address prior to all asynchronous I/O being completed, then the behavior is undefined.

## RETURN VALUE

If the *mode* argument has the value LIO\_NOWAIT, the *lio\_listio()* function shall return the value zero if the I/O operations are successfully queued; otherwise, the function shall return the value -1 and set *errno* to indicate the error.

If the *mode* argument has the value LIO\_WAIT, the *lio\_listio()* function shall return the value zero when all the indicated I/O has completed successfully. Otherwise, *lio\_listio()* shall return a value of -1 and set *errno* to indicate the error.

In either case, the return value only indicates the success or failure of the *lio\_listio()* call itself, not the status of the individual I/O requests. In some cases one or more of the I/O requests contained in the list may fail. Failure of an individual request does not prevent completion of any other individual request. To determine the outcome of each I/O request, the application shall examine the error status associated with each **aio**cb control block. The error statuses so returned are identical to those returned as the result of an *aio\_read()* or *aio\_write()* function.

## ERRORS

The *lio\_listio()* function shall fail if:

[EAGAIN] The resources necessary to queue all the I/O requests were not available. The application may check the error status for each **aio**cb to determine the individual request(s) that failed.

[EAGAIN] The number of entries indicated by *nent* would cause the system-wide limit {AIO\_MAX} to be exceeded.

[EINVAL] The *mode* argument is not a proper value, or the value of *nent* was greater than {AIO\_LISTIO\_MAX}.

[EINTR] A signal was delivered while waiting for all I/O requests to complete during an LIO\_WAIT operation. Note that, since each I/O operation invoked by *lio\_listio()* may possibly provoke a signal when it completes, this error return may be caused by the completion of one (or more) of the very I/O operations being awaited. Outstanding I/O requests are not canceled, and the application shall examine each list element to determine whether the request was initiated, canceled, or completed.

[EIO] One or more of the individual I/O operations failed. The application may check the error status for each **aio**cb structure to determine the individual request(s) that failed.

In addition to the errors returned by the *lio\_listio()* function, if the *lio\_listio()* function succeeds or fails with errors of [EAGAIN], [EINTR], or [EIO], then some of the I/O specified by the list may have been initiated. If the *lio\_listio()* function fails with an error code other than [EAGAIN], [EINTR], or [EIO], no operations from the list shall have been initiated. The I/O operation indicated by each list element can encounter errors specific to the individual read or write function being performed. In this event, the error status for each **aio**cb control block contains the associated error code. The error codes that can be set are the same as would be set by a *read()* or *write()* function, with the following additional error codes possible:

41593 [EAGAIN] The requested I/O operation was not queued due to resource limitations.

41594 [ECANCELED] The requested I/O was canceled before the I/O completed due to an explicit  
41595 *aio\_cancel()* request.

41596 [EFBIG] The *aiocbp->aio\_lio\_opcode* is LIO\_WRITE, the file is a regular file,  
41597 *aiocbp->aio\_nbytes* is greater than 0, and the *aiocbp->aio\_offset* is greater than or  
41598 equal to the offset maximum in the open file description associated with  
41599 *aiocbp->aio\_fildes*.

41600 [EINPROGRESS] The requested I/O is in progress.

41601 [EOVERFLOW] The *aiocbp->aio\_lio\_opcode* is LIO\_READ, the file is a regular file,  
41602 *aiocbp->aio\_nbytes* is greater than 0, and the *aiocbp->aio\_offset* is before the  
41603 end-of-file and is greater than or equal to the offset maximum in the open file  
41604 description associated with *aiocbp->aio\_fildes*.

**EXAMPLES**

41605 None.

**APPLICATION USAGE**

41608 None.

**RATIONALE**

41610 Although it may appear that there are inconsistencies in the specified circumstances for error  
41611 codes, the [EIO] error condition applies when any circumstance relating to an individual  
41612 operation makes that operation fail. This might be due to a badly formulated request (for  
41613 example, the *aio\_lio\_opcode* field is invalid, and *aio\_error()* returns [EINVAL]) or might arise from  
41614 application behavior (for example, the file descriptor is closed before the operation is initiated,  
41615 and *aio\_error()* returns [EBADF]).

41616 The limitation on the set of error codes returned when operations from the list shall have been  
41617 initiated enables applications to know when operations have been started and whether  
41618 *aio\_error()* is valid for a specific operation.

**FUTURE DIRECTIONS**

41619 None.

**SEE ALSO**

41622 *aio\_read()*, *aio\_write()*, *aio\_error()*, *aio\_return()*, *aio\_cancel()*, *close()*, *exec*, *exit()*, *fork()*, *lseek()*,  
41623 *read()*

41624 XBD <*aio.h*>

**CHANGE HISTORY**

41625 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

41627 Large File Summit extensions are added.

**Issue 6**

41629 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
41630 implementation does not support the Asynchronous Input and Output option.

41631 The *lio\_listio()* function is marked as part of the Asynchronous Input and Output option.

41632 The following new requirements on POSIX implementations derive from alignment with the  
41633 Single UNIX Specification:

41634 • In the DESCRIPTION, text is added to indicate that for regular files no data transfer occurs  
41635 past the offset maximum established in the open file description associated with  
41636 *aiocbp*→*aio\_fildes*. This change is to support large files.

41637 • The [EBIG] and [EOVERFLOW] error conditions are defined. This change is to support  
41638 large files.

41639 The normative text is updated to avoid use of the term “must” for application requirements.

41640 The **restrict** keyword is added to the *lio\_listio()* prototype for alignment with the  
41641 ISO/IEC 9899:1999 standard.

#### 41642 Issue 6

41643 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/53 is applied, adding new text for  
41644 symmetry with the *aio\_read()* and *aio\_write()* functions to the DESCRIPTION.

41645 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/54 is applied, adding text to the  
41646 DESCRIPTION making it explicit that the user is required to keep the structure pointed to by  
41647 *sig*→*sigev\_notify\_attributes* valid until the last asynchronous operation finished and the  
41648 notification has been sent.

#### 41649 Issue 7

41650 The *lio\_listio()* function is moved from the Asynchronous Input and Output option to the Base.



41651 **NAME**

41652 listen — listen for socket connections and limit the queue of incoming connections

41653 **SYNOPSIS**

41654 #include &lt;sys/socket.h&gt;

41655 int listen(int *socket*, int *backlog*);41656 **DESCRIPTION**41657 The *listen()* function shall mark a connection-mode socket, specified by the *socket* argument, as  
41658 accepting connections.41659 The *backlog* argument provides a hint to the implementation which the implementation shall use  
41660 to limit the number of outstanding connections in the socket's listen queue. Implementations  
41661 may impose a limit on *backlog* and silently reduce the specified value. Normally, a larger *backlog*  
41662 argument value shall result in a larger or equal length of the listen queue. Implementations shall  
41663 support values of *backlog* up to SOMAXCONN, defined in <sys/socket.h>.41664 The implementation may include incomplete connections in its listen queue. The limits on the  
41665 number of incomplete connections and completed connections queued may be different.41666 The implementation may have an upper limit on the length of the listen queue—either global or  
41667 per accepting socket. If *backlog* exceeds this limit, the length of the listen queue is set to the limit.41668 If *listen()* is called with a *backlog* argument value that is less than 0, the function behaves as if it  
41669 had been called with a *backlog* argument value of 0.41670 A *backlog* argument of 0 may allow the socket to accept connections, in which case the length of  
41671 the listen queue may be set to an implementation-defined minimum value.41672 The socket in use may require the process to have appropriate privileges to use the *listen()*  
41673 function.41674 **RETURN VALUE**41675 Upon successful completions, *listen()* shall return 0; otherwise, -1 shall be returned and *errno* set  
41676 to indicate the error.41677 **ERRORS**41678 The *listen()* function shall fail if:41679 [EBADF] The *socket* argument is not a valid file descriptor.

41680 [EDESTADDRREQ]

41681 The socket is not bound to a local address, and the protocol does not support  
41682 listening on an unbound socket.41683 [EINVAL] The *socket* is already connected.41684 [ENOTSOCK] The *socket* argument does not refer to a socket.41685 [EOPNOTSUPP] The socket protocol does not support *listen()*.41686 The *listen()* function may fail if:

41687 [EACCES] The calling process does not have appropriate privileges.

41688 [EINVAL] The *socket* has been shut down.

41689 [ENOBUFS] Insufficient resources are available in the system to complete the call.



41690 **EXAMPLES**

41691 None.

41692 **APPLICATION USAGE**

41693 None.

41694 **RATIONALE**

41695 None.

41696 **FUTURE DIRECTIONS**

41697 None.

41698 **SEE ALSO**41699 *accept()*, *connect()*, *socket()*

41700 XBD &lt;sys/socket.h&gt;

41701 **CHANGE HISTORY**

41702 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

41703 The DESCRIPTION is updated to describe the relationship of SOMAXCONN and the *backlog*  
41704 argument.

41705 **NAME**

41706       llabs — return a long integer absolute value

41707 **SYNOPSIS**

41708       #include &lt;stdlib.h&gt;

41709       long long llabs(long long i);

41710 **DESCRIPTION**41711       Refer to *labs()*.

41712 **NAME**

41713       lldiv — compute quotient and remainder of a long division

41714 **SYNOPSIS**

41715       #include &lt;stdlib.h&gt;

41716       lldiv\_t lldiv(long long *numer*, long long *denom*);41717 **DESCRIPTION**41718       Refer to *ldiv()*.

**NAME**

llrint, llrintf, llrintl — round to the nearest integer value using current rounding direction

**SYNOPSIS**

```
#include <math.h>

long long llrint(double x);
long long llrintf(float x);
long long llrintl(long double x);
```

**DESCRIPTION**

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

These functions shall round their argument to the nearest integer value, rounding according to the current rounding direction.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-zero, an error has occurred.

**RETURN VALUE**

Upon successful completion, these functions shall return the rounded integer value.

MX If *x* is NaN, a domain error shall occur, and an unspecified value is returned.

If *x* is +Inf, a domain error shall occur and an unspecified value is returned.

If *x* is -Inf, a domain error shall occur and an unspecified value is returned.

MX If the correct value is positive and too large to represent as a **long long**, an unspecified value shall be returned. On systems that support the IEC 60559 Floating-Point option, a domain error shall occur; otherwise, a domain error may occur.

MX If the correct value is negative and too large to represent as a **long long**, an unspecified value shall be returned. On systems that support the IEC 60559 Floating-Point option, a domain error shall occur; otherwise, a domain error may occur.

**ERRORS**

These functions shall fail if:

MX **Domain Error** The *x* argument is NaN or  $\pm$ Inf, or the correct value is not representable as an integer.

If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

These functions may fail if:

**Domain Error** The correct value is not representable as an integer.

If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero, then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.

41761 **EXAMPLES**

41762       None.

41763 **APPLICATION USAGE**

41764       On error, the expressions *(math\_errhandling & MATH\_ERRNO)* and *(math\_errhandling &*  
41765       *MATH\_ERREXCEPT)* are independent of each other, but at least one of them must be non-zero.

41766 **RATIONALE**

41767       These functions provide floating-to-integer conversions. They round according to the current  
41768       rounding direction. If the rounded value is outside the range of the return type, the numeric  
41769       result is unspecified and the invalid floating-point exception is raised. When they raise no other  
41770       floating-point exception and the result differs from the argument, they raise the inexact floating-  
41771       point exception.

41772 **FUTURE DIRECTIONS**

41773       None.

41774 **SEE ALSO**41775       *feclearexcept()*, *fetestexcept()*, *lrint()*41776       XBD Section 4.20 (on page 117), *<math.h>*41777 **CHANGE HISTORY**

41778       First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

41779 **Issue 7**

41780       ISO/IEC 9899: 1999 standard, Technical Corrigendum 2 #53 is applied.

41781 **NAME**

41782 llround, llroundf, llroundl — round to nearest integer value

41783 **SYNOPSIS**

41784 #include <math.h>  
41785 long long llround(double x);  
41786 long long llroundf(float x);  
41787 long long llroundl(long double x);

41788 **DESCRIPTION**

41789 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
41790 conflict between the requirements described here and the ISO C standard is unintentional. This  
41791 volume of POSIX.1-2008 defers to the ISO C standard.

41792 These functions shall round their argument to the nearest integer value, rounding halfway cases  
41793 away from zero, regardless of the current rounding direction.

41794 An application wishing to check for error situations should set *errno* to zero and call  
41795 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
41796 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
41797 zero, an error has occurred.

41798 **RETURN VALUE**

41799 Upon successful completion, these functions shall return the rounded integer value.

41800 MX If *x* is NaN, a domain error shall occur, and an unspecified value is returned.

41801 If *x* is +Inf, a domain error shall occur and an unspecified value is returned.

41802 If *x* is -Inf, a domain error shall occur and an unspecified value is returned.

41803 If the correct value is positive and too large to represent as a **long long**, an unspecified value  
41804 MX shall be returned. On systems that support the IEC 60559 Floating-Point option, a domain error  
41805 CX shall occur; otherwise, a domain error may occur.

41806 If the correct value is negative and too large to represent as a **long long**, an unspecified value  
41807 MX shall be returned. On systems that support the IEC 60559 Floating-Point option, a domain error  
41808 CX shall occur; otherwise, a domain error may occur.

41809 **ERRORS**

41810 These functions shall fail if:

41811 MX **Domain Error** The *x* argument is NaN or ±Inf, or the correct value is not representable as an  
41812 integer.

41813 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
41814 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
41815 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
41816 shall be raised.

41817 These functions may fail if:

41818 **Domain Error** The correct value is not representable as an integer.

41819 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
41820 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
41821 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
41822 shall be raised.

**41823 EXAMPLES**

41824       None.

**41825 APPLICATION USAGE**

41826       On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
41827       MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

**41828 RATIONALE**

41829       These functions differ from the *llrint()* functions in that the default rounding direction for the  
41830       *llround()* functions round halfway cases away from zero and need not raise the inexact floating-  
41831       point exception for non-integer arguments that round to within the range of the return type.

**41832 FUTURE DIRECTIONS**

41833       None.

**41834 SEE ALSO**

41835       *feclearexcept()*, *fetestexcept()*, *lround()*

41836       XBD Section 4.20 (on page 117), [<math.h>](#)

**41837 CHANGE HISTORY**

41838       First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

**41839 Issue 7**

41840       ISO/IEC 9899: 1999 standard, Technical Corrigendum 2 #54 (SD5-XSH-ERN-75) is applied.

41841 **NAME**

41842 localeconv — return locale-specific information

41843 **SYNOPSIS**

41844 #include &lt;locale.h&gt;

41845 struct lconv \*localeconv(void);

41846 **DESCRIPTION**

41847 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
41848 conflict between the requirements described here and the ISO C standard is unintentional. This  
41849 volume of POSIX.1-2008 defers to the ISO C standard.

41850 The *localeconv()* function shall set the components of an object with the type **struct lconv** with  
41851 the values appropriate for the formatting of numeric quantities (monetary and otherwise)  
41852 according to the rules of the current locale.

41853 The members of the structure with type **char \*** are pointers to strings, any of which (except  
41854 **decimal\_point**) can point to " ", to indicate that the value is not available in the current locale or  
41855 is of zero length. The members with type **char** are non-negative numbers, any of which can be  
41856 {CHAR\_MAX} to indicate that the value is not available in the current locale.

41857 The members include the following:

41858 **char \*decimal\_point**

41859 The radix character used to format non-monetary quantities.

41860 **char \*thousands\_sep**

41861 The character used to separate groups of digits before the decimal-point character in  
41862 formatted non-monetary quantities.

41863 **char \*grouping**

41864 A string whose elements taken as one-byte integer values indicate the size of each group of  
41865 digits in formatted non-monetary quantities.

41866 **char \*int\_curr\_symbol**

41867 The international currency symbol applicable to the current locale. The first three characters  
41868 contain the alphabetic international currency symbol in accordance with those specified in  
41869 the ISO 4217:2001 standard. The fourth character (immediately preceding the null byte) is  
41870 the character used to separate the international currency symbol from the monetary  
41871 quantity.

41872 **char \*currency\_symbol**

41873 The local currency symbol applicable to the current locale.

41874 **char \*mon\_decimal\_point**

41875 The radix character used to format monetary quantities.

41876 **char \*mon\_thousands\_sep**

41877 The separator for groups of digits before the decimal-point in formatted monetary  
41878 quantities.

41879 **char \*mon\_grouping**

41880 A string whose elements taken as one-byte integer values indicate the size of each group of  
41881 digits in formatted monetary quantities.

41882 **char \*positive\_sign**

41883 The string used to indicate a non-negative valued formatted monetary quantity.



41884 **char \*negative\_sign**  
41885 The string used to indicate a negative valued formatted monetary quantity.

41886 **char int\_frac\_digits**  
41887 The number of fractional digits (those after the decimal-point) to be displayed in an  
41888 internationally formatted monetary quantity.

41889 **char frac\_digits**  
41890 The number of fractional digits (those after the decimal-point) to be displayed in a  
41891 formatted monetary quantity.

41892 **char p\_cs\_precedes**  
41893 Set to 1 if the **currency\_symbol** precedes the value for a non-negative formatted monetary  
41894 quantity. Set to 0 if the symbol succeeds the value.

41895 **char p\_sep\_by\_space**  
41896 Set to a value indicating the separation of the **currency\_symbol**, the sign string, and the  
41897 value for a non-negative formatted monetary quantity.

41898 **char n\_cs\_precedes**  
41899 Set to 1 if the **currency\_symbol** precedes the value for a negative formatted monetary  
41900 quantity. Set to 0 if the symbol succeeds the value.

41901 **char n\_sep\_by\_space**  
41902 Set to a value indicating the separation of the **currency\_symbol**, the sign string, and the  
41903 value for a negative formatted monetary quantity.

41904 **char p\_sign\_posn**  
41905 Set to a value indicating the positioning of the **positive\_sign** for a non-negative formatted  
41906 monetary quantity.

41907 **char n\_sign\_posn**  
41908 Set to a value indicating the positioning of the **negative\_sign** for a negative formatted  
41909 monetary quantity.

41910 **char int\_p\_cs\_precedes**  
41911 Set to 1 or 0 if the **int\_curr\_symbol** respectively precedes or succeeds the value for a non-  
41912 negative internationally formatted monetary quantity.

41913 **char int\_n\_cs\_precedes**  
41914 Set to 1 or 0 if the **int\_curr\_symbol** respectively precedes or succeeds the value for a  
41915 negative internationally formatted monetary quantity.

41916 **char int\_p\_sep\_by\_space**  
41917 Set to a value indicating the separation of the **int\_curr\_symbol**, the sign string, and the  
41918 value for a non-negative internationally formatted monetary quantity.

41919 **char int\_n\_sep\_by\_space**  
41920 Set to a value indicating the separation of the **int\_curr\_symbol**, the sign string, and the  
41921 value for a negative internationally formatted monetary quantity.

41922 **char int\_p\_sign\_posn**  
41923 Set to a value indicating the positioning of the **positive\_sign** for a non-negative  
41924 internationally formatted monetary quantity.

41925 **char int\_n\_sign\_posn**  
41926 Set to a value indicating the positioning of the **negative\_sign** for a negative internationally  
41927 formatted monetary quantity.

- 41928 The elements of **grouping** and **mon\_grouping** are interpreted according to the following:
- 41929 {CHAR\_MAX} No further grouping is to be performed.
- 41930 0 The previous element is to be repeatedly used for the remainder of the digits.
- 41931 *other* The integer value is the number of digits that comprise the current group. The
- 41932 next element is examined to determine the size of the next group of digits
- 41933 before the current group.
- 41934 The values of **p\_sep\_by\_space**, **n\_sep\_by\_space**, **int\_p\_sep\_by\_space**, and **int\_n\_sep\_by\_space**
- 41935 are interpreted according to the following:
- 41936 0 No space separates the currency symbol and value.
- 41937 1 If the currency symbol and sign string are adjacent, a space separates them from the value;
- 41938 otherwise, a space separates the currency symbol from the value.
- 41939 2 If the currency symbol and sign string are adjacent, a space separates them; otherwise, a
- 41940 space separates the sign string from the value.
- 41941 For **int\_p\_sep\_by\_space** and **int\_n\_sep\_by\_space**, the fourth character of **int\_curr\_symbol** is
- 41942 used instead of a space.
- 41943 The values of **p\_sign\_posn**, **n\_sign\_posn**, **int\_p\_sign\_posn**, and **int\_n\_sign\_posn** are
- 41944 interpreted according to the following:
- 41945 0 Parentheses surround the quantity and **currency\_symbol** or **int\_curr\_symbol**.
- 41946 1 The sign string precedes the quantity and **currency\_symbol** or **int\_curr\_symbol**.
- 41947 2 The sign string succeeds the quantity and **currency\_symbol** or **int\_curr\_symbol**.
- 41948 3 The sign string immediately precedes the **currency\_symbol** or **int\_curr\_symbol**.
- 41949 4 The sign string immediately succeeds the **currency\_symbol** or **int\_curr\_symbol**.
- 41950 The implementation shall behave as if no function in this volume of POSIX.1-2008 calls
- 41951 *localeconv*( ).
- 41952 CX The *localeconv*( ) function need not be thread-safe.
- 41953 **RETURN VALUE**
- 41954 The *localeconv*( ) function shall return a pointer to the filled-in object. The application shall not
- 41955 CX modify the structure to which the return value points, nor any storage areas pointed to by
- 41956 pointers within the structure. The returned pointer, and pointers within the structure, might be
- 41957 CX invalidated or the structure or the storage areas might be overwritten by a subsequent call to
- 41958 CX *localeconv*( ). In addition, the returned pointer, and pointers within the structure, might be
- 41959 CX invalidated or the structure or the storage areas might be overwritten by subsequent calls to
- 41960 CX *setlocale*( ) with the categories LC\_ALL, LC\_MONETARY, or LC\_NUMERIC, or by calls to
- 41961 *uselocale*( ) which change the categories LC\_MONETARY or LC\_NUMERIC. The returned
- 41962 pointer, pointers within the structure, the structure, and the storage areas might also be
- 41963 invalidated if the calling thread is terminated.
- 41964 **ERRORS**
- 41965 No errors are defined.

**EXAMPLES**

None.

**APPLICATION USAGE**

The following table illustrates the rules which may be used by four countries to format monetary quantities.

| Country     | Positive Format | Negative Format | International Format |
|-------------|-----------------|-----------------|----------------------|
| Italy       | €1.230          | −€1.230         | EUR.1.230            |
| Netherlands | € 1.234,56      | € −1.234,56     | EUR 1.234,56         |
| Norway      | kr1.234,56      | kr1.234,56−     | NOK 1.234,56         |
| Switzerland | SFrs.1,234.56   | SFrs.1,234.56C  | CHF 1,234.56         |

For these four countries, the respective values for the monetary members of the structure returned by *localeconv()* are:

|                           | Italy   | Netherlands | Norway | Switzerland |
|---------------------------|---------|-------------|--------|-------------|
| <b>int_curr_symbol</b>    | "EUR. " | "EUR "      | "NOK " | "CHF "      |
| <b>currency_symbol</b>    | "€. "   | "€ "        | "kr "  | "SFrs. "    |
| <b>mon_decimal_point</b>  | " "     | " , "       | " , "  | " . "       |
| <b>mon_thousands_sep</b>  | " . "   | " . "       | " . "  | " , "       |
| <b>mon_grouping</b>       | "\3 "   | "\3 "       | "\3 "  | "\3 "       |
| <b>positive_sign</b>      | " "     | " "         | " "    | " "         |
| <b>negative_sign</b>      | " − "   | " − "       | " − "  | "C "        |
| <b>int_frac_digits</b>    | 0       | 2           | 2      | 2           |
| <b>frac_digits</b>        | 0       | 2           | 2      | 2           |
| <b>p_cs_precedes</b>      | 1       | 1           | 1      | 1           |
| <b>p_sep_by_space</b>     | 0       | 1           | 0      | 0           |
| <b>n_cs_precedes</b>      | 1       | 1           | 1      | 1           |
| <b>n_sep_by_space</b>     | 0       | 1           | 0      | 0           |
| <b>p_sign_posn</b>        | 1       | 1           | 1      | 1           |
| <b>n_sign_posn</b>        | 1       | 4           | 2      | 2           |
| <b>int_p_cs_precedes</b>  | 1       | 1           | 1      | 1           |
| <b>int_n_cs_precedes</b>  | 1       | 1           | 1      | 1           |
| <b>int_p_sep_by_space</b> | 0       | 0           | 0      | 0           |
| <b>int_n_sep_by_space</b> | 0       | 0           | 0      | 0           |
| <b>int_p_sign_posn</b>    | 1       | 1           | 1      | 1           |
| <b>int_n_sign_posn</b>    | 1       | 4           | 4      | 2           |

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*fprintf()*, *fscanf()*, *isalpha()*, *isascii()*, *nl\_langinfo()*, *setlocale()*, *strcat()*, *strchr()*, *strcmp()*, *strcoll()*, *strcpy()*, *strptime()*, *strlen()*, *strpbrk()*, *strspn()*, *strtok()*, *strxfrm()*, *strtod()*, *uselocale()*

XBD [<langinfo.h>](#), [<locale.h>](#)

42008 **CHANGE HISTORY**

42009 First released in Issue 4. Derived from the ANSI C standard.

42010 **Issue 6**

42011 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

42012 The RETURN VALUE section is rewritten to avoid use of the term “must”.

42013 This reference page is updated for alignment with the ISO/IEC 9899: 1999 standard.

42014 ISO/IEC 9899: 1999 standard, Technical Corrigendum 1 is incorporated.

42015 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/31 is applied, removing references to  
42016 **int\_curr\_symbol** and updating the descriptions of **p\_sep\_by\_space** and **n\_sep\_by\_space**. These  
42017 changes are for alignment with the ISO C standard.42018 **Issue 7**

42019 Austin Group Interpretation 1003.1-2001 #156 is applied.

42020 The definitions of **int\_curr\_symbol** and **currency\_symbol** are updated.

42021 The examples in the APPLICATION USAGE section are updated.

42022 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0362 [75] is applied.

42023 **NAME**42024 `localtime`, `localtime_r` — convert a time value to a broken-down local time42025 **SYNOPSIS**

```
42026 #include <time.h>
42027 struct tm *localtime(const time_t *timer);
42028 CX struct tm *localtime_r(const time_t *restrict timer,
42029 struct tm *restrict result);
```

42030 **DESCRIPTION**

42031 CX For `localtime()`: The functionality described on this reference page is aligned with the ISO C  
 42032 standard. Any conflict between the requirements described here and the ISO C standard is  
 42033 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

42034 The `localtime()` function shall convert the time in seconds since the Epoch pointed to by `timer`  
 42035 into a broken-down time, expressed as a local time. The function corrects for the timezone and  
 42036 CX any seasonal time adjustments. Local timezone information is used as though `localtime()` calls  
 42037 `tzset()`.

42038 The relationship between a time in seconds since the Epoch used as an argument to `localtime()`  
 42039 and the `tm` structure (defined in the `<time.h>` header) is that the result shall be as specified in  
 42040 the expression given in the definition of seconds since the Epoch (see XBD Section 4.16, on page  
 42041 113) corrected for timezone and any seasonal time adjustments, where the names in the structure  
 42042 and in the expression correspond.

42043 The same relationship shall apply for `localtime_r()`.

42044 The `localtime()` function need not be thread-safe.

42045 The `asctime()`, `ctime()`, `gmtime()`, and `localtime()` functions shall return values in one of two static  
 42046 objects: a broken-down time structure and an array of type `char`. Execution of any of the  
 42047 functions may overwrite the information returned in either of these objects by any of the other  
 42048 functions.

42049 The `localtime_r()` function shall convert the time in seconds since the Epoch pointed to by `timer`  
 42050 into a broken-down time stored in the structure to which `result` points. The `localtime_r()` function  
 42051 shall also return a pointer to that same structure.

42052 Unlike `localtime()`, the `localtime_r()` function is not required to set `tzname`. If `localtime_r()` sets +  
 42053 `tzname`, it shall also set `daylight` and `timezone`. If `localtime_r()` does not set `tzname`, it shall not set  
 42054 `daylight` and shall not set `timezone`.

42055 **RETURN VALUE**

42056 Upon successful completion, the `localtime()` function shall return a pointer to the broken-down  
 42057 CX time structure. If an error is detected, `localtime()` shall return a null pointer and set `errno` to  
 42058 indicate the error.

42059 Upon successful completion, `localtime_r()` shall return a pointer to the structure pointed to by the  
 42060 argument `result`. If an error is detected, `localtime_r()` shall return a null pointer and set `errno` to  
 42061 indicate the error.

42062 **ERRORS**

42063 CX The `localtime()` and `localtime_r()` functions shall fail if:

42064 CX [EOVERFLOW] The result cannot be represented.

42065 **EXAMPLES**42066 **Getting the Local Date and Time**

42067 The following example uses the *time()* function to calculate the time elapsed, in seconds, since  
42068 January 1, 1970 0:00 UTC (the Epoch), *localtime()* to convert that value to a broken-down time,  
42069 and *asctime()* to convert the broken-down time values into a printable string.

```
42070 #include <stdio.h>
42071 #include <time.h>
42072 int main(void)
42073 {
42074     time_t result;
42075     result = time(NULL);
42076     printf("%s%ju secs since the Epoch\n",
42077           asctime(localtime(&result)),
42078           (uintmax_t)result);
42079     return(0);
42080 }
```

42081 This example writes the current time to *stdout* in a form like this:

```
42082 Wed Jun 26 10:32:15 1996
42083 835810335 secs since the Epoch
```

42084 **Getting the Modification Time for a File**

42085 The following example prints the last data modification timestamp in the local timezone for a  
42086 given file.

```
42087 #include <stdio.h>
42088 #include <time.h>
42089 #include <sys/stat.h>
42090 int
42091 print_file_time(const char *pathname)
42092 {
42093     struct stat statbuf;
42094     struct tm *tm;
42095     char timestr[BUFSIZ];
42096     if(stat(pathname, &statbuf) == -1)
42097         return -1;
42098     if((tm = localtime(&statbuf.st_mtime)) == NULL)
42099         return -1;
42100     if(strftime(timestr, sizeof(timestr), "%Y-%m-%d %H:%M:%S", tm) == 0)
42101         return -1;
42102     printf("%s: %s.%09ld\n", pathname, timestr, statbuf.st_mtim.tv_nsec);
42103     return 0;
42104 }
```

**Timing an Event**

The following example gets the current time, converts it to a string using *localtime()* and *asctime()*, and prints it to standard output using *fputs()*. It then prints the number of minutes to an event being timed.

```
#include <time.h>
#include <stdio.h>
...
time_t now;
int minutes_to_event;
...
time(&now);
printf("The time is ");
fputs(asctime(localtime(&now)), stdout);
printf("There are still %d minutes to the event.\n",
       minutes_to_event);
...
```

**APPLICATION USAGE**

The *localtime\_r()* function is thread-safe and returns values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*asctime()*, *clock()*, *ctime()*, *difftime()*, *getdate()*, *gmtime()*, *mktime()*, *strftime()*, *strptime()*, *time()*, *tzset()*, *utime()*

XBD Section 4.16 (on page 113), *<time.h>*

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

A note indicating that the *localtime()* function need not be reentrant is added to the DESCRIPTION.

The *localtime\_r()* function is included for alignment with the POSIX Threads Extension.

**Issue 6**

The *localtime\_r()* function is marked as part of the Thread-Safe Functions option.

Extensions beyond the ISO C standard are marked.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

The **restrict** keyword is added to the *localtime\_r()* prototype for alignment with the ISO/IEC 9899:1999 standard.

Examples are added.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/32 is applied, adding the [EOVERFLOW] error.

- 42148 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/55 is applied, updating the error handling  
42149 for *localtime\_r()*.
- 42150 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/56 is applied, adding a requirement that if  
42151 *localtime\_r()* does not set the *tzname* variable, it shall not set the *daylight* or *timezone* variables. On  
42152 systems supporting XSI, the *daylight*, *timezone*, and *tzname* variables should all be set to provide  
42153 information for the same timezone. This updates the description of *localtime\_r()* to mention  
42154 *daylight* and *timezone* as well as *tzname*. The SEE ALSO section is updated.
- 42155 **Issue 7**
- 42156 Austin Group Interpretation 1003.1-2001 #156 is applied.
- 42157 The *localtime\_r()* function is moved from the Thread-Safe Functions option to the Base.
- 42158 Changes are made to the EXAMPLES section related to support for finegrained timestamps.
- 42159 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0363 [291] is applied.



**NAME**

lockf — record locking on files

**SYNOPSIS**

```

#include <unistd.h>

int lockf(int fildes, int function, off_t size);

```

**DESCRIPTION**

The *lockf()* function shall lock sections of a file with advisory-mode locks. Calls to *lockf()* from threads in other processes which attempt to lock the locked file section shall either return an error value or block until the section becomes unlocked. All the locks for a process are removed when the process terminates. Record locking with *lockf()* shall be supported for regular files and may be supported for other files.

The *fildes* argument is an open file descriptor. To establish a lock with this function, the file descriptor shall be opened with write-only permission (O\_WRONLY) or with read/write permission (O\_RDWR).

The *function* argument is a control value which specifies the action to be taken. The permissible values for *function* are defined in **<unistd.h>** as follows:

| Function | Description                                  |
|----------|----------------------------------------------|
| F_ULOCK  | Unlock locked sections.                      |
| F_LOCK   | Lock a section for exclusive use.            |
| F_TLOCK  | Test and lock a section for exclusive use.   |
| F_TEST   | Test a section for locks by other processes. |

F\_TEST shall detect if a lock by another process is present on the specified section.

F\_LOCK and F\_TLOCK shall both lock a section of a file if the section is available.

F\_ULOCK shall remove locks from a section of the file.

The *size* argument is the number of contiguous bytes to be locked or unlocked. The section to be locked or unlocked starts at the current offset in the file and extends forward for a positive size or backward for a negative size (the preceding bytes up to but not including the current offset). If *size* is 0, the section from the current offset through the largest possible file offset shall be locked (that is, from the current offset through the present or any future end-of-file). An area need not be allocated to the file to be locked because locks may exist past the end-of-file.

The sections locked with F\_LOCK or F\_TLOCK may, in whole or in part, contain or be contained by a previously locked section for the same process. When this occurs, or if adjacent locked sections would occur, the sections shall be combined into a single locked section. If the request would cause the number of locks to exceed a system-imposed limit, the request shall fail.

F\_LOCK and F\_TLOCK requests differ only by the action taken if the section is not available. F\_LOCK shall block the calling thread until the section is available. F\_TLOCK shall cause the function to fail if the section is already locked by another process.

File locks shall be released on first close by the locking process of any file descriptor for the file.

F\_ULOCK requests may release (wholly or in part) one or more locked sections controlled by the process. Locked sections shall be unlocked starting at the current file offset through *size* bytes or to the end-of-file if *size* is (off\_t)0. When all of a locked section is not released (that is, when the beginning or end of the area to be unlocked falls within a locked section), the remaining portions of that section shall remain locked by the process. Releasing the center portion of a locked section shall cause the remaining locked beginning and end portions to become two separate

locked sections. If the request would cause the number of locks in the system to exceed a system-imposed limit, the request shall fail.

A potential for deadlock occurs if the threads of a process controlling a locked section are blocked by accessing a locked section of another process. If the system detects that deadlock would occur, *lockf()* shall fail with an [EDEADLK] error.

The interaction between *fcntl()* and *lockf()* locks is unspecified.

Blocking on a section shall be interrupted by any signal.

An F\_ULOCK request in which *size* is non-zero and the offset of the last byte of the requested section is the maximum value for an object of type **off\_t**, when the process has an existing lock in which *size* is 0 and which includes the last byte of the requested section, shall be treated as a request to unlock from the start of the requested section with a size equal to 0. Otherwise, an F\_ULOCK request shall attempt to unlock only the requested section.

Attempting to lock a section of a file that is associated with a buffered stream produces unspecified results.

#### RETURN VALUE

Upon successful completion, *lockf()* shall return 0. Otherwise, it shall return -1, set *errno* to indicate an error, and existing locks shall not be changed.

#### ERRORS

The *lockf()* function shall fail if:

[EBADF]            The *fildes* argument is not a valid open file descriptor; or *function* is F\_LOCK or F\_TLOCK and *fildes* is not a valid file descriptor open for writing.

[EACCES] or [EAGAIN]            The *function* argument is F\_TLOCK or F\_TEST and the section is already locked by another process.

[EDEADLK]            The *function* argument is F\_LOCK and a deadlock is detected.

[EINTR]            A signal was caught during execution of the function.

[EINVAL]            The *function* argument is not one of F\_LOCK, F\_TLOCK, F\_TEST, or F\_ULOCK; or *size* plus the current file offset is less than 0.

[EOVERFLOW]            The offset of the first, or if *size* is not 0 then the last, byte in the requested section cannot be represented correctly in an object of type **off\_t**.

The *lockf()* function may fail if:

[EAGAIN]            The *function* argument is F\_LOCK or F\_TLOCK and the file is mapped with *mmap()*.

[EDEADLK] or [ENOLCK]            The *function* argument is F\_LOCK, F\_TLOCK, or F\_ULOCK, and the request would cause the number of locks to exceed a system-imposed limit.

[EOPNOTSUPP] or [EINVAL]            The implementation does not support the locking of files of the type indicated by the *fildes* argument.

42243 **EXAMPLES**42244 **Locking a Portion of a File**

42245 In the following example, a file named `/home/cnd/mod1` is being modified. Other processes that  
 42246 use locking are prevented from changing it during this process. Only the first 10 000 bytes are  
 42247 locked, and the lock call fails if another process has any part of this area locked already.

```
42248 #include <fcntl.h>
42249 #include <unistd.h>

42250 int fildes;
42251 int status;
42252 ...
42253 fildes = open("/home/cnd/mod1", O_RDWR);
42254 status = lockf(fildes, F_TLOCK, (off_t)10000);
```

42255 **APPLICATION USAGE**

42256 Record-locking should not be used in combination with the *fopen()*, *fread()*, *fwrite()*, and other  
 42257 *stdio* functions. Instead, the more primitive, non-buffered functions (such as *open()*) should be  
 42258 used. Unexpected results may occur in processes that do buffering in the user address space. The  
 42259 process may later read/write data which is/was locked. The *stdio* functions are the most  
 42260 common source of unexpected buffering.

42261 The *alarm()* function may be used to provide a timeout facility in applications requiring it.

42262 **RATIONALE**

42263 None.

42264 **FUTURE DIRECTIONS**

42265 None.

42266 **SEE ALSO**

42267 *alarm()*, *chmod()*, *close()*, *creat()*, *fcntl()*, *fopen()*, *mmap()*, *open()*, *read()*, *write()*

42268 XBD **<unistd.h>**

42269 **CHANGE HISTORY**

42270 First released in Issue 4, Version 2.

42271 **Issue 5**

42272 Moved from X/OPEN UNIX extension to BASE.

42273 Large File Summit extensions are added. In particular, the description of [EINVAL] is clarified  
 42274 and moved from optional to mandatory status.

42275 A note is added to the DESCRIPTION indicating the effects of attempting to lock a section of a  
 42276 file that is associated with a buffered stream.

42277 **Issue 6**

42278 The normative text is updated to avoid use of the term “must” for application requirements.

42279 **Issue 7**

42280 Austin Group Interpretation 1003.1-2001 #054 is applied, updating the DESCRIPTION.

42281 **NAME**

42282 log, logf, logl — natural logarithm function

42283 **SYNOPSIS**

```
42284 #include <math.h>
42285 double log(double x);
42286 float logf(float x);
42287 long double logl(long double x);
```

42288 **DESCRIPTION**

42289 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 42290 conflict between the requirements described here and the ISO C standard is unintentional. This  
 42291 volume of POSIX.1-2008 defers to the ISO C standard.

42292 These functions shall compute the natural logarithm of their argument  $x$ ,  $\log_e(x)$ .

42293 An application wishing to check for error situations should set *errno* to zero and call  
 42294 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 42295 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 42296 zero, an error has occurred.

42297 **RETURN VALUE**

42298 Upon successful completion, these functions shall return the natural logarithm of  $x$ .

42299 If  $x$  is  $\pm 0$ , a pole error shall occur and *log()*, *logf()*, and *logl()* shall return  $-\text{HUGE\_VAL}$ ,  
 42300  $-\text{HUGE\_VALF}$ , and  $-\text{HUGE\_VALL}$ , respectively.

42301 MX For finite values of  $x$  that are less than 0, or if  $x$  is  $-\text{Inf}$ , a domain error shall occur, and either a  
 42302 NaN (if supported), or an implementation-defined value shall be returned.

42303 MX If  $x$  is NaN, a NaN shall be returned.

42304 If  $x$  is 1, +0 shall be returned.

42305 If  $x$  is  $+\text{Inf}$ ,  $x$  shall be returned.

42306 **ERRORS**

42307 These functions shall fail if:

|       |    |              |                                                                                               |
|-------|----|--------------|-----------------------------------------------------------------------------------------------|
| 42308 | MX | Domain Error | The finite value of $x$ is negative, or $x$ is $-\text{Inf}$ .                                |
| 42309 |    |              | If the integer expression ( <i>math_errhandling</i> & MATH_ERRNO) is non-zero,                |
| 42310 |    |              | then <i>errno</i> shall be set to [EDOM]. If the integer expression ( <i>math_errhandling</i> |
| 42311 |    |              | & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception                      |
| 42312 |    |              | shall be raised.                                                                              |

|       |  |            |                           |
|-------|--|------------|---------------------------|
| 42313 |  | Pole Error | The value of $x$ is zero. |
|-------|--|------------|---------------------------|

|       |  |  |                                                                                  |
|-------|--|--|----------------------------------------------------------------------------------|
| 42314 |  |  | If the integer expression ( <i>math_errhandling</i> & MATH_ERRNO) is non-zero,   |
| 42315 |  |  | then <i>errno</i> shall be set to [ERANGE]. If the integer expression            |
| 42316 |  |  | ( <i>math_errhandling</i> & MATH_ERREXCEPT) is non-zero, then the divide-by-zero |
| 42317 |  |  | floating-point exception shall be raised.                                        |

42318 **EXAMPLES**

42319 None.

42320 **APPLICATION USAGE**

42321 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 42322 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

42323 **RATIONALE**

42324 None.

42325 **FUTURE DIRECTIONS**

42326 None.

42327 **SEE ALSO**42328 *exp()*, *feclearexcept()*, *fetestexcept()*, *isnan()*, *log10()*, *log1p()*

42329 XBD Section 4.20 (on page 117), &lt;math.h&gt;

42330 **CHANGE HISTORY**

42331 First released in Issue 1. Derived from Issue 1 of the SVID.

42332 **Issue 5**

42333 The DESCRIPTION is updated to indicate how an application should check for an error. This  
 42334 text was previously published in the APPLICATION USAGE section.

42335 **Issue 6**

42336 The normative text is updated to avoid use of the term “must” for application requirements.

42337 The *logf()* and *logl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

42338 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
 42339 revised to align with the ISO/IEC 9899:1999 standard.

42340 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
 42341 marked.

42342 **NAME**

42343 log10, log10f, log10l — base 10 logarithm function

42344 **SYNOPSIS**

```
42345 #include <math.h>
42346 double log10(double x);
42347 float log10f(float x);
42348 long double log10l(long double x);
```

42349 **DESCRIPTION**

42350 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 42351 conflict between the requirements described here and the ISO C standard is unintentional. This  
 42352 volume of POSIX.1-2008 defers to the ISO C standard.

42353 These functions shall compute the base 10 logarithm of their argument  $x$ ,  $\log_{10}(x)$ .

42354 An application wishing to check for error situations should set *errno* to zero and call  
 42355 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 42356 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 42357 zero, an error has occurred.

42358 **RETURN VALUE**

42359 Upon successful completion, these functions shall return the base 10 logarithm of  $x$ .

42360 If  $x$  is  $\pm 0$ , a pole error shall occur and *log10()*, *log10f()*, and *log10l()* shall return  $-\text{HUGE\_VAL}$ ,  
 42361  $-\text{HUGE\_VALF}$ , and  $-\text{HUGE\_VALL}$ , respectively.

42362 MX For finite values of  $x$  that are less than 0, or if  $x$  is  $-\text{Inf}$ , a domain error shall occur, and either a  
 42363 NaN (if supported), or an implementation-defined value shall be returned.

42364 MX If  $x$  is NaN, a NaN shall be returned.

42365 If  $x$  is 1, +0 shall be returned.

42366 If  $x$  is  $+\text{Inf}$ ,  $+\text{Inf}$  shall be returned.

42367 **ERRORS**

42368 These functions shall fail if:

42369 MX Domain Error The finite value of  $x$  is negative, or  $x$  is  $-\text{Inf}$ .

42370 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 42371 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 42372 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 42373 shall be raised.

42374 Pole Error The value of  $x$  is zero.

42375 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 42376 then *errno* shall be set to [ERANGE]. If the integer expression  
 42377 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the divide-by-zero  
 42378 floating-point exception shall be raised.

42379 **EXAMPLES**

42380 None.

42381 **APPLICATION USAGE**

42382 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 42383 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

42384 **RATIONALE**

42385 None.

42386 **FUTURE DIRECTIONS**

42387 None.

42388 **SEE ALSO**42389 *feclearexcept()*, *fetestexcept()*, *isnan()*, *log()*, *pow()*42390 XBD Section 4.20 (on page 117), **<math.h>**42391 **CHANGE HISTORY**

42392 First released in Issue 1. Derived from Issue 1 of the SVID.

42393 **Issue 5**

42394 The DESCRIPTION is updated to indicate how an application should check for an error. This  
 42395 text was previously published in the APPLICATION USAGE section.

42396 **Issue 6**

42397 The normative text is updated to avoid use of the term “must” for application requirements.

42398 The *log10f()* and *log10l()* functions are added for alignment with the ISO/IEC 9899:1999  
 42399 standard.

42400 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
 42401 revised to align with the ISO/IEC 9899:1999 standard.

42402 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
 42403 marked.



42404 **NAME**

42405 log1p, log1pf, log1pl — compute a natural logarithm

42406 **SYNOPSIS**

```
42407 #include <math.h>
42408 double log1p(double x);
42409 float log1pf(float x);
42410 long double log1pl(long double x);
```

42411 **DESCRIPTION**

42412 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 42413 conflict between the requirements described here and the ISO C standard is unintentional. This  
 42414 volume of POSIX.1-2008 defers to the ISO C standard.

42415 These functions shall compute  $\log_e(1.0 + x)$ .

42416 An application wishing to check for error situations should set *errno* to zero and call  
 42417 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 42418 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 42419 zero, an error has occurred.

42420 **RETURN VALUE**42421 Upon successful completion, these functions shall return the natural logarithm of  $1.0 + x$ .

42422 If  $x$  is  $-1$ , a pole error shall occur and *log1p()*, *log1pf()*, and *log1pl()* shall return  $-\text{HUGE\_VAL}$ ,  
 42423  $-\text{HUGE\_VALF}$ , and  $-\text{HUGE\_VALL}$ , respectively.

42424 MX For finite values of  $x$  that are less than  $-1$ , or if  $x$  is  $-\text{Inf}$ , a domain error shall occur, and either a  
 42425 NaN (if supported), or an implementation-defined value shall be returned.

42426 MX If  $x$  is NaN, a NaN shall be returned.42427 If  $x$  is  $\pm 0$ , or  $+\text{Inf}$ ,  $x$  shall be returned.42428 If  $x$  is subnormal, a range error may occur42429 MXX and  $x$  should be returned.

42430 MX If  $x$  is not returned, *log1p()*, *log1pf()*, and *log1pl()* shall return an implementation-defined value  
 42431 no greater in magnitude than DBL\_MIN, FLT\_MIN, and LDBL\_MIN, respectively.

42432 **ERRORS**

42433 These functions shall fail if:

42434 MX Domain Error The finite value of  $x$  is less than  $-1$ , or  $x$  is  $-\text{Inf}$ .

42435 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 42436 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 42437 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 42438 shall be raised.

42439 Pole Error The value of  $x$  is  $-1$ .

42440 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 42441 then *errno* shall be set to [ERANGE]. If the integer expression  
 42442 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the divide-by-zero  
 42443 floating-point exception shall be raised.



42444 These functions may fail if:

42445 MX **Range Error** The value of  $x$  is subnormal.

42446 If the integer expression  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$  is non-zero,  
 42447 then *errno* shall be set to [ERANGE]. If the integer expression  
 42448  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$  is non-zero, then the underflow  
 42449 floating-point exception shall be raised.

#### 42450 EXAMPLES

42451 None.

#### 42452 APPLICATION USAGE

42453 On error, the expressions  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$  and  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$  are independent of each other, but at least one of them must be non-zero.

#### 42455 RATIONALE

42456 None.

#### 42457 FUTURE DIRECTIONS

42458 None.

#### 42459 SEE ALSO

42460 *feclearexcept()*, *fetestexcept()*, *log()*

42461 XBD Section 4.20 (on page 117), **<math.h>**

#### 42462 CHANGE HISTORY

42463 First released in Issue 4, Version 2.

#### 42464 Issue 5

42465 Moved from X/OPEN UNIX extension to BASE.

#### 42466 Issue 6

42467 The normative text is updated to avoid use of the term “must” for application requirements.

42468 The *log1p()* function is no longer marked as an extension.

42469 The *log1pf()* and *log1pl()* functions are added for alignment with the ISO/IEC 9899:1999  
 42470 standard.

42471 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
 42472 revised to align with the ISO/IEC 9899:1999 standard.

42473 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
 42474 marked.

#### 42475 Issue 7

42476 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0364 [68] is applied.

42477 **NAME**

42478 log2, log2f, log2l — compute base 2 logarithm functions

42479 **SYNOPSIS**

```
42480 #include <math.h>
42481 double log2(double x);
42482 float log2f(float x);
42483 long double log2l(long double x);
```

42484 **DESCRIPTION**

42485 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 42486 conflict between the requirements described here and the ISO C standard is unintentional. This  
 42487 volume of POSIX.1-2008 defers to the ISO C standard.

42488 These functions shall compute the base 2 logarithm of their argument  $x$ ,  $\log_2(x)$ .

42489 An application wishing to check for error situations should set *errno* to zero and call  
 42490 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 42491 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 42492 zero, an error has occurred.

42493 **RETURN VALUE**

42494 Upon successful completion, these functions shall return the base 2 logarithm of  $x$ .

42495 If  $x$  is  $\pm 0$ , a pole error shall occur and *log2()*, *log2f()*, and *log2l()* shall return  $-\text{HUGE\_VAL}$ ,  
 42496  $-\text{HUGE\_VALF}$ , and  $-\text{HUGE\_VALL}$ , respectively.

42497 MX For finite values of  $x$  that are less than 0, or if  $x$  is  $-\text{Inf}$ , a domain error shall occur, and either a  
 42498 NaN (if supported), or an implementation-defined value shall be returned.

42499 MX If  $x$  is NaN, a NaN shall be returned.

42500 If  $x$  is 1, +0 shall be returned.

42501 If  $x$  is  $+\text{Inf}$ ,  $x$  shall be returned.

42502 **ERRORS**

42503 These functions shall fail if:

42504 MX Domain Error The finite value of  $x$  is less than zero, or  $x$  is  $-\text{Inf}$ .

42505 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 42506 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 42507 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 42508 shall be raised.

42509 Pole Error The value of  $x$  is zero.

42510 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 42511 then *errno* shall be set to [ERANGE]. If the integer expression  
 42512 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the divide-by-zero  
 42513 floating-point exception shall be raised.

42514 **EXAMPLES**

42515       None.

42516 **APPLICATION USAGE**

42517       On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
42518       MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

42519 **RATIONALE**

42520       None.

42521 **FUTURE DIRECTIONS**

42522       None.

42523 **SEE ALSO**42524       *feclearexcept()*, *fetestexcept()*, *log()*42525       XBD Section 4.20 (on page 117), **<math.h>**42526 **CHANGE HISTORY**

42527       First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

42528 **NAME**

42529 logb, logbf, logbl — radix-independent exponent

42530 **SYNOPSIS**

```
42531 #include <math.h>
42532 double logb(double x);
42533 float logbf(float x);
42534 long double logbl(long double x);
```

42535 **DESCRIPTION**

42536 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 42537 conflict between the requirements described here and the ISO C standard is unintentional. This  
 42538 volume of POSIX.1-2008 defers to the ISO C standard.

42539 These functions shall compute the exponent of  $x$ , which is the integral part of  $\log_r |x|$ , as a  
 42540 signed floating-point value, for non-zero  $x$ , where  $r$  is the radix of the machine's floating-point  
 42541 arithmetic, which is the value of FLT\_RADIX defined in the **<float.h>** header.

42542 If  $x$  is subnormal it is treated as though it were normalized; thus for finite positive  $x$ :

42543  $1 \leq x * \text{FLT\_RADIX}^{-\log_b(x)} < \text{FLT\_RADIX}$

42544 An application wishing to check for error situations should set *errno* to zero and call  
 42545 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 42546 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 42547 zero, an error has occurred.

42548 **RETURN VALUE**

42549 Upon successful completion, these functions shall return the exponent of  $x$ .

42550 If  $x$  is  $\pm 0$ , *logb()*, *logbf()*, and *logbl()* shall return  $-\text{HUGE\_VAL}$ ,  $-\text{HUGE\_VALF}$ , and  
 42551  $-\text{HUGE\_VALL}$ , respectively.

42552 MX On systems that support the IEC 60559 Floating-Point option, a pole error shall occur;

42553 CX otherwise, a pole error may occur.

42554 MX If  $x$  is NaN, a NaN shall be returned.

42555 MX If  $x$  is  $\pm\text{Inf}$ ,  $+\text{Inf}$  shall be returned.

42556 **ERRORS**

42557 These functions shall fail if:

42558 MX **Pole Error** The value of  $x$  is  $\pm 0$ .

42559 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 42560 then *errno* shall be set to [ERANGE]. If the integer expression  
 42561 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the divide-by-zero  
 42562 floating-point exception shall be raised.

42563 These functions may fail if:

42564 **Pole Error** The value of  $x$  is 0.

42565 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 42566 then *errno* shall be set to [ERANGE]. If the integer expression  
 42567 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the divide-by-zero  
 42568 floating-point exception shall be raised.

42569 **EXAMPLES**

42570 None.

42571 **APPLICATION USAGE**

42572 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 42573 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

42574 **RATIONALE**

42575 None.

42576 **FUTURE DIRECTIONS**

42577 None.

42578 **SEE ALSO**42579 *feclearexcept()*, *fetestexcept()*, *ilogb()*, *scalbln()*42580 XBD Section 4.20 (on page 117), **<float.h>**, **<math.h>**42581 **CHANGE HISTORY**

42582 First released in Issue 4, Version 2.

42583 **Issue 5**

42584 Moved from X/OPEN UNIX extension to BASE.

42585 **Issue 6**42586 The *logb()* function is no longer marked as an extension.42587 The *logbf()* and *logbl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

42588 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
 42589 revised to align with the ISO/IEC 9899:1999 standard.

42590 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
 42591 marked.

42592 **Issue 7**

42593 ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #50 (SD5-XSH-ERN-76) is applied.

42594 **NAME**

42595       logf, logl — natural logarithm function

42596 **SYNOPSIS**

42597       #include &lt;math.h&gt;

42598       float logf(float x);

42599       long double logl(long double x);

42600 **DESCRIPTION**42601       Refer to *log()*.

42602 **NAME**

42603 longjmp — non-local goto

42604 **SYNOPSIS**

42605 #include &lt;setjmp.h&gt;

42606 void longjmp(jmp\_buf env, int val);

42607 **DESCRIPTION**

42608 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 42609 conflict between the requirements described here and the ISO C standard is unintentional. This  
 42610 volume of POSIX.1-2008 defers to the ISO C standard.

42611 The *longjmp()* function shall restore the environment saved by the most recent invocation of  
 42612 *setjmp()* in the same process, with the corresponding **jmp\_buf** argument. If the most recent  
 42613 invocation of *setjmp()* with the corresponding **jmp\_buf** occurred in another thread, or if there is  
 42614 no such invocation, or if the function containing the invocation of *setjmp()* has terminated  
 42615 execution in the interim, or if the invocation of *setjmp()* was within the scope of an identifier  
 42616 with variably modified type and execution has left that scope in the interim, the behavior is  
 42617 CX undefined. It is unspecified whether *longjmp()* restores the signal mask, leaves the signal mask  
 42618 unchanged, or restores it to its value at the time *setjmp()* was called.

42619 All accessible objects have values, and all other components of the abstract machine have state  
 42620 (for example, floating-point status flags and open files), as of the time *longjmp()* was called,  
 42621 except that the values of objects of automatic storage duration are unspecified if they meet all  
 42622 the following conditions:

- 42623 • They are local to the function containing the corresponding *setjmp()* invocation.
- 42624 • They do not have volatile-qualified type.
- 42625 • They are changed between the *setjmp()* invocation and *longjmp()* call.

42626 CX Although *longjmp()* is an async-signal-safe function, if it is invoked from a signal handler which  
 42627 interrupted a non-async-signal-safe function or equivalent (such as the processing equivalent to  
 42628 *exit()* performed after a return from the initial call to *main()*), the behavior of any subsequent  
 42629 call to a non-async-signal-safe function or equivalent is undefined.

42630 The effect of a call to *longjmp()* where initialization of the **jmp\_buf** structure was not performed  
 42631 in the calling thread is undefined.

42632 **RETURN VALUE**

42633 After *longjmp()* is completed, program execution continues as if the corresponding invocation of  
 42634 *setjmp()* had just returned the value specified by *val*. The *longjmp()* function shall not cause  
 42635 *setjmp()* to return 0; if *val* is 0, *setjmp()* shall return 1.

42636 **ERRORS**

42637 No errors are defined.

42638 **EXAMPLES**

42639 None.

42640 **APPLICATION USAGE**

42641 Applications whose behavior depends on the value of the signal mask should not use *longjmp()*  
 42642 and *setjmp()*, since their effect on the signal mask is unspecified, but should instead use the  
 42643 *siglongjmp()* and *sigsetjmp()* functions (which can save and restore the signal mask under  
 42644 application control).

42645 It is recommended that applications do not call *longjmp()* or *siglongjmp()* from signal handlers. +  
 42646 To avoid undefined behavior when calling these functions from a signal handler, the application +

|       |                                                                                                                    |   |
|-------|--------------------------------------------------------------------------------------------------------------------|---|
| 42647 | needs to ensure one of the following two things:                                                                   | + |
| 42648 | 1. After the call to <i>longjmp()</i> or <i>siglongjmp()</i> the process only calls async-signal-safe              | + |
| 42649 | functions and does not return from the initial call to <i>main()</i> .                                             | + |
| 42650 | 2. Any signal whose handler calls <i>longjmp()</i> or <i>siglongjmp()</i> is blocked during <i>every</i> call to a | + |
| 42651 | non-async-signal-safe function, and no such calls are made after returning from the initial                        | + |
| 42652 | call to <i>main()</i> .                                                                                            | + |
| 42653 | <b>RATIONALE</b>                                                                                                   |   |
| 42654 | None.                                                                                                              |   |
| 42655 | <b>FUTURE DIRECTIONS</b>                                                                                           |   |
| 42656 | None.                                                                                                              |   |
| 42657 | <b>SEE ALSO</b>                                                                                                    |   |
| 42658 | <i>setjmp()</i> , <i>sigaction()</i> , <i>siglongjmp()</i> , <i>sigsetjmp()</i>                                    |   |
| 42659 | XBD <setjmp.h>                                                                                                     |   |
| 42660 | <b>CHANGE HISTORY</b>                                                                                              |   |
| 42661 | First released in Issue 1. Derived from Issue 1 of the SVID.                                                       |   |
| 42662 | <b>Issue 5</b>                                                                                                     |   |
| 42663 | The DESCRIPTION is updated for alignment with the POSIX Threads Extension.                                         |   |
| 42664 | <b>Issue 6</b>                                                                                                     |   |
| 42665 | Extensions beyond the ISO C standard are marked.                                                                   |   |
| 42666 | The following new requirements on POSIX implementations derive from alignment with the                             |   |
| 42667 | Single UNIX Specification:                                                                                         |   |
| 42668 | • The DESCRIPTION now explicitly makes <i>longjmp()</i> 's effect on the signal mask                               |   |
| 42669 | unspecified.                                                                                                       |   |
| 42670 | The DESCRIPTION is updated for alignment with the ISO/IEC 9899:1999 standard.                                      |   |
| 42671 | <b>Issue 7</b>                                                                                                     |   |
| 42672 | POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0365 [394] is applied.                                         |   |



42673 **NAME**

42674 lrand48 — generate uniformly distributed pseudo-random non-negative long integers

42675 **SYNOPSIS**

```
42676 XSI      #include <stdlib.h>  
42677          long lrand48(void);
```

42678 **DESCRIPTION**42679 Refer to *drand48()*.

42680 **NAME**

42681 lrint, lrintf, lrintl — round to nearest integer value using current rounding direction

42682 **SYNOPSIS**

```
42683 #include <math.h>
42684 long lrint(double x);
42685 long lrintf(float x);
42686 long lrintl(long double x);
```

42687 **DESCRIPTION**

42688 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 42689 conflict between the requirements described here and the ISO C standard is unintentional. This  
 42690 volume of POSIX.1-2008 defers to the ISO C standard.

42691 These functions shall round their argument to the nearest integer value, rounding according to  
 42692 the current rounding direction.

42693 An application wishing to check for error situations should set *errno* to zero and call  
 42694 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 42695 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 42696 zero, an error has occurred.

42697 **RETURN VALUE**

42698 Upon successful completion, these functions shall return the rounded integer value.

42699 MX If *x* is NaN, a domain error shall occur and an unspecified value is returned.

42700 If *x* is +Inf, a domain error shall occur and an unspecified value is returned.

42701 If *x* is -Inf, a domain error shall occur and an unspecified value is returned.

42702 If the correct value is positive and too large to represent as a **long**, an unspecified value shall be  
 42703 MX returned. On systems that support the IEC 60559 Floating-Point option, a domain error shall  
 42704 CX occur; otherwise, a domain error may occur.

42705 If the correct value is negative and too large to represent as a **long**, an unspecified value shall be  
 42706 MX returned. On systems that support the IEC 60559 Floating-Point option, a domain error shall  
 42707 CX occur; otherwise, a domain error may occur.

42708 **ERRORS**

42709 These functions shall fail if:

42710 MX **Domain Error** The *x* argument is NaN or ±Inf, or the correct value is not representable as an  
 42711 integer.

42712 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 42713 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 42714 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 42715 shall be raised.

42716 These functions may fail if:

42717 **Domain Error** The correct value is not representable as an integer.

42718 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 42719 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 42720 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 42721 shall be raised.

**42722 EXAMPLES**

42723 None.

**42724 APPLICATION USAGE**

42725 On error, the expressions *(math\_errhandling & MATH\_ERRNO)* and *(math\_errhandling &*  
42726 *MATH\_ERREXCEPT)* are independent of each other, but at least one of them must be non-zero.

**42727 RATIONALE**

42728 These functions provide floating-to-integer conversions. They round according to the current  
42729 rounding direction. If the rounded value is outside the range of the return type, the numeric  
42730 result is unspecified and the invalid floating-point exception is raised. When they raise no other  
42731 floating-point exception and the result differs from the argument, they raise the inexact floating-  
42732 point exception.

**42733 FUTURE DIRECTIONS**

42734 None.

**42735 SEE ALSO**

42736 *feclearexcept()*, *fetestexcept()*, *llrint()*

42737 XBD Section 4.20 (on page 117), *<math.h>*

**42738 CHANGE HISTORY**

42739 First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

**42740 Issue 7**

42741 ISO/IEC 9899: 1999 standard, Technical Corrigendum 2 #53 (SD5-XSH-ERN-77) is applied.

42742 **NAME**

42743 lround, lroundf, lroundl — round to nearest integer value

42744 **SYNOPSIS**

```
42745 #include <math.h>
42746 long lround(double x);
42747 long lroundf(float x);
42748 long lroundl(long double x);
```

42749 **DESCRIPTION**

42750 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 42751 conflict between the requirements described here and the ISO C standard is unintentional. This  
 42752 volume of POSIX.1-2008 defers to the ISO C standard.

42753 These functions shall round their argument to the nearest integer value, rounding halfway cases  
 42754 away from zero, regardless of the current rounding direction.

42755 An application wishing to check for error situations should set *errno* to zero and call  
 42756 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 42757 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 42758 zero, an error has occurred.

42759 **RETURN VALUE**

42760 Upon successful completion, these functions shall return the rounded integer value.

42761 MX If *x* is NaN, a domain error shall occur and an unspecified value is returned.

42762 If *x* is +Inf, a domain error shall occur and an unspecified value is returned.

42763 If *x* is -Inf, a domain error shall occur and an unspecified value is returned.

42764 If the correct value is positive and too large to represent as a **long**, an unspecified value shall be  
 42765 MX returned. On systems that support the IEC 60559 Floating-Point option, a domain shall occur;  
 42766 CX otherwise, a domain error may occur.

42767 If the correct value is negative and too large to represent as a **long**, an unspecified value shall be  
 42768 MX returned. On systems that support the IEC 60559 Floating-Point option, a domain shall occur;  
 42769 CX otherwise, a domain error may occur.

42770 **ERRORS**

42771 These functions shall fail if:

42772 MX **Domain Error** The *x* argument is NaN or  $\pm\text{Inf}$ , or the correct value is not representable as an  
 42773 integer.

42774 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 42775 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 42776 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 42777 shall be raised.

42778 These functions may fail if:

42779 **Domain Error** The correct value is not representable as an integer.

42780 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 42781 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 42782 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 42783 shall be raised.

42784 **EXAMPLES**

42785 None.

42786 **APPLICATION USAGE**

42787 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 42788 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

42789 **RATIONALE**

42790 These functions differ from the *lrint()* functions in the default rounding direction, with the  
 42791 *lround()* functions rounding halfway cases away from zero and needing not to raise the inexact  
 42792 floating-point exception for non-integer arguments that round to within the range of the return  
 42793 type.

42794 **FUTURE DIRECTIONS**

42795 None.

42796 **SEE ALSO**42797 *feclearexcept()*, *fetestexcept()*, *llround()*

42798 XBD Section 4.20 (on page 117), &lt;math.h&gt;

42799 **CHANGE HISTORY**

42800 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

42801 **Issue 7**

42802 ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #54 (SD5-XSH-ERN-78) is applied.

42803 **NAME**

42804 lsearch, lfind — linear search and update

42805 **SYNOPSIS**

```

42806 XSI #include <search.h>
42807 void *lsearch(const void *key, void *base, size_t *nel, size_t width,
42808             int (*compar)(const void *, const void *));
42809 void *lfind(const void *key, const void *base, size_t *nel,
42810            size_t width, int (*compar)(const void *, const void *));

```

42811 **DESCRIPTION**

42812 The *lsearch()* function shall linearly search the table and return a pointer into the table for the  
 42813 matching entry. If the entry does not occur, it shall be added at the end of the table. The *key*  
 42814 argument points to the entry to be sought in the table. The *base* argument points to the first  
 42815 element in the table. The *width* argument is the size of an element in bytes. The *nel* argument  
 42816 points to an integer containing the current number of elements in the table. The integer to which  
 42817 *nel* points shall be incremented if the entry is added to the table. The *compar* argument points to  
 42818 a comparison function which the application shall supply (for example, *strcmp()*). It is called  
 42819 with two arguments that point to the elements being compared. The application shall ensure  
 42820 that the function returns 0 if the elements are equal, and non-zero otherwise.

42821 The *lfind()* function shall be equivalent to *lsearch()*, except that if the entry is not found, it is not  
 42822 added to the table. Instead, a null pointer is returned.

42823 **RETURN VALUE**

42824 If the searched for entry is found, both *lsearch()* and *lfind()* shall return a pointer to it.  
 42825 Otherwise, *lfind()* shall return a null pointer and *lsearch()* shall return a pointer to the newly  
 42826 added element.

42827 Both functions shall return a null pointer in case of error.

42828 **ERRORS**

42829 No errors are defined.

42830 **EXAMPLES**42831 **Storing Strings in a Table**

42832 This fragment reads in less than or equal to TABSIZE strings of length less than or equal to  
 42833 ELSIZE and stores them in a table, eliminating duplicates.

```

42834 #include <stdio.h>
42835 #include <string.h>
42836 #include <search.h>
42837 #define TABSIZE 50
42838 #define ELSIZE 120
42839 ...
42840     char line[ELSIZE], tab[TABSIZE][ELSIZE];
42841     size_t nel = 0;
42842     ...
42843     while (fgets(line, ELSIZE, stdin) != NULL && nel < TABSIZE)
42844         (void) lsearch(line, tab, &nel,
42845                       ELSIZE, (int (*)(const void *, const void *)) strcmp);
42846     ...

```

**Finding a Matching Entry**

The following example finds any line that reads "This is a test.".

```
#include <search.h>
#include <string.h>
...
char line[ELSIZE], tab[TABSIZE][ELSIZE];
size_t nel = 0;
char *findline;
void *entry;

findline = "This is a test.\n";

entry = lfind(findline, tab, &nel, ELSIZE, (
    int (*)(const void *, const void *)) strcmp);
```

**APPLICATION USAGE**

The comparison function need not compare every byte, so arbitrary data may be contained in the elements in addition to the values being compared.

Undefined results can occur if there is not enough room in the table to add a new item.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*hcreate(), tdelete()*

XBD [<search.h>](#)

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 6**

The normative text is updated to avoid use of the term “must” for application requirements.

42874 **NAME**

42875 lseek — move the read/write file offset

42876 **SYNOPSIS**

42877 #include &lt;unistd.h&gt;

42878 off\_t lseek(int *fildev*, off\_t *offset*, int *whence*);42879 **DESCRIPTION**42880 The *lseek()* function shall set the file offset for the open file description associated with the file  
42881 descriptor *fildev*, as follows:

- 42882 • If *whence* is SEEK\_SET, the file offset shall be set to *offset* bytes.
- 42883 • If *whence* is SEEK\_CUR, the file offset shall be set to its current location plus *offset*.
- 42884 • If *whence* is SEEK\_END, the file offset shall be set to the size of the file plus *offset*.

42885 The symbolic constants SEEK\_SET, SEEK\_CUR, and SEEK\_END are defined in &lt;unistd.h&gt;.

42886 The behavior of *lseek()* on devices which are incapable of seeking is implementation-defined.  
42887 The value of the file offset associated with such a device is undefined.42888 The *lseek()* function shall allow the file offset to be set beyond the end of the existing data in the  
42889 file. If data is later written at this point, subsequent reads of data in the gap shall return bytes  
42890 with the value 0 until data is actually written into the gap.42891 The *lseek()* function shall not, by itself, extend the size of a file.42892 SHM If *fildev* refers to a shared memory object, the result of the *lseek()* function is unspecified.42893 TYM If *fildev* refers to a typed memory object, the result of the *lseek()* function is unspecified.42894 **RETURN VALUE**42895 Upon successful completion, the resulting offset, as measured in bytes from the beginning of the  
42896 file, shall be returned. Otherwise, -1 shall be returned, *errno* shall be set to indicate the error, and  
42897 the file offset shall remain unchanged.42898 **ERRORS**42899 The *lseek()* function shall fail if:

- |       |             |                                                                                                                                                        |
|-------|-------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| 42900 | [EBADF]     | The <i>fildev</i> argument is not an open file descriptor.                                                                                             |
| 42901 | [EINVAL]    | The <i>whence</i> argument is not a proper value, or the resulting file offset would be negative for a regular file, block special file, or directory. |
| 42902 |             |                                                                                                                                                        |
| 42903 | [EOVERFLOW] | The resulting file offset would be a value which cannot be represented correctly in an object of type <b>off_t</b> .                                   |
| 42904 |             |                                                                                                                                                        |
| 42905 | [ESPIPE]    | The <i>fildev</i> argument is associated with a pipe, FIFO, or socket.                                                                                 |

42906 **EXAMPLES**

42907 None.

42908 **APPLICATION USAGE**

42909 None.

42910 **RATIONALE**42911 The ISO C standard includes the functions *fgetpos()* and *fsetpos()*, which work on very large files  
42912 by use of a special positioning type.42913 Although *lseek()* may position the file offset beyond the end of the file, this function does not  
42914 itself extend the size of the file. While the only function in POSIX.1-2008 that may directly extend



the size of the file is *write()*, *truncate()*, and *ftruncate()*, several functions originally derived from the ISO C standard, such as *fwrite()*, *fprintf()*, and so on, may do so (by causing calls on *write()*).

An invalid file offset that would cause [EINVAL] to be returned may be both implementation-defined and device-dependent (for example, memory may have few invalid values). A negative file offset may be valid for some devices in some implementations.

The POSIX.1-1990 standard did not specifically prohibit *lseek()* from returning a negative offset. Therefore, an application was required to clear *errno* prior to the call and check *errno* upon return to determine whether a return value of (**off\_t**)-1 is a negative offset or an indication of an error condition. The standard developers did not wish to require this action on the part of a conforming application, and chose to require that *errno* be set to [EINVAL] when the resulting file offset would be negative for a regular file, block special file, or directory.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*open()*

XBD [`<sys/types.h>`](#), [`<unistd.h>`](#)

#### CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

##### Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension.

Large File Summit extensions are added.

##### Issue 6

In the SYNOPSIS, the optional include of the [`<sys/types.h>`](#) header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include [`<sys/types.h>`](#) has been removed. Although [`<sys/types.h>`](#) was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The [EOVERFLOW] error condition is added. This change is to support large files.

An additional [ESPIPE] error condition is added for sockets.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that *lseek()* results are unspecified for typed memory objects.

##### Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0366 [421] is applied.

42949 **NAME**

42950       lstat — get file status

42951 **SYNOPSIS**

42952       #include &lt;sys/stat.h&gt;

42953       int lstat(const char \*restrict *path*, struct stat \*restrict *buf*);42954 **DESCRIPTION**42955       Refer to *fstatat()*.

42956 **NAME**

42957        malloc — a memory allocator

42958 **SYNOPSIS**

42959        #include &lt;stdlib.h&gt;

42960        void \*malloc(size\_t size);

42961 **DESCRIPTION**

42962 CX        The functionality described on this reference page is aligned with the ISO C standard. Any  
 42963            conflict between the requirements described here and the ISO C standard is unintentional. This  
 42964            volume of POSIX.1-2008 defers to the ISO C standard.

42965        The *malloc()* function shall allocate unused space for an object whose size in bytes is specified by  
 42966        *size* and whose value is unspecified.

42967        The order and contiguity of storage allocated by successive calls to *malloc()* is unspecified. The  
 42968        pointer returned if the allocation succeeds shall be suitably aligned so that it may be assigned to  
 42969        a pointer to any type of object and then used to access such an object in the space allocated (until  
 42970        the space is explicitly freed or reallocated). Each such allocation shall yield a pointer to an object  
 42971        disjoint from any other object. The pointer returned points to the start (lowest byte address) of  
 42972        the allocated space. If the space cannot be allocated, a null pointer shall be returned. If the size of  
 42973        the space requested is 0, the behavior is implementation-defined: either a null pointer shall be  
 42974        returned, or the behavior shall be as if the size were some non-zero value, except that the  
 42975        behavior is undefined if the returned pointer is used to access an object.

42976 **RETURN VALUE**

42977        Upon successful completion with *size* not equal to 0, *malloc()* shall return a pointer to the  
 42978        allocated space. If *size* is 0, either:

- 42979 CX        • A null pointer shall be returned and *errno* may be set to an implementation-defined value,  
 42980            or  
 42981            • A pointer to the allocated space shall be returned. The application shall ensure that the  
 42982            pointer is not used to access an object.

42983 CX        Otherwise, it shall return a null pointer and set *errno* to indicate the error.

42984 **ERRORS**42985        The *malloc()* function shall fail if:

42986 CX        [ENOMEM]        Insufficient storage space is available.

42987 **EXAMPLES**

42988        None.

42989 **APPLICATION USAGE**

42990        None.

42991 **RATIONALE**

42992        None.

42993 **FUTURE DIRECTIONS**

42994        None.

42995 **SEE ALSO**42996        *calloc()*, *free()*, *getrlimit()*, *posix\_memalign()*, *realloc()*

42997        XBD &lt;stdlib.h&gt;

42998 **CHANGE HISTORY**

42999 First released in Issue 1. Derived from Issue 1 of the SVID.

43000 **Issue 6**

43001 Extensions beyond the ISO C standard are marked.

43002 The following new requirements on POSIX implementations derive from alignment with the  
43003 Single UNIX Specification:

- 43004
- In the RETURN VALUE section, the requirement to set *errno* to indicate an error is added.
  - The [ENOMEM] error condition is added.
- 43005

43006 **NAME**

43007       mblen — get number of bytes in a character

43008 **SYNOPSIS**

43009       #include &lt;stdlib.h&gt;

43010       int mblen(const char \*s, size\_t n);

43011 **DESCRIPTION**

43012 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 43013 conflict between the requirements described here and the ISO C standard is unintentional. This  
 43014 volume of POSIX.1-2008 defers to the ISO C standard.

43015       If *s* is not a null pointer, *mblen()* shall determine the number of bytes constituting the character  
 43016 pointed to by *s*. Except that the shift state of *mbtowc()* is not affected, it shall be equivalent to:

43017       mbtowc((wchar\_t \*)0, s, n);

43018       The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls  
 43019 *mblen()*.

43020       The behavior of this function is affected by the *LC\_CTYPE* category of the current locale. For a  
 43021 state-dependent encoding, this function shall be placed into its initial state by a call for which its  
 43022 character pointer argument, *s*, is a null pointer. Subsequent calls with *s* as other than a null  
 43023 pointer shall cause the internal state of the function to be altered as necessary. A call with *s* as a  
 43024 null pointer shall cause this function to return a non-zero value if encodings have state  
 43025 dependency, and 0 otherwise. If the implementation employs special bytes to change the shift  
 43026 state, these bytes shall not produce separate wide-character codes, but shall be grouped with an  
 43027 adjacent character. Changing the *LC\_CTYPE* category causes the shift state of this function to be  
 43028 unspecified.

43029 CX       The *mblen()* function need not be thread-safe.

43030 **RETURN VALUE**

43031       If *s* is a null pointer, *mblen()* shall return a non-zero or 0 value, if character encodings,  
 43032 respectively, do or do not have state-dependent encodings. If *s* is not a null pointer, *mblen()* shall  
 43033 either return 0 (if *s* points to the null byte), or return the number of bytes that constitute the  
 43034 character (if the next *n* or fewer bytes form a valid character), or return -1 (if they do not form a  
 43035 CX       valid character) and may set *errno* to indicate the error. In no case shall the value returned be  
 43036 greater than *n* or the value of the {MB\_CUR\_MAX} macro.

43037 **ERRORS**43038       The *mblen()* function may fail if:

43039 CX       [EILSEQ]       An invalid character sequence is detected. In the POSIX locale an [EILSEQ]  
 43040 error cannot occur since all byte values are valid characters.

43041 **EXAMPLES**

43042       None.

43043 **APPLICATION USAGE**

43044       None.

43045 **RATIONALE**

43046       None.

43047 **FUTURE DIRECTIONS**

43048 None.

43049 **SEE ALSO**43050 *mbtowc()*, *mbstowcs()*, *wctomb()*, *wcstombs()*43051 XBD **<stdlib.h>**43052 **CHANGE HISTORY**

43053 First released in Issue 4. Aligned with the ISO C standard.

43054 **Issue 7**

43055 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0367 [109] is applied.

43056 **NAME**43057 `mbrlen` — get number of bytes in a character (restartable)43058 **SYNOPSIS**43059 `#include <wchar.h>`

43060 `size_t mbrlen(const char *restrict s, size_t n,`  
 43061 `mbstate_t *restrict ps);`

43062 **DESCRIPTION**

43063 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 43064 conflict between the requirements described here and the ISO C standard is unintentional. This  
 43065 volume of POSIX.1-2008 defers to the ISO C standard.

43066 If *s* is not a null pointer, `mbrlen()` shall determine the number of bytes constituting the character  
 43067 pointed to by *s*. It shall be equivalent to:

43068 `mbstate_t internal;`  
 43069 `mbrtowc(NULL, s, n, ps != NULL ? ps : &internal);`

43070 If *ps* is a null pointer, the `mbrlen()` function shall use its own internal **mbstate\_t** object, which is  
 43071 initialized at program start-up to the initial conversion state. Otherwise, the **mbstate\_t** object  
 43072 pointed to by *ps* shall be used to completely describe the current conversion state of the  
 43073 associated character sequence. The implementation shall behave as if no function defined in this  
 43074 volume of POSIX.1-2008 calls `mbrlen()`.

43075 The behavior of this function is affected by the `LC_CTYPE` category of the current locale.

43076 CX The `mbrlen()` function need not be thread-safe if called with a NULL *ps* argument.

43077 The `mbrlen()` function shall not change the setting of `errno` if successful.

43078 **RETURN VALUE**

43079 The `mbrlen()` function shall return the first of the following that applies:

43080 0 If the next *n* or fewer bytes complete the character that corresponds to the null  
 43081 wide character.

43082 *positive* If the next *n* or fewer bytes complete a valid character; the value returned shall  
 43083 be the number of bytes that complete the character.

43084 **(size\_t)–2** If the next *n* bytes contribute to an incomplete but potentially valid character,  
 43085 and all *n* bytes have been processed. When *n* has at least the value of the  
 43086 {MB\_CUR\_MAX} macro, this case can only occur if *s* points at a sequence of  
 43087 redundant shift sequences (for implementations with state-dependent  
 43088 encodings).

43089 **(size\_t)–1** If an encoding error occurs, in which case the next *n* or fewer bytes do not  
 43090 contribute to a complete and valid character. In this case, [EILSEQ] shall be  
 43091 stored in `errno` and the conversion state is undefined.

43092 **ERRORS**

43093 The `mbrlen()` function shall fail if:

43094 CX [EILSEQ] An invalid character sequence is detected. In the POSIX locale an [EILSEQ] +  
 43095 error cannot occur since all byte values are valid characters.

43096 The `mbrlen()` function may fail if:

43097 [EINVAL] *ps* points to an object that contains an invalid conversion state.

43098 **EXAMPLES**

43099 None.

43100 **APPLICATION USAGE**

43101 None.

43102 **RATIONALE**

43103 None.

43104 **FUTURE DIRECTIONS**

43105 None.

43106 **SEE ALSO**

43107 *mbstowc()*, *mbstowc()*

43108 XBD <**wchar.h**>

43109 **CHANGE HISTORY**

43110 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
43111 (E).

43112 **Issue 6**

43113 The *mbrlen()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

43114 **Issue 7**

43115 Austin Group Interpretation 1003.1-2001 #170 is applied.

43116 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0368 [109,105] is applied.



**NAME**

`mbrtowc` — convert a character to a wide-character code (restartable)

**SYNOPSIS**

```
#include <wchar.h>
```

```
size_t mbrtowc(wchar_t *restrict pwc, const char *restrict s,
               size_t n, mbstate_t *restrict ps);
```

**DESCRIPTION**

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

If *s* is a null pointer, the `mbrtowc()` function shall be equivalent to the call:

```
mbrtowc(NULL, "", 1, ps)
```

In this case, the values of the arguments *pwc* and *n* are ignored.

If *s* is not a null pointer, the `mbrtowc()` function shall inspect at most *n* bytes beginning at the byte pointed to by *s* to determine the number of bytes needed to complete the next character (including any shift sequences). If the function determines that the next character is completed, it shall determine the value of the corresponding wide character and then, if *pwc* is not a null pointer, shall store that value in the object pointed to by *pwc*. If the corresponding wide character is the null wide character, the resulting state described shall be the initial conversion state.

If *ps* is a null pointer, the `mbrtowc()` function shall use its own internal `mbstate_t` object, which shall be initialized at program start-up to the initial conversion state. Otherwise, the `mbstate_t` object pointed to by *ps* shall be used to completely describe the current conversion state of the associated character sequence. The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls `mbrtowc()`.

The behavior of this function is affected by the `LC_CTYPE` category of the current locale.

The `mbrtowc()` function need not be thread-safe if called with a NULL *ps* argument.

The `mbrtowc()` function shall not change the setting of *errno* if successful.

**RETURN VALUE**

The `mbrtowc()` function shall return the first of the following that applies:

0 If the next *n* or fewer bytes complete the character that corresponds to the null wide character (which is the value stored).

between 1 and *n* inclusive

If the next *n* or fewer bytes complete a valid character (which is the value stored); the value returned shall be the number of bytes that complete the character.

(`size_t`)−2

If the next *n* bytes contribute to an incomplete but potentially valid character, and all *n* bytes have been processed (no value is stored). When *n* has at least the value of the `{MB_CUR_MAX}` macro, this case can only occur if *s* points at a sequence of redundant shift sequences (for implementations with state-dependent encodings).

(`size_t`)−1

If an encoding error occurs, in which case the next *n* or fewer bytes do not contribute to a complete and valid character (no value is stored). In this case, `[EILSEQ]` shall be stored in *errno* and the conversion state is undefined.

43161 **ERRORS**43162 The *mbrtowc*() function shall fail if:

43163 CX [EILSEQ] An invalid character sequence is detected. In the POSIX locale an [EILSEQ] +  
43164 error cannot occur since all byte values are valid characters.

43165 The *mbrtowc*() function may fail if:

43166 CX [EINVAL] *ps* points to an object that contains an invalid conversion state.

43167 **EXAMPLES**

43168 None.

43169 **APPLICATION USAGE**

43170 None.

43171 **RATIONALE**

43172 None.

43173 **FUTURE DIRECTIONS**

43174 None.

43175 **SEE ALSO**43176 *mbsinit()*, *mbsrtowcs()*

43177 XBD &lt;wchar.h&gt;

43178 **CHANGE HISTORY**

43179 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
43180 (E).

43181 **Issue 6**43182 The *mbrtowc*() prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

43183 The following new requirements on POSIX implementations derive from alignment with the  
43184 Single UNIX Specification:

- 43185 • The [EINVAL] error condition is added.

43186 ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

43187 **Issue 7**

43188 Austin Group Interpretation 1003.1-2001 #170 is applied.

43189 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0369 [109,105] is applied.

43190 **NAME**

43191       mbsinit — determine conversion object status

43192 **SYNOPSIS**

43193       #include &lt;wchar.h&gt;

43194       int mbsinit(const mbstate\_t \*ps);

43195 **DESCRIPTION**

43196 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 43197 conflict between the requirements described here and the ISO C standard is unintentional. This  
 43198 volume of POSIX.1-2008 defers to the ISO C standard.

43199       If *ps* is not a null pointer, the *mbsinit()* function shall determine whether the object pointed to by  
 43200 *ps* describes an initial conversion state.

43201 **RETURN VALUE**

43202       The *mbsinit()* function shall return non-zero if *ps* is a null pointer, or if the pointed-to object  
 43203 describes an initial conversion state; otherwise, it shall return zero.

43204       If an **mbstate\_t** object is altered by any of the functions described as “restartable”, and is then  
 43205 used with a different character sequence, or in the other conversion direction, or with a different  
 43206 *LC\_CTYPE* category setting than on earlier function calls, the behavior is undefined.

43207 **ERRORS**

43208       No errors are defined.

43209 **EXAMPLES**

43210       None.

43211 **APPLICATION USAGE**

43212       The **mbstate\_t** object is used to describe the current conversion state from a particular character  
 43213 sequence to a wide-character sequence (or *vice versa*) under the rules of a particular setting of the  
 43214 *LC\_CTYPE* category of the current locale.

43215       The initial conversion state corresponds, for a conversion in either direction, to the beginning of  
 43216 a new character sequence in the initial shift state. A zero valued **mbstate\_t** object is at least one  
 43217 way to describe an initial conversion state. A zero valued **mbstate\_t** object can be used to initiate  
 43218 conversion involving any character sequence, in any *LC\_CTYPE* category setting.

43219 **RATIONALE**

43220       None.

43221 **FUTURE DIRECTIONS**

43222       None.

43223 **SEE ALSO**43224       *mbrlen()*, *mbrtowc()*, *mbstowcs()*, *wcrtomb()*, *wcsrtombs()*

43225       XBD &lt;wchar.h&gt;

43226 **CHANGE HISTORY**

43227       First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
 43228 (E).

## 43229 NAME

43230 mbsnrtowcs, mbsrtowcs — convert a character string to a wide-character string (restartable)

## 43231 SYNOPSIS

43232 #include &lt;wchar.h&gt;

```
43233 CX size_t mbsnrtowcs(wchar_t *restrict dst, const char **restrict src,
43234 size_t nmc, size_t len, mbstate_t *restrict ps);
43235 size_t mbsrtowcs(wchar_t *restrict dst, const char **restrict src,
43236 size_t len, mbstate_t *restrict ps);
```

## 43237 DESCRIPTION

43238 CX For *mbsrtowcs()*: The functionality described on this reference page is aligned with the ISO C  
 43239 standard. Any conflict between the requirements described here and the ISO C standard is  
 43240 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

43241 The *mbsrtowcs()* function shall convert a sequence of characters, beginning in the conversion  
 43242 state described by the object pointed to by *ps*, from the array indirectly pointed to by *src* into a  
 43243 sequence of corresponding wide characters. If *dst* is not a null pointer, the converted characters  
 43244 shall be stored into the array pointed to by *dst*. Conversion continues up to and including a  
 43245 terminating null character, which shall also be stored. Conversion shall stop early in either of the  
 43246 following cases:

- 43247 • A sequence of bytes is encountered that does not form a valid character.
- 43248 • *len* codes have been stored into the array pointed to by *dst* (and *dst* is not a null pointer).

43249 Each conversion shall take place as if by a call to the *mbrtowc()* function.

43250 If *dst* is not a null pointer, the pointer object pointed to by *src* shall be assigned either a null  
 43251 pointer (if conversion stopped due to reaching a terminating null character) or the address just  
 43252 past the last character converted (if any). If conversion stopped due to reaching a terminating  
 43253 null character, and if *dst* is not a null pointer, the resulting state described shall be the initial  
 43254 conversion state.

43255 If *ps* is a null pointer, the *mbsrtowcs()* function shall use its own internal **mbstate\_t** object, which  
 43256 is initialized at program start-up to the initial conversion state. Otherwise, the **mbstate\_t** object  
 43257 pointed to by *ps* shall be used to completely describe the current conversion state of the  
 43258 associated character sequence.

43259 CX The *mbsnrtowcs()* function shall be equivalent to the *mbsrtowcs()* function, except that the  
 43260 conversion of characters indirectly pointed to by *src* is limited to at most *nmc* bytes (the size of  
 43261 the input buffer), and under conditions where *mbsrtowcs()* would assign the address just past  
 43262 the last character converted (if any) to the pointer object pointed to by *src*, *mbsnrtowcs()* shall  
 43263 instead assign the address just past the last byte processed (if any) to that pointer object. If the  
 43264 input buffer ends with an incomplete character, it is unspecified whether conversion stops at the  
 43265 end of the previous character (if any), or at the end of the input buffer. In the latter case, a  
 43266 subsequent call to *mbsnrtowcs()* with an input buffer that starts with the remainder of the  
 43267 incomplete character shall correctly complete the conversion of that character.

43268 The behavior of these functions shall be affected by the *LC\_CTYPE* category of the current locale.

43269 The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls  
 43270 these functions.

43271 CX The *mbsnrtowcs()* and *mbsrtowcs()* functions need not be thread-safe if called with a NULL *ps*  
 43272 argument.

43273 The *mbsrtowcs()* function shall not change the setting of *errno* if successful.

43274 **RETURN VALUE**

43275 If the input conversion encounters a sequence of bytes that do not form a valid character, an  
 43276 encoding error occurs. In this case, these functions shall store the value of the macro [EILSEQ] in  
 43277 *errno* and shall return (**size\_t**)−1; the conversion state is undefined. Otherwise, these functions  
 43278 shall return the number of characters successfully converted, not including the terminating null  
 43279 (if any).

43280 **ERRORS**

43281 These functions shall fail if:

43282 CX [EILSEQ] An invalid character sequence is detected. In the POSIX locale an [EILSEQ] +  
 43283 error cannot occur since all byte values are valid characters.

43284 These functions may fail if:

43285 CX [EINVAL] *ps* points to an object that contains an invalid conversion state.

43286 **EXAMPLES**

43287 None.

43288 **APPLICATION USAGE**

43289 None.

43290 **RATIONALE**

43291 None.

43292 **FUTURE DIRECTIONS**

43293 A future version may require that when the input buffer ends with an incomplete character, |  
 43294 conversion stops at the end of the input buffer.

43295 **SEE ALSO**

43296 *iconv()*, *mbrtowc()*, *mbsinit()*

43297 XBD <wchar.h>

43298 **CHANGE HISTORY**

43299 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
 43300 (E).

43301 **Issue 6**

43302 The *mbsrtowcs()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

43303 The [EINVAL] error condition is marked CX.

43304 **Issue 7**

43305 Austin Group Interpretation 1003.1-2001 #170 is applied.

43306 The *mbsnrtowcs()* function is added from The Open Group Technical Standard, 2006, Extended  
 43307 API Set Part 1.

43308 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0370 [109,105] is applied.

43309 **NAME**

43310 mbstowcs — convert a character string to a wide-character string

43311 **SYNOPSIS**

43312 #include &lt;stdlib.h&gt;

43313 size\_t mbstowcs(wchar\_t \*restrict pwcs, const char \*restrict s,  
43314 size\_t n);43315 **DESCRIPTION**43316 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
43317 conflict between the requirements described here and the ISO C standard is unintentional. This  
43318 volume of POSIX.1-2008 defers to the ISO C standard.43319 The *mbstowcs()* function shall convert a sequence of characters that begins in the initial shift  
43320 state from the array pointed to by *s* into a sequence of corresponding wide-character codes and  
43321 shall store not more than *n* wide-character codes into the array pointed to by *pwcs*. No  
43322 characters that follow a null byte (which is converted into a wide-character code with value 0)  
43323 shall be examined or converted. Each character shall be converted as if by a call to *mbtowc()*,  
43324 except that the shift state of *mbtowc()* is not affected.43325 No more than *n* elements shall be modified in the array pointed to by *pwcs*. If copying takes  
43326 place between objects that overlap, the behavior is undefined.43327 XSI The behavior of this function shall be affected by the *LC\_CTYPE* category of the current locale.  
43328 If *pwcs* is a null pointer, *mbstowcs()* shall return the length required to convert the entire array  
43329 regardless of the value of *n*, but no values are stored.43330 **RETURN VALUE**43331 CX If an invalid character is encountered, *mbstowcs()* shall return (size\_t)-1 and shall set *errno* to  
43332 indicate the error.43333 XSI Otherwise, *mbstowcs()* shall return the number of the array elements modified (or required if  
43334 *pwcs* is null), not including a terminating 0 code, if any. The array shall not be zero-terminated if  
43335 the value returned is *n*.43336 **ERRORS**43337 The *mbstowcs()* function shall fail if:43338 CX [EILSEQ] An invalid character sequence is detected. In the POSIX locale an [EILSEQ]  
43339 error cannot occur since all byte values are valid characters.43340 **EXAMPLES**

43341 None.

43342 **APPLICATION USAGE**

43343 None.

43344 **RATIONALE**

43345 None.

43346 **FUTURE DIRECTIONS**

43347 None.

43348 **SEE ALSO**43349 *mblen()*, *mbtowc()*, *wctomb()*, *wcstombs()*

43350 XBD &lt;stdlib.h&gt;

43351 **CHANGE HISTORY**

43352 First released in Issue 4. Aligned with the ISO C standard.

43353 **Issue 6**43354 The *mbstowcs()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

43355 Extensions beyond the ISO C standard are marked.

43356 **Issue 7**

43357 Austin Group Interpretation 1003.1-2001 #170 is applied.

43358 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0371 [195] is applied.

43359 **NAME**

43360 mbtowc — convert a character to a wide-character code

43361 **SYNOPSIS**

43362 #include &lt;stdlib.h&gt;

43363 int mbtowc(wchar\_t \*restrict *pwc*, const char \*restrict *s*, size\_t *n*);43364 **DESCRIPTION**

43365 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 43366 conflict between the requirements described here and the ISO C standard is unintentional. This  
 43367 volume of POSIX.1-2008 defers to the ISO C standard.

43368 If *s* is not a null pointer, *mbtowc*() shall determine the number of bytes that constitute the  
 43369 character pointed to by *s*. It shall then determine the wide-character code for the value of type  
 43370 **wchar\_t** that corresponds to that character. (The value of the wide-character code corresponding  
 43371 to the null byte is 0.) If the character is valid and *pwc* is not a null pointer, *mbtowc*() shall store  
 43372 the wide-character code in the object pointed to by *pwc*.

43373 The behavior of this function is affected by the *LC\_CTYPE* category of the current locale. For a  
 43374 state-dependent encoding, this function is placed into its initial state by a call for which its  
 43375 character pointer argument, *s*, is a null pointer. Subsequent calls with *s* as other than a null  
 43376 pointer shall cause the internal state of the function to be altered as necessary. A call with *s* as a  
 43377 null pointer shall cause this function to return a non-zero value if encodings have state  
 43378 dependency, and 0 otherwise. If the implementation employs special bytes to change the shift  
 43379 state, these bytes shall not produce separate wide-character codes, but shall be grouped with an  
 43380 adjacent character. Changing the *LC\_CTYPE* category causes the shift state of this function to be  
 43381 unspecified. At most *n* bytes of the array pointed to by *s* shall be examined.

43382 The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls  
 43383 *mbtowc*()).

43384 CX The *mbtowc*() function need not be thread-safe.

43385 **RETURN VALUE**

43386 If *s* is a null pointer, *mbtowc*() shall return a non-zero or 0 value, if character encodings,  
 43387 respectively, do or do not have state-dependent encodings. If *s* is not a null pointer, *mbtowc*()  
 43388 shall either return 0 (if *s* points to the null byte), or return the number of bytes that constitute the  
 43389 CX converted character (if the next *n* or fewer bytes form a valid character), or return -1 and shall  
 43390 set *errno* to indicate the error (if they do not form a valid character).

43391 In no case shall the value returned be greater than *n* or the value of the {MB\_CUR\_MAX} macro.

43392 **ERRORS**

43393 The *mbtowc*() function shall fail if:

43394 CX [EILSEQ] An invalid character sequence is detected. In the POSIX locale an [EILSEQ]  
 43395 error cannot occur since all byte values are valid characters.



43396 **EXAMPLES**

43397 None.

43398 **APPLICATION USAGE**

43399 None.

43400 **RATIONALE**

43401 None.

43402 **FUTURE DIRECTIONS**

43403 None.

43404 **SEE ALSO**43405 *mblen()*, *mbstowcs()*, *wctomb()*, *wcstombs()*

43406 XBD &lt;stdlib.h&gt;

43407 **CHANGE HISTORY**

43408 First released in Issue 4. Aligned with the ISO C standard.

43409 **Issue 6**43410 The *mbtowc()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

43411 Extensions beyond the ISO C standard are marked.

43412 **Issue 7**

43413 Austin Group Interpretation 1003.1-2001 #170 is applied.

43414 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0372 [109] and XSH/TC1-2008/0373  
43415 [195] are applied.

43416 **NAME**

43417 memccpy — copy bytes in memory

43418 **SYNOPSIS**

```
43419 XSI      #include <string.h>
43420          void *memccpy(void *restrict s1, const void *restrict s2,
43421                        int c, size_t n);
```

43422 **DESCRIPTION**

43423 The *memccpy()* function shall copy bytes from memory area *s2* into *s1*, stopping after the first  
43424 occurrence of byte *c* (converted to an **unsigned char**) is copied, or after *n* bytes are copied,  
43425 whichever comes first. If copying takes place between objects that overlap, the behavior is  
43426 undefined.

43427 **RETURN VALUE**

43428 The *memccpy()* function shall return a pointer to the byte after the copy of *c* in *s1*, or a null  
43429 pointer if *c* was not found in the first *n* bytes of *s2*.

43430 **ERRORS**

43431 No errors are defined.

43432 **EXAMPLES**

43433 None.

43434 **APPLICATION USAGE**43435 The *memccpy()* function does not check for the overflow of the receiving memory area.43436 **RATIONALE**

43437 None.

43438 **FUTURE DIRECTIONS**

43439 None.

43440 **SEE ALSO**43441 XBD [<string.h>](#)43442 **CHANGE HISTORY**

43443 First released in Issue 1. Derived from Issue 1 of the SVID.

43444 **Issue 6**

43445 The **restrict** keyword is added to the *memccpy()* prototype for alignment with the  
43446 ISO/IEC 9899:1999 standard.

43447 **NAME**

43448 memchr — find byte in memory

43449 **SYNOPSIS**

43450 #include &lt;string.h&gt;

43451 void \*memchr(const void \*s, int c, size\_t n);

43452 **DESCRIPTION**

43453 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 43454 conflict between the requirements described here and the ISO C standard is unintentional. This  
 43455 volume of POSIX.1-2008 defers to the ISO C standard.

43456 The *memchr()* function shall locate the first occurrence of *c* (converted to an **unsigned char**) in  
 43457 the initial *n* bytes (each interpreted as **unsigned char**) pointed to by *s*.

43458 Implementations shall behave as if they read the memory byte by byte from the beginning of the  
 43459 bytes pointed to by *s* and stop at the first occurrence of *c* (if it is found in the initial *n* bytes).

43460 **RETURN VALUE**

43461 The *memchr()* function shall return a pointer to the located byte, or a null pointer if the byte is  
 43462 not found.

43463 **ERRORS**

43464 No errors are defined.

43465 **EXAMPLES**

43466 None.

43467 **APPLICATION USAGE**

43468 None.

43469 **RATIONALE**

43470 None.

43471 **FUTURE DIRECTIONS**

43472 None.

43473 **SEE ALSO**43474 XBD [<string.h>](#)43475 **CHANGE HISTORY**

43476 First released in Issue 1. Derived from Issue 1 of the SVID.

43477 **Issue 7**

43478 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0374 [110] is applied.

## 43479 NAME

43480 memcmp — compare bytes in memory

## 43481 SYNOPSIS

43482 #include <string.h>

43483 int memcmp(const void \*s1, const void \*s2, size\_t n);

## 43484 DESCRIPTION

43485 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
43486 conflict between the requirements described here and the ISO C standard is unintentional. This  
43487 volume of POSIX.1-2008 defers to the ISO C standard.

43488 The *memcmp()* function shall compare the first *n* bytes (each interpreted as **unsigned char**) of the  
43489 object pointed to by *s1* to the first *n* bytes of the object pointed to by *s2*.

43490 The sign of a non-zero return value shall be determined by the sign of the difference between the  
43491 values of the first pair of bytes (both interpreted as type **unsigned char**) that differ in the objects  
43492 being compared.

## 43493 RETURN VALUE

43494 The *memcmp()* function shall return an integer greater than, equal to, or less than 0, if the object  
43495 pointed to by *s1* is greater than, equal to, or less than the object pointed to by *s2*, respectively.

## 43496 ERRORS

43497 No errors are defined.

## 43498 EXAMPLES

43499 None.

## 43500 APPLICATION USAGE

43501 None.

## 43502 RATIONALE

43503 None.

## 43504 FUTURE DIRECTIONS

43505 None.

## 43506 SEE ALSO

43507 XBD <string.h>

## 43508 CHANGE HISTORY

43509 First released in Issue 1. Derived from Issue 1 of the SVID.

43510 **NAME**

43511       memcpy — copy bytes in memory

43512 **SYNOPSIS**

43513       #include &lt;string.h&gt;

43514       void \*memcpy(void \*restrict s1, const void \*restrict s2, size\_t n);

43515 **DESCRIPTION**

43516 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 43517 conflict between the requirements described here and the ISO C standard is unintentional. This  
 43518 volume of POSIX.1-2008 defers to the ISO C standard.

43519       The *memcpy()* function shall copy *n* bytes from the object pointed to by *s2* into the object pointed  
 43520 to by *s1*. If copying takes place between objects that overlap, the behavior is undefined.

43521 **RETURN VALUE**43522       The *memcpy()* function shall return *s1*; no return value is reserved to indicate an error.43523 **ERRORS**

43524       No errors are defined.

43525 **EXAMPLES**

43526       None.

43527 **APPLICATION USAGE**43528       The *memcpy()* function does not check for the overflow of the receiving memory area.43529 **RATIONALE**

43530       None.

43531 **FUTURE DIRECTIONS**

43532       None.

43533 **SEE ALSO**

43534       XBD &lt;string.h&gt;

43535 **CHANGE HISTORY**

43536       First released in Issue 1. Derived from Issue 1 of the SVID.

43537 **Issue 6**43538       The *memcpy()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

43539 **NAME**

43540 memmove — copy bytes in memory with overlapping areas

43541 **SYNOPSIS**

43542 #include &lt;string.h&gt;

43543 void \*memmove(void \*s1, const void \*s2, size\_t n);

43544 **DESCRIPTION**

43545 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
43546 conflict between the requirements described here and the ISO C standard is unintentional. This  
43547 volume of POSIX.1-2008 defers to the ISO C standard.

43548 The *memmove()* function shall copy *n* bytes from the object pointed to by *s2* into the object  
43549 pointed to by *s1*. Copying takes place as if the *n* bytes from the object pointed to by *s2* are first  
43550 copied into a temporary array of *n* bytes that does not overlap the objects pointed to by *s1* and  
43551 *s2*, and then the *n* bytes from the temporary array are copied into the object pointed to by *s1*.

43552 **RETURN VALUE**43553 The *memmove()* function shall return *s1*; no return value is reserved to indicate an error.43554 **ERRORS**

43555 No errors are defined.

43556 **EXAMPLES**

43557 None.

43558 **APPLICATION USAGE**

43559 None.

43560 **RATIONALE**

43561 None.

43562 **FUTURE DIRECTIONS**

43563 None.

43564 **SEE ALSO**

43565 XBD &lt;string.h&gt;

43566 **CHANGE HISTORY**

43567 First released in Issue 4. Derived from the ANSI C standard.

43568 **NAME**

43569       memset — set bytes in memory

43570 **SYNOPSIS**

43571       #include &lt;string.h&gt;

43572       void \*memset(void \*s, int c, size\_t n);

43573 **DESCRIPTION**

43574 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 43575       conflict between the requirements described here and the ISO C standard is unintentional. This  
 43576       volume of POSIX.1-2008 defers to the ISO C standard.

43577       The `memset()` function shall copy `c` (converted to an **unsigned char**) into each of the first `n` bytes  
 43578       of the object pointed to by `s`.

43579 **RETURN VALUE**43580       The `memset()` function shall return `s`; no return value is reserved to indicate an error.43581 **ERRORS**

43582       No errors are defined.

43583 **EXAMPLES**

43584       None.

43585 **APPLICATION USAGE**

43586       None.

43587 **RATIONALE**

43588       None.

43589 **FUTURE DIRECTIONS**

43590       None.

43591 **SEE ALSO**

43592       XBD &lt;string.h&gt;

43593 **CHANGE HISTORY**

43594       First released in Issue 1. Derived from Issue 1 of the SVID.

43595 **NAME**

43596       mkdir, mkdirat — make a directory |

43597 **SYNOPSIS**

43598       #include &lt;sys/stat.h&gt;

43599       int mkdir(const char \*path, mode\_t mode);

43600 OH      #include &lt;fcntl.h&gt; +

43601 +

43602       int mkdirat(int fd, const char \*path, mode\_t mode);

43603 **DESCRIPTION**

43604       The *mkdir()* function shall create a new directory with name *path*. The file permission bits of the  
 43605       new directory shall be initialized from *mode*. These file permission bits of the *mode* argument  
 43606       shall be modified by the process' file creation mask.

43607       When bits in *mode* other than the file permission bits are set, the meaning of these additional bits  
 43608       is implementation-defined.

43609       The directory's user ID shall be set to the process' effective user ID. The directory's group ID  
 43610       shall be set to the group ID of the parent directory or to the effective group ID of the process.  
 43611       Implementations shall provide a way to initialize the directory's group ID to the group ID of the  
 43612       parent directory. Implementations may, but need not, provide an implementation-defined way  
 43613       to initialize the directory's group ID to the effective group ID of the calling process.

43614       The newly created directory shall be an empty directory.

43615       If *path* names a symbolic link, *mkdir()* shall fail and set *errno* to [EEXIST].

43616       Upon successful completion, *mkdir()* shall mark for update the last data access, last data  
 43617       modification, and last file status change timestamps of the directory. Also, the last data  
 43618       modification and last file status change timestamps of the directory that contains the new entry  
 43619       shall be marked for update.

43620       The *mkdirat()* function shall be equivalent to the *mkdir()* function except in the case where *path*  
 43621       specifies a relative path. In this case the newly created directory is created relative to the  
 43622       directory associated with the file descriptor *fd* instead of the current working directory. If the  
 43623       access mode of the open file description associated with the file descriptor is not O\_SEARCH, |  
 43624       the function shall check whether directory searches are permitted using the current permissions |  
 43625       of the directory underlying the file descriptor. If the access mode is O\_SEARCH, the function |  
 43626       shall not perform the check.

43627       If *mkdirat()* is passed the special value AT\_FDCWD in the *fd* parameter, the current working  
 43628       directory shall be used and the behavior shall be identical to a call to *mkdir()*.

43629 **RETURN VALUE**

43630       Upon successful completion, these functions shall return 0. Otherwise, these functions shall  
 43631       return -1 and set *errno* to indicate the error. If -1 is returned, no directory shall be created.

43632 **ERRORS**

43633       These functions shall fail if:

43634       [EACCES]       Search permission is denied on a component of the path prefix, or write  
 43635       permission is denied on the parent directory of the directory to be created.

43636       [EEXIST]       The named file exists.



|       |                                                            |                                                                                                |
|-------|------------------------------------------------------------|------------------------------------------------------------------------------------------------|
| 43637 | [ELOOP]                                                    | A loop exists in symbolic links encountered during resolution of the <i>path</i>               |
| 43638 |                                                            | argument.                                                                                      |
| 43639 | [EMLINK]                                                   | The link count of the parent directory would exceed {LINK_MAX}.                                |
| 43640 | [ENAMETOOLONG]                                             |                                                                                                |
| 43641 |                                                            | The length of a component of a pathname is longer than {NAME_MAX}.                             |
| 43642 | [ENOENT]                                                   | A component of the path prefix specified by <i>path</i> does not name an existing              |
| 43643 |                                                            | directory or <i>path</i> is an empty string.                                                   |
| 43644 | [ENOSPC]                                                   | The file system does not contain enough space to hold the contents of the new                  |
| 43645 |                                                            | directory or to extend the parent directory of the new directory.                              |
| 43646 | [ENOTDIR]                                                  | A component of the path prefix names an existing file that is neither a                        |
| 43647 |                                                            | directory nor a symbolic link to a directory.                                                  |
| 43648 | [EROFS]                                                    | The parent directory resides on a read-only file system.                                       |
| 43649 | In addition, the <i>mkdirat</i> () function shall fail if: |                                                                                                |
| 43650 | [EACCES]                                                   | The access mode of the open file description associated with <i>fd</i> is not +                |
| 43651 |                                                            | O_SEARCH and the permissions of the directory underlying <i>fd</i> do not permit +             |
| 43652 |                                                            | directory searches.                                                                            |
| 43653 | [EBADF]                                                    | The <i>path</i> argument does not specify an absolute path and the <i>fd</i> argument is       |
| 43654 |                                                            | neither AT_FDCWD nor a valid file descriptor open for reading or searching.                    |
| 43655 | [ENOTDIR]                                                  | The <i>path</i> argument is not an absolute path and <i>fd</i> is a file descriptor associated |
| 43656 |                                                            | with a non-directory file.                                                                     |
| 43657 | These functions may fail if:                               |                                                                                                |
| 43658 | [ELOOP]                                                    | More than {SYMLOOP_MAX} symbolic links were encountered during                                 |
| 43659 |                                                            | resolution of the <i>path</i> argument.                                                        |
| 43660 | [ENAMETOOLONG]                                             |                                                                                                |
| 43661 |                                                            | The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a                       |
| 43662 |                                                            | symbolic link produced an intermediate result with a length that exceeds                       |
| 43663 |                                                            | {PATH_MAX}.                                                                                    |

## 43664 EXAMPLES

### 43665 Creating a Directory

43666 The following example shows how to create a directory named **/home/cnd/mod1**, with  
 43667 read/write/search permissions for owner and group, and with read/search permissions for  
 43668 others.

```

43669 #include <sys/types.h>
43670 #include <sys/stat.h>

43671 int status;
43672 ...
43673 status = mkdir("/home/cnd/mod1", S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);

```

## 43674 APPLICATION USAGE

43675 None.

**RATIONALE**

The *mkdir()* function originated in 4.2 BSD and was added to System V in Release 3.0.

4.3 BSD detects [ENAMETOOLONG].

The POSIX.1-1990 standard required that the group ID of a newly created directory be set to the group ID of its parent directory or to the effective group ID of the creating process. FIPS 151-2 required that implementations provide a way to have the group ID be set to the group ID of the containing directory, but did not prohibit implementations also supporting a way to set the group ID to the effective group ID of the creating process. Conforming applications should not assume which group ID will be used. If it matters, an application can use *chown()* to set the group ID after the directory is created, or determine under what conditions the implementation will set the desired group ID.

The purpose of the *mkdirat()* function is to create a directory in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to the call to *mkdir()*, resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *mkdirat()* function it can be guaranteed that the newly created directory is located relative to the desired directory.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*chmod()*, *mkdtemp()*, *mknod()*, *umask()*

XBD [\*<fcntl.h>\*](#), [\*<sys/stat.h>\*](#), [\*<sys/types.h>\*](#)

**CHANGE HISTORY**

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

**Issue 6**

In the SYNOPSIS, the optional include of the [\*<sys/types.h>\*](#) header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include [\*<sys/types.h>\*](#) has been removed. Although [\*<sys/types.h>\*](#) was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #143 is applied.

The *mkdirat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Changes are made related to support for finegrained timestamps.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0375 [461], XSH/TC1-2008/0376 [324], XSH/TC1-2008/0377 [277], XSH/TC1-2008/0378 [278], and XSH/TC1-2008/0379 [278] are applied.



43718 **NAME**

43719 mkdtemp, mkstemp — create a unique directory or file

43720 **SYNOPSIS**

```
43721 CX      #include <stdlib.h>
43722          char *mkdtemp(char *template);
43723          int mkstemp(char *template);
```

43724 **DESCRIPTION**

43725 The *mkdtemp()* function shall create a directory with a unique name derived from *template*. The  
 43726 application shall ensure that the string provided in *template* is a pathname ending with at least  
 43727 six trailing 'X' characters. The *mkdtemp()* function shall modify the contents of *template* by  
 43728 replacing six or more 'X' characters at the end of the pathname with the same number of  
 43729 characters from the portable filename character set. The characters shall be chosen such that the  
 43730 resulting pathname does not duplicate the name of an existing file at the time of the call to  
 43731 *mkdtemp()*. The *mkdtemp()* function shall use the resulting pathname to create the new directory  
 43732 as if by a call to:

```
43733 mkdir(pathname, S_IRWXU)
```

43734 The *mkstemp()* function shall create a regular file with a unique name derived from *template* and  
 43735 return a file descriptor for the file open for reading and writing. The application shall ensure that  
 43736 the string provided in *template* is a pathname ending with at least six trailing 'X' characters. The  
 43737 *mkstemp()* function shall modify the contents of *template* by replacing six or more 'X' characters  
 43738 at the end of the pathname with the same number of characters from the portable filename  
 43739 character set. The characters shall be chosen such that the resulting pathname does not duplicate  
 43740 the name of an existing file at the time of the call to *mkstemp()*. The *mkstemp()* function shall use  
 43741 the resulting pathname to create the file, and obtain a file descriptor for it, as if by a call to:

```
43742 open(pathname, O_RDWR|O_CREAT|O_EXCL, S_IRUSR|S_IWUSR)
```

43743 By behaving as if the O\_EXCL flag for *open()* is set, the function prevents any possible race  
 43744 condition between testing whether the file exists and opening it for use.

43745 **RETURN VALUE**

43746 Upon successful completion, the *mkdtemp()* function shall return the value of *template*.  
 43747 Otherwise, it shall return a null pointer and shall set *errno* to indicate the error.

43748 Upon successful completion, the *mkstemp()* function shall return an open file descriptor.  
 43749 Otherwise, it shall return -1 and shall set *errno* to indicate the error.

43750 **ERRORS**

43751 The *mkdtemp()* function shall fail if:

- |       |                |                                                                                                                                                      |
|-------|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| 43752 | [EACCES]       | Search permission is denied on a component of the path prefix, or write permission is denied on the parent directory of the directory to be created. |
| 43753 |                |                                                                                                                                                      |
| 43754 | [EINVAL]       | The string pointed to by <i>template</i> does not end in "XXXXXX".                                                                                   |
| 43755 | [ELOOP]        | A loop exists in symbolic links encountered during resolution of the path of the directory to be created.                                            |
| 43756 |                |                                                                                                                                                      |
| 43757 | [EMLINK]       | The link count of the parent directory would exceed {LINK_MAX}.                                                                                      |
| 43758 | [ENAMETOOLONG] |                                                                                                                                                      |
| 43759 |                | The length of a component of a pathname is longer than {NAME_MAX}.                                                                                   |

|       |                |                                                                                       |
|-------|----------------|---------------------------------------------------------------------------------------|
| 43760 | [ENOENT]       | A component of the path prefix specified by the <i>template</i> argument does not     |
| 43761 |                | name an existing directory.                                                           |
| 43762 | [ENOSPC]       | The file system does not contain enough space to hold the contents of the new         |
| 43763 |                | directory or to extend the parent directory of the new directory.                     |
| 43764 | [ENOTDIR]      | A component of the path prefix names an existing file that is neither a               |
| 43765 |                | directory nor a symbolic link to a directory.                                         |
| 43766 | [EROFS]        | The parent directory resides on a read-only file system.                              |
| 43767 |                | The <i>mkdtemp()</i> function may fail if:                                            |
| 43768 | [ELOOP]        | More than {SYMLOOP_MAX} symbolic links were encountered during                        |
| 43769 |                | resolution of the path of the directory to be created.                                |
| 43770 | [ENAMETOOLONG] |                                                                                       |
| 43771 |                | The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a              |
| 43772 |                | symbolic link produced an intermediate result with a length that exceeds              |
| 43773 |                | {PATH_MAX}.                                                                           |
| 43774 |                | The error conditions for the <i>mkstemp()</i> function are defined in <i>open()</i> . |

## 43775 EXAMPLES

### 43776 Generating a Pathname

43777 The following example creates a file with a 10-character name beginning with the characters  
 43778 "file" and opens the file for reading and writing. The value returned as the value of *fd* is a file  
 43779 descriptor that identifies the file.

```
43780 #include <stdlib.h>
43781 ...
43782 char template[] = "/tmp/fileXXXXXX";
43783 int fd;
43784 fd = mkstemp(template);
```

## 43785 APPLICATION USAGE

43786 It is possible to run out of letters.

43787 Portable applications should pass exactly six trailing 'X's in the template and no more; |  
 43788 implementations may treat any additional trailing 'X's as either a fixed or replaceable part of |  
 43789 the template. To be sure of only passing six, a fixed string of at least one non-'X' character |  
 43790 should precede the six 'X's. |

43791 Since 'X' is in the portable filename character set, some of the replacement characters can be |  
 43792 'X's, leaving part (or even all) of the template effectively unchanged. |

## 43793 RATIONALE

43794 None.

## 43795 FUTURE DIRECTIONS

43796 None.

## 43797 SEE ALSO

43798 *getpid()*, *mkdir()*, *open()*, *tmpfile()*, *tmpnam()*

43799 XBD <stdlib.h>

43800 **CHANGE HISTORY**

43801 First released in Issue 4, Version 2.

43802 **Issue 5**

43803 Moved from X/OPEN UNIX extension to BASE.

43804 **Issue 7**

43805 Austin Group Interpretation 1003.1-2001 #143 is applied.

43806 SD5-XSH-ERN-168 is applied, clarifying file permissions upon creation.

43807 The *mkstemp()* function is moved from the XSI option to the Base.43808 The *mkdtemp()* function is added from The Open Group Technical Standard, 2006, Extended API  
43809 Set Part 1.43810 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0380 [291], XSH/TC1-2008/0381 [324],  
43811 and XSH/TC1-2008/0382 [291] are applied.

43812 **NAME**

43813 mkfifo, mkfifoat — make a FIFO special file |

43814 **SYNOPSIS**

43815 #include &lt;sys/stat.h&gt;

43816 int mkfifo(const char \*path, mode\_t mode);

43817 OH #include &lt;fcntl.h&gt; +

43818 +

43819 int mkfifoat(int fd, const char \*path, mode\_t mode);

43820 **DESCRIPTION**

43821 The *mkfifo()* function shall create a new FIFO special file named by the pathname pointed to by  
 43822 *path*. The file permission bits of the new FIFO shall be initialized from *mode*. The file permission  
 43823 bits of the *mode* argument shall be modified by the process' file creation mask.

43824 When bits in *mode* other than the file permission bits are set, the effect is implementation-  
 43825 defined.

43826 If *path* names a symbolic link, *mkfifo()* shall fail and set *errno* to [EEXIST].

43827 The FIFO's user ID shall be set to the process' effective user ID. The FIFO's group ID shall be set  
 43828 to the group ID of the parent directory or to the effective group ID of the process.  
 43829 Implementations shall provide a way to initialize the FIFO's group ID to the group ID of the  
 43830 parent directory. Implementations may, but need not, provide an implementation-defined way  
 43831 to initialize the FIFO's group ID to the effective group ID of the calling process.

43832 Upon successful completion, *mkfifo()* shall mark for update the last data access, last data  
 43833 modification, and last file status change timestamps of the file. Also, the last data modification  
 43834 and last file status change timestamps of the directory that contains the new entry shall be  
 43835 marked for update.

43836 The *mkfifoat()* function shall be equivalent to the *mkfifo()* function except in the case where *path*  
 43837 specifies a relative path. In this case the newly created FIFO is created relative to the directory  
 43838 associated with the file descriptor *fd* instead of the current working directory. If the access mode |  
 43839 of the open file description associated with the file descriptor is not O\_SEARCH, the function |  
 43840 shall check whether directory searches are permitted using the current permissions of the |  
 43841 directory underlying the file descriptor. If the access mode is O\_SEARCH, the function shall not |  
 43842 perform the check.

43843 If *mkfifoat()* is passed the special value AT\_FDCWD in the *fd* parameter, the current working  
 43844 directory shall be used and the behavior shall be identical to a call to *mkfifo()*.

43845 **RETURN VALUE**

43846 Upon successful completion, these functions shall return 0. Otherwise, these functions shall  
 43847 return -1 and set *errno* to indicate the error. If -1 is returned, no FIFO shall be created.

43848 **ERRORS**

43849 These functions shall fail if:

43850 [EACCES] A component of the path prefix denies search permission, or write permission  
 43851 is denied on the parent directory of the FIFO to be created.

43852 [EEXIST] The named file already exists.

43853 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*  
 43854 argument.

43855 [ENAMETOOLONG]  
 43856 The length of a component of a pathname is longer than {NAME\_MAX}.

43857 [ENOENT] A component of the path prefix of *path* does not name an existing file or *path* is  
 43858 an empty string.

43859 [ENOENT] or [ENOTDIR]  
 43860 The *path* argument contains at least one non-`<slash>` character and ends with  
 43861 one or more trailing `<slash>` characters. If *path* without the trailing `<slash>`  
 43862 characters would name an existing file, an [ENOENT] error shall not occur.

43863 [ENOSPC] The directory that would contain the new file cannot be extended or the file  
 43864 system is out of file-allocation resources.

43865 [ENOTDIR] A component of the path prefix names an existing file that is neither a  
 43866 directory nor a symbolic link to a directory.

43867 [EROFS] The named file resides on a read-only file system.

43868 The *mkfifoat()* function shall fail if:

43869 [EACCES] The access mode of the open file description associated with *fd* is not  
 43870 O\_SEARCH and the permissions of the directory underlying *fd* do not permit  
 43871 directory searches.

43872 [EBADF] The *path* argument does not specify an absolute path and the *fd* argument is  
 43873 neither AT\_FDCWD nor a valid file descriptor open for reading or searching.

43874 [ENOTDIR] The *path* argument is not an absolute path and *fd* is a file descriptor associated  
 43875 with a non-directory file.

43876 These functions may fail if:

43877 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
 43878 resolution of the *path* argument.

43879 [ENAMETOOLONG]  
 43880 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
 43881 symbolic link produced an intermediate result with a length that exceeds  
 43882 {PATH\_MAX}.

## 43883 EXAMPLES

### 43884 Creating a FIFO File

43885 The following example shows how to create a FIFO file named `/home/cnd/mod_done`, with  
 43886 read/write permissions for owner, and with read permissions for group and others.

```
43887 #include <sys/types.h>
43888 #include <sys/stat.h>
43889
43889 int status;
43890 ...
43891 status = mkfifo("/home/cnd/mod_done", S_IWUSR | S_IRUSR |
43892               S_IRGRP | S_IROTH);
```



**APPLICATION USAGE**

None.

**RATIONALE**

The syntax of this function is intended to maintain compatibility with historical implementations of *mknod()*. The latter function was included in the 1984 /usr/group standard but only for use in creating FIFO special files. The *mknod()* function was originally excluded from the POSIX.1-1988 standard as implementation-defined and replaced by *mkdir()* and *mkfifo()*. The *mknod()* function is now included for alignment with the Single UNIX Specification.

The POSIX.1-1990 standard required that the group ID of a newly created FIFO be set to the group ID of its parent directory or to the effective group ID of the creating process. FIPS 151-2 required that implementations provide a way to have the group ID be set to the group ID of the containing directory, but did not prohibit implementations also supporting a way to set the group ID to the effective group ID of the creating process. Conforming applications should not assume which group ID will be used. If it matters, an application can use *chown()* to set the group ID after the FIFO is created, or determine under what conditions the implementation will set the desired group ID.

The purpose of the *mkfifoat()* function is to create a FIFO special file in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *mkfifo()*, resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *mkfifoat()* function it can be guaranteed that the newly created FIFO is located relative to the desired directory.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*chmod()*, *mknod()*, *umask()*

XBD [<fcntl.h>](#), [<sys/stat.h>](#), [<sys/types.h>](#)

**CHANGE HISTORY**

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

**Issue 6**

In the SYNOPSIS, the optional include of the [<sys/types.h>](#) header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include [<sys/types.h>](#) has been removed. Although [<sys/types.h>](#) was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The [ELOOP] optional error condition is added.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #143 is applied.

The *mkfifoat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

43937 Changes are made related to support for finegrained timestamps.  
43938 Changes are made to allow a directory to be opened for searching.  
43939 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0383 [461], XSH/TC1-2008/0384  
43940 [146,435], XSH/TC1-2008/0385 [324], XSH/TC1-2008/0386 [278], and XSH/TC1-2008/0387  
43941 [278] are applied.

43942 **NAME**

43943 mknod, mknodat — make directory, special file, or regular file

43944 **SYNOPSIS**43945 XSI `#include <sys/stat.h>`43946 `int mknod(const char *path, mode_t mode, dev_t dev);`43947 OH XSI `#include <fcntl.h>`

43948

43949 XSI `int mknodat(int fd, const char *path, mode_t mode, dev_t dev);`+  
+43950 **DESCRIPTION**43951 The *mknod()* function shall create a new file named by the pathname to which the argument *path*  
43952 points.43953 The file type for *path* is OR'ed into the *mode* argument, and the application shall select one of the  
43954 following symbolic constants:

| Name    | Description                      |
|---------|----------------------------------|
| S_IFIFO | FIFO-special                     |
| S_IFCHR | Character-special (non-portable) |
| S_IFDIR | Directory (non-portable)         |
| S_IFBLK | Block-special (non-portable)     |
| S_IFREG | Regular (non-portable)           |

43961 The only portable use of *mknod()* is to create a FIFO-special file. If *mode* is not S\_IFIFO or *dev* is  
43962 not 0, the behavior of *mknod()* is unspecified.43963 The permissions for the new file are OR'ed into the *mode* argument, and may be selected from  
43964 any combination of the following symbolic constants:

| Name    | Description                                 |
|---------|---------------------------------------------|
| S_ISUID | Set user ID on execution.                   |
| S_ISGID | Set group ID on execution.                  |
| S_IRWXU | Read, write, or execute (search) by owner.  |
| S_IRUSR | Read by owner.                              |
| S_IWUSR | Write by owner.                             |
| S_IXUSR | Execute (search) by owner.                  |
| S_IRWXG | Read, write, or execute (search) by group.  |
| S_IRGRP | Read by group.                              |
| S_IWGRP | Write by group.                             |
| S_IXGRP | Execute (search) by group.                  |
| S_IRWXO | Read, write, or execute (search) by others. |
| S_IROTH | Read by others.                             |
| S_IWOTH | Write by others.                            |
| S_IXOTH | Execute (search) by others.                 |
| S_ISVTX | On directories, restricted deletion flag.   |

43981 The user ID of the file shall be initialized to the effective user ID of the process. The group ID of  
 43982 the file shall be initialized to either the effective group ID of the process or the group ID of the  
 43983 parent directory. Implementations shall provide a way to initialize the file's group ID to the  
 43984 group ID of the parent directory. Implementations may, but need not, provide an

implementation-defined way to initialize the file's group ID to the effective group ID of the calling process. The owner, group, and other permission bits of *mode* shall be modified by the file mode creation mask of the process. The *mknod()* function shall clear each bit whose corresponding bit in the file mode creation mask of the process is set.

If *path* names a symbolic link, *mknod()* shall fail and set *errno* to [EEXIST].

Upon successful completion, *mknod()* shall mark for update the last data access, last data modification, and last file status change timestamps of the file. Also, the last data modification and last file status change timestamps of the directory that contains the new entry shall be marked for update.

Only a process with appropriate privileges may invoke *mknod()* for file types other than FIFO-special.

The *mknodat()* function shall be equivalent to the *mknod()* function except in the case where *path* specifies a relative path. In this case the newly created directory, special file, or regular file is located relative to the directory associated with the file descriptor *fd* instead of the current working directory. If the access mode of the open file description associated with the file descriptor is not O\_SEARCH, the function shall check whether directory searches are permitted using the current permissions of the directory underlying the file descriptor. If the access mode is O\_SEARCH, the function shall not perform the check.

If *mknodat()* is passed the special value AT\_FDCWD in the *fd* parameter, the current working directory shall be used and the behavior shall be identical to a call to *mknod()*.

#### RETURN VALUE

Upon successful completion, these functions shall return 0. Otherwise, these functions shall return -1 and set *errno* to indicate the error. If -1 is returned, the new file shall not be created.

#### ERRORS

These functions shall fail if:

[EACCES] A component of the path prefix denies search permission, or write permission is denied on the parent directory.

[EEXIST] The named file exists.

[EINVAL] An invalid argument exists.

[EIO] An I/O error occurred while accessing the file system.

[ELOOP] A loop exists in symbolic links encountered during resolution of the *path* argument.

[ENAMETOOLONG]

The length of a component of a pathname is longer than {NAME\_MAX}.

[ENOENT] A component of the path prefix of *path* does not name an existing file or *path* is an empty string.

[ENOENT] or [ENOTDIR]

The *path* argument contains at least one non-*<slash>* character and ends with one or more trailing *<slash>* characters. If *path* without the trailing *<slash>* characters would name an existing file, an [ENOENT] error shall not occur.

[ENOSPC] The directory that would contain the new file cannot be extended or the file system is out of file allocation resources.

|       |                                              |                                                                                                |
|-------|----------------------------------------------|------------------------------------------------------------------------------------------------|
| 44027 | [ENOTDIR]                                    | A component of the path prefix names an existing file that is neither a                        |
| 44028 |                                              | directory nor a symbolic link to a directory.                                                  |
| 44029 | [EPERM]                                      | The invoking process does not have appropriate privileges and the file type is                 |
| 44030 |                                              | not FIFO-special.                                                                              |
| 44031 | [EROFS]                                      | The directory in which the file is to be created is located on a read-only file                |
| 44032 |                                              | system.                                                                                        |
| 44033 | The <i>mknodat()</i> function shall fail if: |                                                                                                |
| 44034 | [EACCES]                                     | The access mode of the open file description associated with <i>fd</i> is not                  |
| 44035 |                                              | O_SEARCH and the permissions of the directory underlying <i>fd</i> do not permit               |
| 44036 |                                              | directory searches.                                                                            |
| 44037 | [EBADF]                                      | The <i>path</i> argument does not specify an absolute path and the <i>fd</i> argument is       |
| 44038 |                                              | neither AT_FDCWD nor a valid file descriptor open for reading or searching.                    |
| 44039 | [ENOTDIR]                                    | The <i>path</i> argument is not an absolute path and <i>fd</i> is a file descriptor associated |
| 44040 |                                              | with a non-directory file.                                                                     |
| 44041 | These functions may fail if:                 |                                                                                                |
| 44042 | [ELOOP]                                      | More than {SYMLOOP_MAX} symbolic links were encountered during                                 |
| 44043 |                                              | resolution of the <i>path</i> argument.                                                        |
| 44044 | [ENAMETOOLONG]                               |                                                                                                |
| 44045 |                                              | The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a                       |
| 44046 |                                              | symbolic link produced an intermediate result with a length that exceeds                       |
| 44047 |                                              | {PATH_MAX}.                                                                                    |

## 44048 EXAMPLES

### 44049 Creating a FIFO Special File

44050 The following example shows how to create a FIFO special file named */home/cnd/mod\_done*,  
 44051 with read/write permissions for owner, and with read permissions for group and others.

```

44052 #include <sys/types.h>
44053 #include <sys/stat.h>
44054 dev_t dev;
44055 int status;
44056 ...
44057 status = mknod("/home/cnd/mod_done", S_IFIFO | S_IWUSR |
44058               S_IRUSR | S_IRGRP | S_IROTH, dev);

```

## 44059 APPLICATION USAGE

44060 The *mkfifo()* function is preferred over this function for making FIFO special files.

## 44061 RATIONALE

44062 The POSIX.1-1990 standard required that the group ID of a newly created file be set to the group  
 44063 ID of its parent directory or to the effective group ID of the creating process. FIPS 151-2 required  
 44064 that implementations provide a way to have the group ID be set to the group ID of the  
 44065 containing directory, but did not prohibit implementations also supporting a way to set the  
 44066 group ID to the effective group ID of the creating process. Conforming applications should not  
 44067 assume which group ID will be used. If it matters, an application can use *chown()* to set the  
 44068 group ID after the file is created, or determine under what conditions the implementation will  
 44069 set the desired group ID.

44070 The purpose of the *mknodat()* function is to create directories, special files, or regular files in  
44071 directories other than the current working directory without exposure to race conditions. Any  
44072 part of the path of a file could be changed in parallel to a call to *mknod()*, resulting in unspecified  
44073 behavior. By opening a file descriptor for the target directory and using the *mknodat()* function it  
44074 can be guaranteed that the newly created directory, special file, or regular file is located relative  
44075 to the desired directory.

#### 44076 FUTURE DIRECTIONS

44077 None.

#### 44078 SEE ALSO

44079 *chmod()*, *creat()*, *exec*, *fstatat()*, *mkdir()*, *mkfifo()*, *open()*, *umask()*

44080 XBD <fcntl.h>, <sys/stat.h>

#### 44081 CHANGE HISTORY

44082 First released in Issue 4, Version 2.

##### 44083 Issue 5

44084 Moved from X/OPEN UNIX extension to BASE.

##### 44085 Issue 6

44086 The normative text is updated to avoid use of the term “must” for application requirements.

44087 The wording of the mandatory [ELOOP] error condition is updated, and a second optional  
44088 [ELOOP] error condition is added.

##### 44089 Issue 7

44090 Austin Group Interpretation 1003.1-2001 #143 is applied.

44091 The *mknodat()* function is added from The Open Group Technical Standard, 2006, Extended API  
44092 Set Part 2.

44093 Changes are made related to support for finegrained timestamps.

44094 Changes are made to allow a directory to be opened for searching.

44095 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0388 [324], XSH/TC1-2008/0389 [461],  
44096 XSH/TC1-2008/0390 [146,435], XSH/TC1-2008/0391 [278], and XSH/TC1-2008/0392 [278] are  
44097 applied.

44098 **NAME**

44099 mkstemp — create a unique file |

44100 **SYNOPSIS**

```
44101 CX      #include <stdlib.h>  
44102      int mkstemp(char *template);
```

44103 **DESCRIPTION**44104 Refer to *mkdtemp()*.

44105 **NAME**

44106 mktime — convert broken-down time into time since the Epoch

44107 **SYNOPSIS**

44108 #include &lt;time.h&gt;

44109 time\_t mktime(struct tm \*timeptr);

44110 **DESCRIPTION**

44111 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 44112 conflict between the requirements described here and the ISO C standard is unintentional. This  
 44113 volume of POSIX.1-2008 defers to the ISO C standard.

44114 The *mktime()* function shall convert the broken-down time, expressed as local time, in the  
 44115 structure pointed to by *timeptr*, into a time since the Epoch value with the same encoding as that  
 44116 of the values returned by *time()*. The original values of the *tm\_wday* and *tm\_yday* components of  
 44117 the structure shall be ignored, and the original values of the other components shall not be  
 44118 restricted to the ranges described in <time.h>.

44119 CX A positive or 0 value for *tm\_isdst* shall cause *mktime()* to presume initially that Daylight Savings  
 44120 Time, respectively, is or is not in effect for the specified time. A negative value for *tm\_isdst* shall  
 44121 cause *mktime()* to attempt to determine whether Daylight Savings Time is in effect for the  
 44122 specified time.

44123 Local timezone information shall be set as though *mktime()* called *tzset()*.

44124 The relationship between the **tm** structure (defined in the <time.h> header) and the time in  
 44125 seconds since the Epoch is that the result shall be as specified in the expression given in the  
 44126 definition of seconds since the Epoch (see XBD Section 4.16, on page 113) corrected for timezone  
 44127 and any seasonal time adjustments, where the names other than *tm\_yday* in the structure and in  
 44128 the expression correspond, and the *tm\_yday* value used in the expression is the day of the year  
 44129 from 0 to 365 inclusive, calculated from the other **tm** structure members specified in <time.h>  
 44130 (excluding *tm\_wday*).

44131 Upon successful completion, the values of the *tm\_wday* and *tm\_yday* components of the structure  
 44132 shall be set appropriately, and the other components shall be set to represent the specified time  
 44133 since the Epoch, but with their values forced to the ranges indicated in the <time.h> entry; the  
 44134 final value of *tm\_mday* shall not be set until *tm\_mon* and *tm\_year* are determined.

44135 **RETURN VALUE**

44136 The *mktime()* function shall return the specified time since the Epoch encoded as a value of type  
 44137 **time\_t**. If the time since the Epoch cannot be represented, the function shall return the value  
 44138 CX (**time\_t**)-1 and set *errno* to indicate the error.

44139 **ERRORS**44140 The *mktime()* function shall fail if:

44141 CX [EOVERFLOW] The result cannot be represented.



**EXAMPLES**

What day of the week is July 4, 2001?

```
#include <stdio.h>
#include <time.h>

struct tm time_str;
char daybuf[20];

int main(void)
{
    time_str.tm_year = 2001 - 1900;
    time_str.tm_mon = 7 - 1;
    time_str.tm_mday = 4;
    time_str.tm_hour = 0;
    time_str.tm_min = 0;
    time_str.tm_sec = 1;
    time_str.tm_isdst = -1;
    if (mktime(&time_str) == -1)
        (void)puts("-unknown-");
    else {
        (void)strftime(daybuf, sizeof(daybuf), "%A", &time_str);
        (void)puts(daybuf);
    }
    return 0;
}
```

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[\*asctime\(\)\*](#), [\*clock\(\)\*](#), [\*ctime\(\)\*](#), [\*difftime\(\)\*](#), [\*gmtime\(\)\*](#), [\*localtime\(\)\*](#), [\*strftime\(\)\*](#), [\*strptime\(\)\*](#), [\*time\(\)\*](#), [\*tzset\(\)\*](#), [\*utime\(\)\*](#)

XBD [Section 4.16](#) (on page 113), [\*\*<time.h>\*\*](#)

**CHANGE HISTORY**

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard and the ANSI C standard.

**Issue 6**

Extensions beyond the ISO C standard are marked.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/58 is applied, updating the RETURN VALUE and ERRORS sections to add the optional [EOVERFLOW] error as a CX extension.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/59 is applied, adding the *tzset()* function to the SEE ALSO section.

44184 **Issue 7**  
44185

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0393 [104] is applied.

**NAME**

mlock, munlock — lock or unlock a range of process address space (**REALTIME**)

**SYNOPSIS**

```
#include <sys/mman.h>

int mlock(const void *addr, size_t len);
int munlock(const void *addr, size_t len);
```

**DESCRIPTION**

The *mlock()* function shall cause those whole pages containing any part of the address space of the process starting at address *addr* and continuing for *len* bytes to be memory-resident until unlocked or until the process exits or *execs* another process image. The implementation may require that *addr* be a multiple of {PAGESIZE}.

The *munlock()* function shall unlock those whole pages containing any part of the address space of the process starting at address *addr* and continuing for *len* bytes, regardless of how many times *mlock()* has been called by the process for any of the pages in the specified range. The implementation may require that *addr* be a multiple of {PAGESIZE}.

If any of the pages in the range specified to a call to *munlock()* are also mapped into the address spaces of other processes, any locks established on those pages by another process are unaffected by the call of this process to *munlock()*. If any of the pages in the range specified by a call to *munlock()* are also mapped into other portions of the address space of the calling process outside the range specified, any locks established on those pages via the other mappings are also unaffected by this call.

Upon successful return from *mlock()*, pages in the specified range shall be locked and memory-resident. Upon successful return from *munlock()*, pages in the specified range shall be unlocked with respect to the address space of the process. Memory residency of unlocked pages is unspecified.

Appropriate privileges are required to lock process memory with *mlock()*.

**RETURN VALUE**

Upon successful completion, the *mlock()* and *munlock()* functions shall return a value of zero. Otherwise, no change is made to any locks in the address space of the process, and the function shall return a value of -1 and set *errno* to indicate the error.

**ERRORS**

The *mlock()* and *munlock()* functions shall fail if:

[ENOMEM] Some or all of the address range specified by the *addr* and *len* arguments does not correspond to valid mapped pages in the address space of the process.

The *mlock()* function shall fail if:

[EAGAIN] Some or all of the memory identified by the operation could not be locked when the call was made.

The *mlock()* and *munlock()* functions may fail if:

[EINVAL] The *addr* argument is not a multiple of {PAGESIZE}.

The *mlock()* function may fail if:

[ENOMEM] Locking the pages mapped by the specified range would exceed an implementation-defined limit on the amount of memory that the process may lock.

44229 [EPERM] The calling process does not have appropriate privileges to perform the  
44230 requested operation.

**44231 EXAMPLES**

44232 None.

**44233 APPLICATION USAGE**

44234 None.

**44235 RATIONALE**

44236 None.

**44237 FUTURE DIRECTIONS**

44238 None.

**44239 SEE ALSO**

44240 *exec, exit(), fork(), mlockall(), munmap()*

44241 XBD <sys/mman.h>

**44242 CHANGE HISTORY**

44243 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**44244 Issue 6**

44245 The *mlock()* and *munlock()* functions are marked as part of the Range Memory Locking option.

44246 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
44247 implementation does not support the Range Memory Locking option.

**NAME**

mlockall, munlockall — lock/unlock the address space of a process (**REALTIME**)

**SYNOPSIS**

```
ML    #include <sys/mman.h>

      int mlockall(int flags);
      int munlockall(void);
```

**DESCRIPTION**

The *mlockall()* function shall cause all of the pages mapped by the address space of a process to be memory-resident until unlocked or until the process exits or *execs* another process image. The *flags* argument determines whether the pages to be locked are those currently mapped by the address space of the process, those that are mapped in the future, or both. The *flags* argument is constructed from the bitwise-inclusive OR of one or more of the following symbolic constants, defined in *<sys/mman.h>*:

**MCL\_CURRENT** Lock all of the pages currently mapped into the address space of the process.

**MCL\_FUTURE** Lock all of the pages that become mapped into the address space of the process in the future, when those mappings are established.

If **MCL\_FUTURE** is specified, and the automatic locking of future mappings eventually causes the amount of locked memory to exceed the amount of available physical memory or any other implementation-defined limit, the behavior is implementation-defined. The manner in which the implementation informs the application of these situations is also implementation-defined.

The *munlockall()* function shall unlock all currently mapped pages of the address space of the process. Any pages that become mapped into the address space of the process after a call to *munlockall()* shall not be locked, unless there is an intervening call to *mlockall()* specifying **MCL\_FUTURE** or a subsequent call to *mlockall()* specifying **MCL\_CURRENT**. If pages mapped into the address space of the process are also mapped into the address spaces of other processes and are locked by those processes, the locks established by the other processes shall be unaffected by a call by this process to *munlockall()*.

Upon successful return from the *mlockall()* function that specifies **MCL\_CURRENT**, all currently mapped pages of the address space of the process shall be memory-resident and locked. Upon return from the *munlockall()* function, all currently mapped pages of the address space of the process shall be unlocked with respect to the address space of the process. The memory residency of unlocked pages is unspecified.

Appropriate privileges are required to lock process memory with *mlockall()*.

**RETURN VALUE**

Upon successful completion, the *mlockall()* function shall return a value of zero. Otherwise, no additional memory shall be locked, and the function shall return a value of *-1* and set *errno* to indicate the error. The effect of failure of *mlockall()* on previously existing locks in the address space is unspecified.

If it is supported by the implementation, the *munlockall()* function shall always return a value of zero. Otherwise, the function shall return a value of *-1* and set *errno* to indicate the error.

**ERRORS**

The *mlockall()* function shall fail if:

**[EAGAIN]** Some or all of the memory identified by the operation could not be locked when the call was made.

44292 [EINVAL] The *flags* argument is zero, or includes unimplemented flags.  
44293 The *mlockall()* function may fail if:  
44294 [ENOMEM] Locking all of the pages currently mapped into the address space of the  
44295 process would exceed an implementation-defined limit on the amount of  
44296 memory that the process may lock.  
44297 [EPERM] The calling process does not have appropriate privileges to perform the  
44298 requested operation.

#### 44299 EXAMPLES

44300 None.

#### 44301 APPLICATION USAGE

44302 None.

#### 44303 RATIONALE

44304 None.

#### 44305 FUTURE DIRECTIONS

44306 None.

#### 44307 SEE ALSO

44308 *exec*, *exit()*, *fork()*, *mlock()*, *munmap()*

44309 XBD <sys/mman.h>

#### 44310 CHANGE HISTORY

44311 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

#### 44312 Issue 6

44313 The *mlockall()* and *munlockall()* functions are marked as part of the Process Memory Locking  
44314 option.

44315 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
44316 implementation does not support the Process Memory Locking option.

44317 **NAME**

44318 mmap — map pages of memory

44319 **SYNOPSIS**

44320 #include &lt;sys/mman.h&gt;

```
44321 void *mmap(void *addr, size_t len, int prot, int flags,
44322           int fildes, off_t off);
```

44323 **DESCRIPTION**

44324 The *mmap()* function shall establish a mapping between an address space of a process and a  
 44325 memory object.

44326 The *mmap()* function shall be supported for the following memory objects:

- 44327 • Regular files
- 44328 SHM • Shared memory objects
- 44329 TYM • Typed memory objects

44330 Support for any other type of file is unspecified.

44331 The format of the call is as follows:

```
44332 pa=mmap(addr, len, prot, flags, fildes, off);
```

44333 The *mmap()* function shall establish a mapping between the address space of the process at an  
 44334 address *pa* for *len* bytes to the memory object represented by the file descriptor *fildes* at offset *off*  
 44335 for *len* bytes. The value of *pa* is an implementation-defined function of the parameter *addr* and  
 44336 the values of *flags*, further described below. A successful *mmap()* call shall return *pa* as its result.  
 44337 The address range starting at *pa* and continuing for *len* bytes shall be legitimate for the possible  
 44338 (not necessarily current) address space of the process. The range of bytes starting at *off* and  
 44339 continuing for *len* bytes shall be legitimate for the possible (not necessarily current) offsets in the  
 44340 memory object represented by *fildes*.

44341 TYM If *fildes* represents a typed memory object opened with either the  
 44342 POSIX\_TYPED\_MEM\_ALLOCATE flag or the POSIX\_TYPED\_MEM\_ALLOCATE\_CONTIG  
 44343 flag, the memory object to be mapped shall be that portion of the typed memory object allocated  
 44344 by the implementation as specified below. In this case, if *off* is non-zero, the behavior of *mmap()*  
 44345 is undefined. If *fildes* refers to a valid typed memory object that is not accessible from the calling  
 44346 process, *mmap()* shall fail.

44347 The mapping established by *mmap()* shall replace any previous mappings for those whole pages  
 44348 containing any part of the address space of the process starting at *pa* and continuing for *len*  
 44349 bytes.

44350 If the size of the mapped file changes after the call to *mmap()* as a result of some other operation  
 44351 on the mapped file, the effect of references to portions of the mapped region that correspond to  
 44352 added or removed portions of the file is unspecified.

44353 If *len* is zero, *mmap()* shall fail and no mapping shall be established.

44354 The parameter *prot* determines whether read, write, execute, or some combination of accesses  
 44355 are permitted to the data being mapped. The *prot* shall be either PROT\_NONE or the bitwise-  
 44356 inclusive OR of one or more of the other flags in the following table, defined in the  
 44357 <sys/mman.h> header.

| Symbolic Constant | Description              |
|-------------------|--------------------------|
| PROT_READ         | Data can be read.        |
| PROT_WRITE        | Data can be written.     |
| PROT_EXEC         | Data can be executed.    |
| PROT_NONE         | Data cannot be accessed. |

If an implementation cannot support the combination of access types specified by *prot*, the call to *mmap()* shall fail.

An implementation may permit accesses other than those specified by *prot*; however, the implementation shall not permit a write to succeed where PROT\_WRITE has not been set and shall not permit any access where PROT\_NONE alone has been set. The implementation shall support at least the following values of *prot*: PROT\_NONE, PROT\_READ, PROT\_WRITE, and the bitwise-inclusive OR of PROT\_READ and PROT\_WRITE. The file descriptor *fildes* shall have been opened with read permission, regardless of the protection options specified. If PROT\_WRITE is specified, the application shall ensure that it has opened the file descriptor *fildes* with write permission unless MAP\_PRIVATE is specified in the *flags* parameter as described below.

The parameter *flags* provides other information about the handling of the mapped data. The value of *flags* is the bitwise-inclusive OR of these options, defined in <sys/mman.h>:

| Symbolic Constant | Description                    |
|-------------------|--------------------------------|
| MAP_SHARED        | Changes are shared.            |
| MAP_PRIVATE       | Changes are private.           |
| MAP_FIXED         | Interpret <i>addr</i> exactly. |

It is implementation-defined whether MAP\_FIXED shall be supported. MAP\_FIXED shall be supported on XSI-conformant systems.

MAP\_SHARED and MAP\_PRIVATE describe the disposition of write references to the memory object. If MAP\_SHARED is specified, write references shall change the underlying object. If MAP\_PRIVATE is specified, modifications to the mapped data by the calling process shall be visible only to the calling process and shall not change the underlying object. It is unspecified whether modifications to the underlying object done after the MAP\_PRIVATE mapping is established are visible through the MAP\_PRIVATE mapping. Either MAP\_SHARED or MAP\_PRIVATE can be specified, but not both. The mapping type is retained across *fork()*.

The state of synchronization objects such as mutexes, semaphores, barriers, and conditional variables placed in shared memory mapped with MAP\_SHARED becomes undefined when the last region in any process containing the synchronization object is unmapped.

When *fildes* represents a typed memory object opened with either the POSIX\_TYPED\_MEM\_ALLOCATE flag or the POSIX\_TYPED\_MEM\_ALLOCATE\_CONTIG flag, *mmap()* shall, if there are enough resources available, map *len* bytes allocated from the corresponding typed memory object which were not previously allocated to any process in any processor that may access that typed memory object. If there are not enough resources available, the function shall fail. If *fildes* represents a typed memory object opened with the POSIX\_TYPED\_MEM\_ALLOCATE\_CONTIG flag, these allocated bytes shall be contiguous within the typed memory object. If *fildes* represents a typed memory object opened with the POSIX\_TYPED\_MEM\_ALLOCATE flag, these allocated bytes may be composed of non-contiguous fragments within the typed memory object. If *fildes* represents a typed memory object opened with neither the POSIX\_TYPED\_MEM\_ALLOCATE\_CONTIG flag nor the POSIX\_TYPED\_MEM\_ALLOCATE flag, *len* bytes starting at offset *off* within the typed memory object are mapped, exactly as when mapping a file or shared memory object. In this case, if two



processes map an area of typed memory using the same *off* and *len* values and using file descriptors that refer to the same memory pool (either from the same port or from a different port), both processes shall map the same region of storage.

When MAP\_FIXED is set in the *flags* argument, the implementation is informed that the value of *pa* shall be *addr*, exactly. If MAP\_FIXED is set, *mmap()* may return MAP\_FAILED and set *errno* to [EINVAL]. If a MAP\_FIXED request is successful, then any previous mappings **or memory locks** for those whole pages containing any part of the address range [*pa*,*pa+len*) shall be removed, as if by an appropriate call to *munmap()*, before the new mapping is established.

When MAP\_FIXED is not set, the implementation uses *addr* in an implementation-defined manner to arrive at *pa*. The *pa* so chosen shall be an area of the address space that the implementation deems suitable for a mapping of *len* bytes to the file. All implementations interpret an *addr* value of 0 as granting the implementation complete freedom in selecting *pa*, subject to constraints described below. A non-zero value of *addr* is taken to be a suggestion of a process address near which the mapping should be placed. When the implementation selects a value for *pa*, it never places a mapping at address 0, nor does it replace any extant mapping.

If MAP\_FIXED is specified and *addr* is non-zero, it shall have the same remainder as the *off* parameter, modulo the page size as returned by *sysconf()* when passed \_SC\_PAGESIZE or \_SC\_PAGE\_SIZE. The implementation may require that *off* is a multiple of the page size. If MAP\_FIXED is specified, the implementation may require that *addr* is a multiple of the page size. The system performs mapping operations over whole pages. Thus, while the parameter *len* need not meet a size or alignment constraint, the system shall include, in any mapping operation, any partial page specified by the address range starting at *pa* and continuing for *len* bytes.

The system shall always zero-fill any partial page at the end of an object. Further, the system shall never write out any modified portions of the last page of an object which are beyond its end. References within the address range starting at *pa* and continuing for *len* bytes to whole pages following the end of an object shall result in delivery of a SIGBUS signal.

An implementation may generate SIGBUS signals when a reference would cause an error in the mapped object, such as out-of-space condition.

The *mmap()* function shall add an extra reference to the file associated with the file descriptor *fd* which is not removed by a subsequent *close()* on that file descriptor. This reference shall be removed when there are no more mappings to the file.

The last data access timestamp of the mapped file may be marked for update at any time between the *mmap()* call and the corresponding *munmap()* call. The initial read or write reference to a mapped region shall cause the file's last data access timestamp to be marked for update if it has not already been marked for update.

The last data modification and last file status change timestamps of a file that is mapped with MAP\_SHARED and PROT\_WRITE shall be marked for update at some point in the interval between a write reference to the mapped region and the next call to *msync()* with MS\_ASYNC or MS\_SYNC for that portion of the file by any process. If there is no such call and if the underlying file is modified as a result of a write reference, then these timestamps shall be marked for update at some time after the write reference.

There may be implementation-defined limits on the number of memory regions that can be mapped (per process or per system).

If such a limit is imposed, whether the number of memory regions that can be mapped by a process is decreased by the use of *shmat()* is implementation-defined.

44451 If *mmap()* fails for reasons other than [EBADF], [EINVAL], or [ENOTSUP], some of the  
 44452 mappings in the address range starting at *addr* and continuing for *len* bytes may have been  
 44453 unmapped.

#### 44454 RETURN VALUE

44455 Upon successful completion, the *mmap()* function shall return the address at which the mapping  
 44456 was placed (*pa*); otherwise, it shall return a value of MAP\_FAILED and set *errno* to indicate the  
 44457 error. The symbol MAP\_FAILED is defined in the <sys/mman.h> header. No successful return  
 44458 from *mmap()* shall return the value MAP\_FAILED.

#### 44459 ERRORS

44460 The *mmap()* function shall fail if:

- |           |             |                                                                                                                                                                                                                                                                            |
|-----------|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 44461     | [EACCES]    | The <i>fildev</i> argument is not open for read, regardless of the protection specified,<br>44462 or <i>fildev</i> is not open for write and PROT_WRITE was specified for a<br>44463 MAP_SHARED type mapping.                                                              |
| 44464 ML  | [EAGAIN]    | The mapping could not be locked in memory, if required by <i>mlockall()</i> , due to<br>44465 a lack of resources.                                                                                                                                                         |
| 44466     | [EBADF]     | The <i>fildev</i> argument is not a valid open file descriptor.                                                                                                                                                                                                            |
| 44467     | [EINVAL]    | The value of <i>len</i> is zero.                                                                                                                                                                                                                                           |
| 44468     | [EINVAL]    | The value of <i>flags</i> is invalid (neither MAP_PRIVATE nor MAP_SHARED is<br>44469 set).                                                                                                                                                                                 |
| 44470     | [EMFILE]    | The number of mapped regions would exceed an implementation-defined<br>44471 limit (per process or per system).                                                                                                                                                            |
| 44472     | [ENODEV]    | The <i>fildev</i> argument refers to a file whose type is not supported by <i>mmap()</i> .                                                                                                                                                                                 |
| 44473     | [ENOMEM]    | MAP_FIXED was specified, and the range [ <i>addr</i> , <i>addr</i> + <i>len</i> ) exceeds that allowed<br>44474 for the address space of a process; or, if MAP_FIXED was not specified and<br>44475 there is insufficient room in the address space to effect the mapping. |
| 44476 ML  | [ENOMEM]    | The mapping could not be locked in memory, if required by <i>mlockall()</i> ,<br>44477 because it would require more space than the system is able to supply.                                                                                                              |
| 44478 TYM | [ENOMEM]    | Not enough unallocated memory resources remain in the typed memory<br>44479 object designated by <i>fildev</i> to allocate <i>len</i> bytes.                                                                                                                               |
| 44480     | [ENOTSUP]   | MAP_FIXED or MAP_PRIVATE was specified in the <i>flags</i> argument and the<br>44481 implementation does not support this functionality.                                                                                                                                   |
| 44482     |             | The implementation does not support the combination of accesses requested<br>44483 in the <i>prot</i> argument.                                                                                                                                                            |
| 44484     | [ENXIO]     | Addresses in the range [ <i>off</i> , <i>off</i> + <i>len</i> ) are invalid for the object specified by <i>fildev</i> .                                                                                                                                                    |
| 44485     | [ENXIO]     | MAP_FIXED was specified in <i>flags</i> and the combination of <i>addr</i> , <i>len</i> , and <i>off</i> is<br>44486 invalid for the object specified by <i>fildev</i> .                                                                                                   |
| 44487 TYM | [ENXIO]     | The <i>fildev</i> argument refers to a typed memory object that is not accessible from<br>44488 the calling process.                                                                                                                                                       |
| 44489     | [EOVERFLOW] | The file is a regular file and the value of <i>off</i> plus <i>len</i> exceeds the offset<br>44490 maximum established in the open file description associated with <i>fildev</i> .                                                                                        |

44491 The *mmap()* function may fail if:

44492 [EINVAL] The *addr* argument (if MAP\_FIXED was specified) or *off* is not a multiple of the  
 44493 page size as returned by *sysconf()*, or is considered invalid by the  
 44494 implementation.

#### 44495 EXAMPLES

44496 None.

#### 44497 APPLICATION USAGE

44498 Use of *mmap()* may reduce the amount of memory available to other memory allocation  
 44499 functions.

44500 Use of MAP\_FIXED may result in unspecified behavior in further use of *malloc()* and *shmat()*.  
 44501 The use of MAP\_FIXED is discouraged, as it may prevent an implementation from making the  
 44502 most effective use of resources. Most implementations require that *off* and *addr* are multiples of  
 44503 the page size as returned by *sysconf()*.

44504 The application must ensure correct synchronization when using *mmap()* in conjunction with  
 44505 any other file access method, such as *read()* and *write()*, standard input/output, and *shmat()*.

44506 The *mmap()* function allows access to resources via address space manipulations, instead of  
 44507 *read()/write()*. Once a file is mapped, all a process has to do to access it is use the data at the  
 44508 address to which the file was mapped. So, using pseudo-code to illustrate the way in which an  
 44509 existing program might be changed to use *mmap()*, the following:

```
44510 fildes = open(...)
44511 lseek(fildes, some_offset)
44512 read(fildes, buf, len)
44513 /* Use data in buf. */
```

44514 becomes:

```
44515 fildes = open(...)
44516 address = mmap(0, len, PROT_READ, MAP_PRIVATE, fildes, some_offset)
44517 /* Use data at address. */
```

#### 44518 RATIONALE

44519 After considering several other alternatives, it was decided to adopt the *mmap()* definition  
 44520 found in SVR4 for mapping memory objects into process address spaces. The SVR4 definition is  
 44521 minimal, in that it describes only what has been built, and what appears to be necessary for a  
 44522 general and portable mapping facility.

44523 Note that while *mmap()* was first designed for mapping files, it is actually a general-purpose  
 44524 mapping facility. It can be used to map any appropriate object, such as memory, files, devices,  
 44525 and so on, into the address space of a process.

44526 When a mapping is established, it is possible that the implementation may need to map more  
 44527 than is requested into the address space of the process because of hardware requirements. An  
 44528 application, however, cannot count on this behavior. Implementations that do not use a paged  
 44529 architecture may simply allocate a common memory region and return the address of it; such  
 44530 implementations probably do not allocate any more than is necessary. References past the end of  
 44531 the requested area are unspecified.

44532 If an application requests a mapping that overlaps existing mappings in the process, it might be  
 44533 desirable that an implementation detect this and inform the application. However, if the  
 44534 program specifies a fixed address mapping (which requires some implementation knowledge to  
 44535 determine a suitable address, if the function is supported at all), then the program is presumed

to be successfully managing its own address space and should be trusted when it asks to map over existing data structures. Furthermore, it is also desirable to make as few system calls as possible, and it might be considered onerous to require an *munmap()* before an *mmap()* to the same address range. This volume of POSIX.1-2008 specifies that the new mapping replaces any existing mappings (implying an automatic *munmap()* on the address range), following existing practice in this regard. The standard developers also considered whether there should be a way for new mappings to overlay existing mappings, but found no existing practice for this.

It is not expected that all hardware implementations are able to support all combinations of permissions at all addresses. Implementations are required to disallow write access to mappings without write permission and to disallow access to mappings without any access permission. Other than these restrictions, implementations may allow access types other than those requested by the application. For example, if the application requests only *PROT\_WRITE*, the implementation may also allow read access. A call to *mmap()* fails if the implementation cannot support allowing all the access requested by the application. For example, some implementations cannot support a request for both write access and execute access simultaneously. All implementations must support requests for no access, read access, write access, and both read and write access. Strictly conforming code must only rely on the required checks. These restrictions allow for portability across a wide range of hardware.

The *MAP\_FIXED* address treatment is likely to fail for non-page-aligned values and for certain architecture-dependent address ranges. Conforming implementations cannot count on being able to choose address values for *MAP\_FIXED* without utilizing non-portable, implementation-defined knowledge. Nonetheless, *MAP\_FIXED* is provided as a standard interface conforming to existing practice for utilizing such knowledge when it is available.

Similarly, in order to allow implementations that do not support virtual addresses, support for directly specifying any mapping addresses via *MAP\_FIXED* is not required and thus a conforming application may not count on it.

The *MAP\_PRIVATE* function can be implemented efficiently when memory protection hardware is available. When such hardware is not available, implementations can implement such “mappings” by simply making a real copy of the relevant data into process private memory, though this tends to behave similarly to *read()*.

The function has been defined to allow for many different models of using shared memory. However, all uses are not equally portable across all machine architectures. In particular, the *mmap()* function allows the system as well as the application to specify the address at which to map a specific region of a memory object. The most portable way to use the function is always to let the system choose the address, specifying *NULL* as the value for the argument *addr* and not to specify *MAP\_FIXED*.

If it is intended that a particular region of a memory object be mapped at the same address in a group of processes (on machines where this is even possible), then *MAP\_FIXED* can be used to pass in the desired mapping address. The system can still be used to choose the desired address if the first such mapping is made without specifying *MAP\_FIXED*, and then the resulting mapping address can be passed to subsequent processes for them to pass in via *MAP\_FIXED*. The availability of a specific address range cannot be guaranteed, in general.

The *mmap()* function can be used to map a region of memory that is larger than the current size of the object. Memory access within the mapping but beyond the current end of the underlying objects may result in *SIGBUS* signals being sent to the process. The reason for this is that the size of the object can be manipulated by other processes and can change at any moment. The implementation should tell the application that a memory reference is outside the object where this can be detected; otherwise, written data may be lost and read data may not reflect actual

data in the object.

Note that references beyond the end of the object do not extend the object as the new end cannot be determined precisely by most virtual memory hardware. Instead, the size can be directly manipulated by *ftruncate()*.

Process memory locking does apply to shared memory regions, and the `MCL_FUTURE` argument to *mlockall()* can be relied upon to cause new shared memory regions to be automatically locked.

Existing implementations of *mmap()* return the value `-1` when unsuccessful. Since the casting of this value to type `void *` cannot be guaranteed by the ISO C standard to be distinct from a successful value, this volume of POSIX.1-2008 defines the symbol `MAP_FAILED`, which a conforming implementation does not return as the result of a successful call.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*exec*, *fcntl()*, *fork()*, *lockf()*, *msync()*, *munmap()*, *mprotect()*, *posix\_typed\_mem\_open()*, *shmat()*, *sysconf()*

XBD [`<sys/mman.h>`](#)

## CHANGE HISTORY

First released in Issue 4, Version 2.

### Issue 5

Moved from X/OPEN UNIX extension to BASE.

Aligned with *mmap()* in the POSIX Realtime Extension as follows:

- The DESCRIPTION is extensively reworded.
- The `[EAGAIN]` and `[ENOTSUP]` mandatory error conditions are added.
- New cases of `[ENOMEM]` and `[ENXIO]` are added as mandatory error conditions.
- The value returned on failure is the value of the constant `MAP_FAILED`; this was previously defined as `-1`.

Large File Summit extensions are added.

### Issue 6

The *mmap()* function is marked as part of the Memory Mapped Files option.

The Open Group Corrigendum U028/6 is applied, changing `(void *)-1` to `MAP_FAILED`.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to describe the use of `MAP_FIXED`.
- The DESCRIPTION is updated to describe the addition of an extra reference to the file associated with the file descriptor passed to *mmap()*.
- The DESCRIPTION is updated to state that there may be implementation-defined limits on the number of memory regions that can be mapped.
- The DESCRIPTION is updated to describe constraints on the alignment and size of the *off* argument.

- The [EINVAL] and [EMFILE] error conditions are added.

- The [EOVERFLOW] error condition is added. This change is to support large files.

The following changes are made for alignment with the ISO POSIX-1: 1996 standard:

- The DESCRIPTION is updated to describe the cases when MAP\_PRIVATE and MAP\_FIXED need not be supported.

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- Semantics for typed memory objects are added to the DESCRIPTION.
- New [ENOMEM] and [ENXIO] errors are added to the ERRORS section.
- The *posix\_typed\_mem\_open()* function is added to the SEE ALSO section.

The normative text is updated to avoid use of the term “must” for application requirements.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/34 is applied, changing the margin code in the SYNOPSIS from MF|SHM to MC3 (notation for MF|SHM|TYM).

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/60 is applied, updating the DESCRIPTION and ERRORS sections to add the [EINVAL] error when *len* is zero.

#### Issue 7

Austin Group Interpretations 1003.1-2001 #078 and #079 are applied, clarifying page alignment requirements and adding a note about the state of synchronization objects becoming undefined when a shared region is unmapped.

Functionality relating to the Memory Protection and Memory Mapped Files options is moved to the Base.

Changes are made related to support for finegrained timestamps.



44645 **NAME**

44646 modf, modff, modfl — decompose a floating-point number

44647 **SYNOPSIS**

44648 #include &lt;math.h&gt;

44649 double modf(double *x*, double \**iptr*);44650 float modff(float *value*, float \**iptr*);44651 long double modfl(long double *value*, long double \**iptr*);44652 **DESCRIPTION**

44653 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 44654 conflict between the requirements described here and the ISO C standard is unintentional. This  
 44655 volume of POSIX.1-2008 defers to the ISO C standard.

44656 These functions shall break the argument *x* into integral and fractional parts, each of which has  
 44657 the same sign as the argument. It stores the integral part as a **double** (for the *modf*() function), a  
 44658 **float** (for the *modff*() function), or a **long double** (for the *modfl*() function), in the object pointed  
 44659 to by *iptr*.

44660 **RETURN VALUE**44661 Upon successful completion, these functions shall return the signed fractional part of *x*.44662 MX If *x* is NaN, a NaN shall be returned, and \**iptr* shall be set to a NaN.44663 If *x* is  $\pm\text{Inf}$ ,  $\pm 0$  shall be returned, and \**iptr* shall be set to  $\pm\text{Inf}$ .44664 **ERRORS**

44665 No errors are defined.

44666 **EXAMPLES**

44667 None.

44668 **APPLICATION USAGE**44669 The *modf*() function computes the function result and \**iptr* such that:44670 *a* = modf(*x*, *iptr*) ;44671 *x* == *a*+\**iptr* ;

44672 allowing for the usual floating-point inaccuracies.

44673 **RATIONALE**

44674 None.

44675 **FUTURE DIRECTIONS**

44676 None.

44677 **SEE ALSO**44678 *frexp()*, *isnan()*, *ldexp()*

44679 XBD &lt;math.h&gt;

44680 **CHANGE HISTORY**

44681 First released in Issue 1. Derived from Issue 1 of the SVID.

44682 **Issue 5**

44683 The DESCRIPTION is updated to indicate how an application should check for an error. This  
 44684 text was previously published in the APPLICATION USAGE section.

44685 **Issue 6**

44686 The *modff()* and *modfl()* functions are added for alignment with the ISO/IEC 9899:1999  
44687 standard.

44688 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
44689 revised to align with the ISO/IEC 9899:1999 standard.

44690 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
44691 marked.

44692 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/35 is applied, correcting the code example  
44693 in the APPLICATION USAGE section.



**NAME**

mprotect — set protection of memory mapping

**SYNOPSIS**

```
#include <sys/mman.h>
```

```
int mprotect(void *addr, size_t len, int prot);
```

**DESCRIPTION**

The *mprotect()* function shall change the access protections to be that specified by *prot* for those whole pages containing any part of the address space of the process starting at address *addr* and continuing for *len* bytes. The parameter *prot* determines whether read, write, execute, or some combination of accesses are permitted to the data being mapped. The *prot* argument should be either *PROT\_NONE* or the bitwise-inclusive OR of one or more of *PROT\_READ*, *PROT\_WRITE*, and *PROT\_EXEC*.

If an implementation cannot support the combination of access types specified by *prot*, the call to *mprotect()* shall fail.

An implementation may permit accesses other than those specified by *prot*; however, no implementation shall permit a write to succeed where *PROT\_WRITE* has not been set or shall permit any access where *PROT\_NONE* alone has been set. Implementations shall support at least the following values of *prot*: *PROT\_NONE*, *PROT\_READ*, *PROT\_WRITE*, and the bitwise-inclusive OR of *PROT\_READ* and *PROT\_WRITE*. If *PROT\_WRITE* is specified, the application shall ensure that it has opened the mapped objects in the specified address range with write permission, unless *MAP\_PRIVATE* was specified in the original mapping, regardless of whether the file descriptors used to map the objects have since been closed.

The implementation may require that *addr* be a multiple of the page size as returned by *sysconf()*.

The behavior of this function is unspecified if the mapping was not established by a call to *mmap()*.

When *mprotect()* fails for reasons other than *[EINVAL]*, the protections on some of the pages in the range [*addr*,*addr*+*len*) may have been changed.

**RETURN VALUE**

Upon successful completion, *mprotect()* shall return 0; otherwise, it shall return *-1* and set *errno* to indicate the error.

**ERRORS**

The *mprotect()* function shall fail if:

**[EACCES]** The *prot* argument specifies a protection that violates the access permission the process has to the underlying memory object.

**[EAGAIN]** The *prot* argument specifies *PROT\_WRITE* over a *MAP\_PRIVATE* mapping and there are insufficient memory resources to reserve for locking the private page.

**[ENOMEM]** Addresses in the range [*addr*,*addr*+*len*) are invalid for the address space of a process, or specify one or more pages which are not mapped.

**[ENOMEM]** The *prot* argument specifies *PROT\_WRITE* on a *MAP\_PRIVATE* mapping, and it would require more space than the system is able to supply for locking the private pages, if required.

44737 [ENOTSUP] The implementation does not support the combination of accesses requested  
44738 in the *prot* argument.

44739 The *mprotect()* function may fail if:

44740 [EINVAL] The *addr* argument is not a multiple of the page size as returned by *sysconf()*.

#### 44741 EXAMPLES

44742 None.

#### 44743 APPLICATION USAGE

44744 Most implementations require that *addr* is a multiple of the page size as returned by *sysconf()*.

#### 44745 RATIONALE

44746 None.

#### 44747 FUTURE DIRECTIONS

44748 None.

#### 44749 SEE ALSO

44750 *mmap()*, *sysconf()*

44751 XBD <sys/mman.h>

#### 44752 CHANGE HISTORY

44753 First released in Issue 4, Version 2.

#### 44754 Issue 5

44755 Moved from X/OPEN UNIX extension to BASE.

44756 Aligned with *mprotect()* in the POSIX Realtime Extension as follows:

- 44757 • The DESCRIPTION is largely reworded.
- 44758 • [ENOTSUP] and a second form of [ENOMEM] are added as mandatory error conditions.
- 44759 • [EAGAIN] is moved from the optional to the mandatory error conditions.

#### 44760 Issue 6

44761 The *mprotect()* function is marked as part of the Memory Protection option.

44762 The following new requirements on POSIX implementations derive from alignment with the  
44763 Single UNIX Specification:

- 44764 • The DESCRIPTION is updated to state that implementations require *addr* to be a multiple  
44765 of the page size as returned by *sysconf()*.
- 44766 • The [EINVAL] error condition is added.

44767 The normative text is updated to avoid use of the term “must” for application requirements.

#### 44768 Issue 7

44769 SD5-XSH-ERN-22 is applied, deleting erroneous APPLICATION USAGE.

44770 Austin Group Interpretation 1003.1-2001 #078 is applied, clarifying page alignment  
44771 requirements.

44772 The *mprotect()* function is moved from the Memory Protection option to the Base.

44773 **NAME**44774 mq\_close — close a message queue (**REALTIME**)44775 **SYNOPSIS**

```
44776 MSG #include <mqueue.h>
44777 int mq_close(mqd_t mqdes);
```

44778 **DESCRIPTION**

44779 The *mq\_close()* function shall remove the association between the message queue descriptor,  
 44780 *mqdes*, and its message queue. The results of using this message queue descriptor after successful  
 44781 return from this *mq\_close()*, and until the return of this message queue descriptor from a  
 44782 subsequent *mq\_open()*, are undefined.

44783 If the process has successfully attached a notification request to the message queue via this  
 44784 *mqdes*, this attachment shall be removed, and the message queue is available for another process  
 44785 to attach for notification.

44786 **RETURN VALUE**

44787 Upon successful completion, the *mq\_close()* function shall return a value of zero; otherwise, the  
 44788 function shall return a value of -1 and set *errno* to indicate the error.

44789 **ERRORS**

44790 The *mq\_close()* function shall fail if:

44791 [EBADF] The *mqdes* argument is not a valid message queue descriptor.

44792 **EXAMPLES**

44793 None.

44794 **APPLICATION USAGE**

44795 None.

44796 **RATIONALE**

44797 None.

44798 **FUTURE DIRECTIONS**

44799 None.

44800 **SEE ALSO**

44801 *mq\_open()*, *mq\_unlink()*, *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*

44802 XBD [<mqueue.h>](#)

44803 **CHANGE HISTORY**

44804 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

44805 **Issue 6**

44806 The *mq\_close()* function is marked as part of the Message Passing option.

44807 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
 44808 implementation does not support the Message Passing option.

44809 **NAME**44810 mq\_getattr — get message queue attributes (**REALTIME**)44811 **SYNOPSIS**

```
44812 MSG #include <mqueue.h>
44813 int mq_getattr(mqd_t mqdes, struct mq_attr *mqstat);
```

44814 **DESCRIPTION**

44815 The *mq\_getattr()* function shall obtain status information and attributes of the message queue  
44816 and the open message queue description associated with the message queue descriptor.

44817 The *mqdes* argument specifies a message queue descriptor.

44818 The results shall be returned in the **mq\_attr** structure referenced by the *mqstat* argument.

44819 Upon return, the following members shall have the values associated with the open message  
44820 queue description as set when the message queue was opened and as modified by subsequent  
44821 *mq\_setattr()* calls: *mq\_flags*.

44822 The following attributes of the message queue shall be returned as set at message queue  
44823 creation: *mq\_maxmsg*, *mq\_msgsize*.

44824 Upon return, the following members within the **mq\_attr** structure referenced by the *mqstat*  
44825 argument shall be set to the current state of the message queue:

44826 *mq\_curmsgs* The number of messages currently on the queue.

44827 **RETURN VALUE**

44828 Upon successful completion, the *mq\_getattr()* function shall return zero. Otherwise, the function  
44829 shall return  $-1$  and set *errno* to indicate the error.

44830 **ERRORS**

44831 The *mq\_getattr()* function may fail if:

44832 [EBADF] The *mqdes* argument is not a valid message queue descriptor.

44833 **EXAMPLES**

44834 See *mq\_notify()*.

44835 **APPLICATION USAGE**

44836 None.

44837 **RATIONALE**

44838 None.

44839 **FUTURE DIRECTIONS**

44840 None.

44841 **SEE ALSO**

44842 *mq\_notify()*, *mq\_open()*, *mq\_send()*, *mq\_setattr()*, *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*

44843 XBD **<mqueue.h>**

44844 **CHANGE HISTORY**

44845 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

44846 **Issue 6**

44847 The *mq\_getattr()* function is marked as part of the Message Passing option.

44848 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
44849 implementation does not support the Message Passing option.

44850 The *mq\_timedsend()* function is added to the SEE ALSO section for alignment with IEEE Std  
44851 1003.1d-1999.

44852 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/61 is applied, updating the ERRORS  
44853 section to change the [EBADF] error from mandatory to optional.

44854 **NAME**44855 mq\_notify — notify process that a message is available (**REALTIME**)44856 **SYNOPSIS**

```
44857 MSG    #include <mqueue.h>
44858        int mq_notify(mqd_t mqdes, const struct sigevent *notification);
```

44859 **DESCRIPTION**

44860 If the argument *notification* is not NULL, this function shall register the calling process to be  
44861 notified of message arrival at an empty message queue associated with the specified message  
44862 queue descriptor, *mqdes*. The notification specified by the *notification* argument shall be sent to  
44863 the process when the message queue transitions from empty to non-empty. At any time, only  
44864 one process may be registered for notification by a message queue. If the calling process or any  
44865 other process has already registered for notification of message arrival at the specified message  
44866 queue, subsequent attempts to register for that message queue shall fail.

44867 If *notification* is NULL and the process is currently registered for notification by the specified  
44868 message queue, the existing registration shall be removed.

44869 When the notification is sent to the registered process, its registration shall be removed. The  
44870 message queue shall then be available for registration.

44871 If a process has registered for notification of message arrival at a message queue and some  
44872 thread is blocked in *mq\_receive()* or *mq\_timedreceive()* waiting to receive a message when a  
44873 message arrives at the queue, the arriving message shall satisfy the appropriate *mq\_receive()* or  
44874 *mq\_timedreceive()*, respectively. The resulting behavior is as if the message queue remains empty,  
44875 and no notification shall be sent.

44876 **RETURN VALUE**

44877 Upon successful completion, the *mq\_notify()* function shall return a value of zero; otherwise, the  
44878 function shall return a value of -1 and set *errno* to indicate the error.

44879 **ERRORS**

44880 The *mq\_notify()* function shall fail if:

44881 [EBADF] The *mqdes* argument is not a valid message queue descriptor.

44882 [EBUSY] A process is already registered for notification by the message queue.

44883 The *mq\_notify()* function may fail if:

44884 [EINVAL] The *notification* argument is NULL and the process is currently not registered.

44885 **EXAMPLES**

44886 The following program registers a notification request for the message queue named in its  
44887 command-line argument. Notification is performed by creating a thread. The thread executes a  
44888 function which reads one message from the queue and then terminates the process.

```
44889 #include <pthread.h>
44890 #include <mqueue.h>
44891 #include <assert.h>
44892 #include <stdio.h>
44893 #include <stdlib.h>
44894 #include <unistd.h>

44895 static void                                /* Thread start function */
44896 tfunc(union sigval sv)
44897 {
```

```
44898     struct mq_attr attr;
44899     ssize_t nr;
44900     void *buf;
44901     mqd_t mqdes = *((mqd_t *) sv.sival_ptr);
44902     /* Determine maximum msg size; allocate buffer to receive msg */
44903     if (mq_getattr(mqdes, &attr) == -1) {
44904         perror("mq_getattr");
44905         exit(EXIT_FAILURE);
44906     }
44907     buf = malloc(attr.mq_msgsize);
44908     if (buf == NULL) {
44909         perror("malloc");
44910         exit(EXIT_FAILURE);
44911     }
44912     nr = mq_receive(mqdes, buf, attr.mq_msgsize, NULL);
44913     if (nr == -1) {
44914         perror("mq_receive");
44915         exit(EXIT_FAILURE);
44916     }
44917     printf("Read %ld bytes from message queue\n", (long) nr);
44918     free(buf);
44919     exit(EXIT_SUCCESS);          /* Terminate the process */
44920 }
44921 int
44922 main(int argc, char *argv[])
44923 {
44924     mqd_t mqdes;
44925     struct sigevent not;
44926     assert(argc == 2);
44927     mqdes = mq_open(argv[1], O_RDONLY);
44928     if (mqdes == (mqd_t) -1) {
44929         perror("mq_open");
44930         exit(EXIT_FAILURE);
44931     }
44932     not.sigev_notify = SIGEV_THREAD;
44933     not.sigev_notify_function = tfunc;
44934     not.sigev_notify_attributes = NULL;
44935     not.sigev_value.sival_ptr = &mqdes;    /* Arg. to thread func. */
44936     if (mq_notify(mqdes, &not) == -1) {
44937         perror("mq_notify");
44938         exit(EXIT_FAILURE);
44939     }
44940     pause();    /* Process will be terminated by thread function */
44941 }
```

44942 **APPLICATION USAGE**

44943 None.

44944 **RATIONALE**

44945 None.

44946 **FUTURE DIRECTIONS**

44947 None.

44948 **SEE ALSO**44949 *mq\_open()*, *mq\_send()*, *mq\_receive()*, *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*

44950 XBD &lt;mqqueue.h&gt;

44951 **CHANGE HISTORY**

44952 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

44953 **Issue 6**44954 The *mq\_notify()* function is marked as part of the Message Passing option.44955 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
44956 implementation does not support the Message Passing option.44957 The *mq\_timedsend()* function is added to the SEE ALSO section for alignment with IEEE Std  
44958 1003.1d-1999.44959 **Issue 7**44960 SD5-XSH-ERN-38 is applied, adding the *mq\_timedreceive()* function to the DESCRIPTION.

44961 Austin Group Interpretation 1003.1-2001 #032 is applied, adding the [EINVAL] error.

44962 An example is added.



44963 **NAME**44964 `mq_open` — open a message queue (**REALTIME**)44965 **SYNOPSIS**

```
44966 MSG      #include <mqueue.h>
44967      mqd_t mq_open(const char *name, int oflag, ...);
```

44968 **DESCRIPTION**

44969 The `mq_open()` function shall establish the connection between a process and a message queue  
 44970 with a message queue descriptor. It shall create an open message queue description that refers to  
 44971 the message queue, and a message queue descriptor that refers to that open message queue  
 44972 description. The message queue descriptor is used by other functions to refer to that message  
 44973 queue. The *name* argument points to a string naming a message queue. It is unspecified whether  
 44974 the name appears in the file system and is visible to other functions that take pathnames as  
 44975 arguments. The *name* argument conforms to the construction rules for a pathname, except that  
 44976 the interpretation of <slash> characters other than the leading <slash> character in *name* is  
 44977 implementation-defined, and that the length limits for the *name* argument are implementation-  
 44978 defined and need not be the same as the pathname limits {PATH\_MAX} and {NAME\_MAX}. If  
 44979 *name* begins with the <slash> character, then processes calling `mq_open()` with the same value of  
 44980 *name* shall refer to the same message queue object, as long as that name has not been removed. If  
 44981 *name* does not begin with the <slash> character, the effect is implementation-defined. If the *name*  
 44982 argument is not the name of an existing message queue and creation is not requested, `mq_open()`  
 44983 shall fail and return an error.

44984 A message queue descriptor may be implemented using a file descriptor, in which case  
 44985 applications can open up to at least {OPEN\_MAX} file and message queues.

44986 The *oflag* argument requests the desired receive and/or send access to the message queue. The  
 44987 requested access permission to receive messages or send messages shall be granted if the calling  
 44988 process would be granted read or write access, respectively, to an equivalently protected file.

44989 The value of *oflag* is the bitwise-inclusive OR of values from the following list. Applications  
 44990 shall specify exactly one of the first three values (access modes) below in the value of *oflag*:

44991 **O\_RDONLY**      Open the message queue for receiving messages. The process can use the  
 44992                    returned message queue descriptor with `mq_receive()`, but not `mq_send()`. A  
 44993                    message queue may be open multiple times in the same or different processes  
 44994                    for receiving messages.

44995 **O\_WRONLY**      Open the queue for sending messages. The process can use the returned  
 44996                    message queue descriptor with `mq_send()` but not `mq_receive()`. A message  
 44997                    queue may be open multiple times in the same or different processes for  
 44998                    sending messages.

44999 **O\_RDWR**        Open the queue for both receiving and sending messages. The process can use  
 45000                    any of the functions allowed for **O\_RDONLY** and **O\_WRONLY**. A message  
 45001                    queue may be open multiple times in the same or different processes for  
 45002                    sending messages.

45003 Any combination of the remaining flags may be specified in the value of *oflag*:

45004 **O\_CREAT**        Create a message queue. It requires two additional arguments: *mode*, which  
 45005                    shall be of type **mode\_t**, and *attr*, which shall be a pointer to an **mq\_attr**  
 45006                    structure. If the pathname *name* has already been used to create a message  
 45007                    queue that still exists, then this flag shall have no effect, except as noted under  
 45008                    **O\_EXCL**. Otherwise, a message queue shall be created without any messages

|       |                                                                                              |                                                                                                   |
|-------|----------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| 45009 |                                                                                              | in it. The user ID of the message queue shall be set to the effective user ID of                  |
| 45010 |                                                                                              | the process. The group ID of the message queue shall be set to the effective                      |
| 45011 |                                                                                              | group ID of the process; however, if the <i>name</i> argument is visible in the file              |
| 45012 |                                                                                              | system, the group ID may be set to the group ID of the containing directory.                      |
| 45013 |                                                                                              | When bits in <i>mode</i> other than the file permission bits are specified, the effect is         |
| 45014 |                                                                                              | unspecified. If <i>attr</i> is NULL, the message queue shall be created with                      |
| 45015 |                                                                                              | implementation-defined default message queue attributes. If <i>attr</i> is non-NULL               |
| 45016 |                                                                                              | and the calling process has appropriate privileges on <i>name</i> , the message queue             |
| 45017 |                                                                                              | <i>mq_maxmsg</i> and <i>mq_msgsize</i> attributes shall be set to the values of the               |
| 45018 |                                                                                              | corresponding members in the <b>mq_attr</b> structure referred to by <i>attr</i> . The            |
| 45019 |                                                                                              | values of the <i>mq_flags</i> and <i>mq_curmsgs</i> members of the <b>mq_attr</b> structure shall |
| 45020 |                                                                                              | be ignored. If <i>attr</i> is non-NULL, but the calling process does not have                     |
| 45021 |                                                                                              | appropriate privileges on <i>name</i> , the <i>mq_open()</i> function shall fail and return an    |
| 45022 |                                                                                              | error without creating the message queue.                                                         |
| 45023 | O_EXCL                                                                                       | If O_EXCL and O_CREAT are set, <i>mq_open()</i> shall fail if the message queue                   |
| 45024 |                                                                                              | <i>name</i> exists. The check for the existence of the message queue and the creation             |
| 45025 |                                                                                              | of the message queue if it does not exist shall be atomic with respect to other                   |
| 45026 |                                                                                              | threads executing <i>mq_open()</i> naming the same <i>name</i> with O_EXCL and                    |
| 45027 |                                                                                              | O_CREAT set. If O_EXCL is set and O_CREAT is not set, the result is                               |
| 45028 |                                                                                              | undefined.                                                                                        |
| 45029 | O_NONBLOCK                                                                                   | Determines whether an <i>mq_send()</i> or <i>mq_receive()</i> waits for resources or              |
| 45030 |                                                                                              | messages that are not currently available, or fails with <i>errno</i> set to [EAGAIN];            |
| 45031 |                                                                                              | see <i>mq_send()</i> and <i>mq_receive()</i> for details.                                         |
| 45032 | The <i>mq_open()</i> function does not add or remove messages from the queue.                |                                                                                                   |
| 45033 | <b>RETURN VALUE</b>                                                                          |                                                                                                   |
| 45034 | Upon successful completion, the function shall return a message queue descriptor; otherwise, |                                                                                                   |
| 45035 | the function shall return ( <b>mqd_t</b> )−1 and set <i>errno</i> to indicate the error.     |                                                                                                   |
| 45036 | <b>ERRORS</b>                                                                                |                                                                                                   |
| 45037 | The <i>mq_open()</i> function shall fail if:                                                 |                                                                                                   |
| 45038 | [EACCES]                                                                                     | The message queue exists and the permissions specified by <i>oflag</i> are denied, or             |
| 45039 |                                                                                              | the message queue does not exist and permission to create the message queue                       |
| 45040 |                                                                                              | is denied.                                                                                        |
| 45041 | [EEXIST]                                                                                     | O_CREAT and O_EXCL are set and the named message queue already exists.                            |
| 45042 | [EINTR]                                                                                      | The <i>mq_open()</i> function was interrupted by a signal.                                        |
| 45043 | [EINVAL]                                                                                     | The <i>mq_open()</i> function is not supported for the given name.                                |
| 45044 | [EINVAL]                                                                                     | O_CREAT was specified in <i>oflag</i> , the value of <i>attr</i> is not NULL, and either          |
| 45045 |                                                                                              | <i>mq_maxmsg</i> or <i>mq_msgsize</i> was less than or equal to zero.                             |
| 45046 | [EMFILE]                                                                                     | Too many message queue descriptors or file descriptors are currently in use by                    |
| 45047 |                                                                                              | this process.                                                                                     |
| 45048 | [ENFILE]                                                                                     | Too many message queues are currently open in the system.                                         |
| 45049 | [ENOENT]                                                                                     | O_CREAT is not set and the named message queue does not exist.                                    |
| 45050 | [ENOSPC]                                                                                     | There is insufficient space for the creation of the new message queue.                            |

45051 If any of the following conditions occur, the *mq\_open()* function may return (**mqd\_t**)–1 and set  
 45052 *errno* to the corresponding value.

45053 [ENAMETOOLONG]

45054 The length of the *name* argument exceeds {\_POSIX\_PATH\_MAX} on systems  
 45055 XSI that do not support the XSI option or exceeds {\_XOPEN\_PATH\_MAX} on XSI  
 45056 systems, or has a pathname component that is longer than  
 45057 XSI {\_POSIX\_NAME\_MAX} on systems that do not support the XSI option or  
 45058 longer than {\_XOPEN\_NAME\_MAX} on XSI systems.

#### 45059 EXAMPLES

45060 None.

#### 45061 APPLICATION USAGE

45062 None.

#### 45063 RATIONALE

45064 None.

#### 45065 FUTURE DIRECTIONS

45066 A future version might require the *mq\_open()* and *mq\_unlink()* functions to have semantics  
 45067 similar to normal file system operations.

#### 45068 SEE ALSO

45069 *mq\_close()*, *mq\_getattr()*, *mq\_receive()*, *mq\_send()*, *mq\_setattr()*, *mq\_unlink()*, *msgctl()*, *msgget()*,  
 45070 *msgrcv()*, *msgsnd()*

45071 XBD <mqqueue.h>

#### 45072 CHANGE HISTORY

45073 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

#### 45074 Issue 6

45075 The *mq\_open()* function is marked as part of the Message Passing option.

45076 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
 45077 implementation does not support the Message Passing option.

45078 The *mq\_timedreceive()* and *mq\_timedsend()* functions are added to the SEE ALSO section for  
 45079 alignment with IEEE Std 1003.1d-1999.

45080 The DESCRIPTION of O\_EXCL is updated in response to IEEE PASC Interpretation 1003.1c #48.

45081 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/62 is applied, updating the description of  
 45082 the permission bits in the DESCRIPTION. The change is made for consistency with the  
 45083 *shm\_open()* and *sem\_open()* functions.

#### 45084 Issue 7

45085 Austin Group Interpretation 1003.1-2001 #077 is applied, clarifying the *name* argument and  
 45086 changing [ENAMETOOLONG] from a “shall fail” to a “may fail” error.

45087 Austin Group Interpretation 1003.1-2001 #141 is applied, adding FUTURE DIRECTIONS.

45088 SD5-XSH-ERN-170 is applied, updating the DESCRIPTION to clarify the wording for setting the  
 45089 user ID and group ID of the message queue.

45090 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0394 [259] is applied.

## NAME

mq\_receive, mq\_timedreceive — receive a message from a message queue (REALTIME)

## SYNOPSIS

```
#include <mqueue.h>

ssize_t mq_receive(mqd_t mqdes, char *msg_ptr, size_t msg_len,
    unsigned *msg_prio);

#include <mqueue.h>
#include <time.h>

ssize_t mq_timedreceive(mqd_t mqdes, char *restrict msg_ptr,
    size_t msg_len, unsigned *restrict msg_prio,
    const struct timespec *restrict abstime);
```

## DESCRIPTION

The *mq\_receive()* function shall receive the oldest of the highest priority message(s) from the message queue specified by *mqdes*. If the size of the buffer in bytes, specified by the *msg\_len* argument, is less than the *mq\_msgsize* attribute of the message queue, the function shall fail and return an error. Otherwise, the selected message shall be removed from the queue and copied to the buffer pointed to by the *msg\_ptr* argument.

If the value of *msg\_len* is greater than {SSIZE\_MAX}, the result is implementation-defined.

If the argument *msg\_prio* is not NULL, the priority of the selected message shall be stored in the location referenced by *msg\_prio*.

If the specified message queue is empty and O\_NONBLOCK is not set in the message queue description associated with *mqdes*, *mq\_receive()* shall block until a message is enqueued on the message queue or until *mq\_receive()* is interrupted by a signal. If more than one thread is waiting to receive a message when a message arrives at an empty queue and the Priority Scheduling option is supported, then the thread of highest priority that has been waiting the longest shall be selected to receive the message. Otherwise, it is unspecified which waiting thread receives the message. If the specified message queue is empty and O\_NONBLOCK is set in the message queue description associated with *mqdes*, no message shall be removed from the queue, and *mq\_receive()* shall return an error.

The *mq\_timedreceive()* function shall receive the oldest of the highest priority messages from the message queue specified by *mqdes* as described for the *mq\_receive()* function. However, if O\_NONBLOCK was not specified when the message queue was opened via the *mq\_open()* function, and no message exists on the queue to satisfy the receive, the wait for such a message shall be terminated when the specified timeout expires. If O\_NONBLOCK is set, this function is equivalent to *mq\_receive()*.

The timeout expires when the absolute time specified by *abstime* passes, as measured by the clock on which timeouts are based (that is, when the value of that clock equals or exceeds *abstime*), or if the absolute time specified by *abstime* has already been passed at the time of the call.

The timeout shall be based on the CLOCK\_REALTIME clock. The resolution of the timeout shall be the resolution of the clock on which it is based. The *timespec* argument is defined in the **<time.h>** header.

Under no circumstance shall the operation fail with a timeout if a message can be removed from the message queue immediately. The validity of the *abstime* parameter need not be checked if a message can be removed from the message queue immediately.

**RETURN VALUE**

Upon successful completion, the *mq\_receive()* and *mq\_timedreceive()* functions shall return the length of the selected message in bytes and the message shall be removed from the queue. Otherwise, no message shall be removed from the queue, the functions shall return a value of -1, and set *errno* to indicate the error.

**ERRORS**

These functions shall fail if:

- [EAGAIN] O\_NONBLOCK was set in the message description associated with *mqdes*, and the specified message queue is empty.
- [EBADF] The *mqdes* argument is not a valid message queue descriptor open for reading.
- [EMSGSIZE] The specified message buffer size, *msg\_len*, is less than the message size attribute of the message queue.
- [EINTR] The *mq\_receive()* or *mq\_timedreceive()* operation was interrupted by a signal.
- [EINVAL] The process or thread would have blocked, and the *abstime* parameter specified a nanoseconds field value less than zero or greater than or equal to 1 000 million.
- [ETIMEDOUT] The O\_NONBLOCK flag was not set when the message queue was opened, but no message arrived on the queue before the specified timeout expired.

These functions may fail if:

- [EBADMSG] The implementation has detected a data corruption problem with the message.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*mq\_open()*, *mq\_send()*, *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*, *time()*

XBD <mqqueue.h>, <time.h>

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**Issue 6**

The *mq\_receive()* function is marked as part of the Message Passing option.

The Open Group Corrigendum U021/4 is applied. The DESCRIPTION is changed to refer to *msg\_len* rather than *maxsize*.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Message Passing option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In this function it is possible for the return value to exceed the range of the type **ssize\_t** (since **size\_t** has a larger range of positive values than **ssize\_t**). A sentence restricting the size of the **size\_t** object is added to the description to resolve this conflict.

The *mq\_timedreceive()* function is added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the *mq\_timedreceive()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE PASC Interpretation 1003.1 #109 is applied, correcting the return type for *mq\_timedreceive()* from **int** to **ssize\_t**.

#### Issue 7

The *mq\_timedreceive()* function is moved from the Timeouts option to the Base.

Functionality relating to the Timers option is moved to the Base.



45189 **NAME**45190 `mq_send`, `mq_timedsend` — send a message to a message queue (**REALTIME**)45191 **SYNOPSIS**

```
45192 MSG    #include <mqueue.h>
45193
45193     int mq_send(mqd_t mqdes, const char *msg_ptr, size_t msg_len,
45194               unsigned msg_prio);
45195
45195     #include <mqueue.h>
45196     #include <time.h>
45197
45197     int mq_timedsend(mqd_t mqdes, const char *msg_ptr, size_t msg_len,
45198                   unsigned msg_prio, const struct timespec *abstime);
```

45199 **DESCRIPTION**

45200 The `mq_send()` function shall add the message pointed to by the argument `msg_ptr` to the  
45201 message queue specified by `mqdes`. The `msg_len` argument specifies the length of the message, in  
45202 bytes, pointed to by `msg_ptr`. The value of `msg_len` shall be less than or equal to the `mq_msgsize`  
45203 attribute of the message queue, or `mq_send()` shall fail.

45204 If the specified message queue is not full, `mq_send()` shall behave as if the message is inserted  
45205 into the message queue at the position indicated by the `msg_prio` argument. A message with a  
45206 larger numeric value of `msg_prio` shall be inserted before messages with lower values of  
45207 `msg_prio`. A message shall be inserted after other messages in the queue, if any, with equal  
45208 `msg_prio`. The value of `msg_prio` shall be less than {MQ\_PRIO\_MAX}.

45209 If the specified message queue is full and O\_NONBLOCK is not set in the message queue  
45210 description associated with `mqdes`, `mq_send()` shall block until space becomes available to  
45211 enqueue the message, or until `mq_send()` is interrupted by a signal. If more than one thread is  
45212 waiting to send when space becomes available in the message queue and the Priority Scheduling  
45213 option is supported, then the thread of the highest priority that has been waiting the longest  
45214 shall be unblocked to send its message. Otherwise, it is unspecified which waiting thread is  
45215 unblocked. If the specified message queue is full and O\_NONBLOCK is set in the message  
45216 queue description associated with `mqdes`, the message shall not be queued and `mq_send()` shall  
45217 return an error.

45218 The `mq_timedsend()` function shall add a message to the message queue specified by `mqdes` in the  
45219 manner defined for the `mq_send()` function. However, if the specified message queue is full and  
45220 O\_NONBLOCK is not set in the message queue description associated with `mqdes`, the wait for  
45221 sufficient room in the queue shall be terminated when the specified timeout expires. If  
45222 O\_NONBLOCK is set in the message queue description, this function shall be equivalent to  
45223 `mq_send()`.

45224 The timeout shall expire when the absolute time specified by `abstime` passes, as measured by the  
45225 clock on which timeouts are based (that is, when the value of that clock equals or exceeds  
45226 `abstime`), or if the absolute time specified by `abstime` has already been passed at the time of the  
45227 call.

45228 The timeout shall be based on the CLOCK\_REALTIME clock. The resolution of the timeout shall  
45229 be the resolution of the clock on which it is based. The `timespec` argument is defined in the  
45230 **<time.h>** header.

45231 Under no circumstance shall the operation fail with a timeout if there is sufficient room in the  
45232 queue to add the message immediately. The validity of the `abstime` parameter need not be  
45233 checked when there is sufficient room in the queue.

**RETURN VALUE**

Upon successful completion, the `mq_send()` and `mq_timedsend()` functions shall return a value of zero. Otherwise, no message shall be enqueued, the functions shall return `-1`, and `errno` shall be set to indicate the error.

**ERRORS**

The `mq_send()` and `mq_timedsend()` functions shall fail if:

[EAGAIN] The `O_NONBLOCK` flag is set in the message queue description associated with `mqdes`, and the specified message queue is full.

[EBADF] The `mqdes` argument is not a valid message queue descriptor open for writing.

[EINTR] A signal interrupted the call to `mq_send()` or `mq_timedsend()`.

[EINVAL] The value of `msg_prio` was outside the valid range.

[EINVAL] The process or thread would have blocked, and the `abstime` parameter specified a nanoseconds field value less than zero or greater than or equal to 1 000 million.

[EMSGSIZE] The specified message length, `msg_len`, exceeds the message size attribute of the message queue.

[ETIMEDOUT] The `O_NONBLOCK` flag was not set when the message queue was opened, but the timeout expired before the message could be added to the queue.

**EXAMPLES**

None.

**APPLICATION USAGE**

The value of the symbol `{MQ_PRIO_MAX}` limits the number of priority levels supported by the application. Message priorities range from 0 to `{MQ_PRIO_MAX}-1`.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

`mq_open()`, `mq_receive()`, `mq_setattr()`, `time()`

XBD `<mqqueue.h>`, `<time.h>`

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**Issue 6**

The `mq_send()` function is marked as part of the Message Passing option.

The `[ENOSYS]` error condition has been removed as stubs need not be provided if an implementation does not support the Message Passing option.

The `mq_timedsend()` function is added for alignment with IEEE Std 1003.1d-1999.

**Issue 7**

The `mq_timedsend()` function is moved from the Timeouts option to the Base.

Functionality relating to the Timers option is moved to the Base.



45274 **NAME**45275 mq\_setattr — set message queue attributes (**REALTIME**)45276 **SYNOPSIS**

```
45277 MSG    #include <mqueue.h>
45278
45278     int mq_setattr(mqd_t mqdes, const struct mq_attr *restrict mqstat,
45279                   struct mq_attr *restrict omqstat);
```

45280 **DESCRIPTION**

45281 The *mq\_setattr()* function shall set attributes associated with the open message queue  
 45282 description referenced by the message queue descriptor specified by *mqdes*.

45283 The message queue attributes corresponding to the following members defined in the **mq\_attr**  
 45284 structure shall be set to the specified values upon successful completion of *mq\_setattr()*:

45285 *mq\_flags*           The value of this member is the bitwise-logical OR of zero or more of  
 45286 O\_NONBLOCK and any implementation-defined flags.

45287 The values of the *mq\_maxmsg*, *mq\_msgsize*, and *mq\_curmsgs* members of the **mq\_attr** structure  
 45288 shall be ignored by *mq\_setattr()*.

45289 If *omqstat* is non-NULL, the *mq\_setattr()* function shall store, in the location referenced by  
 45290 *omqstat*, the previous message queue attributes and the current queue status. These values shall  
 45291 be the same as would be returned by a call to *mq\_getattr()* at that point.

45292 **RETURN VALUE**

45293 Upon successful completion, the function shall return a value of zero and the attributes of the  
 45294 message queue shall have been changed as specified.

45295 Otherwise, the message queue attributes shall be unchanged, and the function shall return a  
 45296 value of -1 and set *errno* to indicate the error.

45297 **ERRORS**

45298 The *mq\_setattr()* function shall fail if:

45299 [EBADF]           The *mqdes* argument is not a valid message queue descriptor.

45300 **EXAMPLES**

45301 None.

45302 **APPLICATION USAGE**

45303 None.

45304 **RATIONALE**

45305 None.

45306 **FUTURE DIRECTIONS**

45307 None.

45308 **SEE ALSO**

45309 *mq\_open()*, *mq\_send()*, *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*

45310 XBD **<mqueue.h>**

45311 **CHANGE HISTORY**

45312 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

45313 **Issue 6**

45314 The *mq\_setattr()* function is marked as part of the Message Passing option.

45315 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
45316 implementation does not support the Message Passing option.

45317 The *mq\_timedsend()* function is added to the SEE ALSO section for alignment with IEEE Std  
45318 1003.1d-1999.

45319 The **restrict** keyword is added to the *mq\_setattr()* prototype for alignment with the  
45320 ISO/IEC 9899:1999 standard.

45321 **NAME**45322 `mq_timedreceive` — receive a message from a message queue (**ADVANCED REALTIME**)45323 **SYNOPSIS**

```
45324 MSG      #include <mqueue.h>
45325           #include <time.h>
45326           ssize_t mq_timedreceive(mqd_t mqdes, char *restrict msg_ptr,
45327                                   size_t msg_len, unsigned *restrict msg_prio,
45328                                   const struct timespec *restrict abstime);
```

45329 **DESCRIPTION**45330 Refer to *mq\_receive()*.

45331 **NAME**45332 mq\_timedsend — send a message to a message queue (**ADVANCED REALTIME**)45333 **SYNOPSIS**

```
45334 MSG    #include <mqueue.h>
45335        #include <time.h>
45336        int mq_timedsend(mqd_t mqdes, const char *msg_ptr, size_t msg_len,
45337                        unsigned msg_prio, const struct timespec *abstime);
```

45338 **DESCRIPTION**45339 Refer to *mq\_send()*.

45340 **NAME**45341 `mq_unlink` — remove a message queue (**REALTIME**)45342 **SYNOPSIS**

```
45343 MSG    #include <mqueue.h>
45344        int mq_unlink(const char *name);
```

45345 **DESCRIPTION**

45346 The `mq_unlink()` function shall remove the message queue named by the string `name`. If one or  
45347 more processes have the message queue open when `mq_unlink()` is called, destruction of the  
45348 message queue shall be postponed until all references to the message queue have been closed.  
45349 However, the `mq_unlink()` call need not block until all references have been closed; it may return  
45350 immediately.

45351 After a successful call to `mq_unlink()`, reuse of the name shall subsequently cause `mq_open()` to  
45352 behave as if no message queue of this name exists (that is, `mq_open()` will fail if `O_CREAT` is not  
45353 set, or will create a new message queue if `O_CREAT` is set).

45354 **RETURN VALUE**

45355 Upon successful completion, the function shall return a value of zero. Otherwise, the named  
45356 message queue shall be unchanged by this function call, and the function shall return a value of  
45357 `-1` and set `errno` to indicate the error.

45358 **ERRORS**45359 The `mq_unlink()` function shall fail if:

45360 [EACCES] Permission is denied to unlink the named message queue.

45361 [EINTR] The call to `mq_unlink()` blocked waiting for all references to the named +  
45362 message queue to be closed and a signal interrupted the call.

45363 [ENOENT] The named message queue does not exist.

45364 The `mq_unlink()` function may fail if:

45365 [ENAMETOOLONG]

45366 The length of the `name` argument exceeds `{_POSIX_PATH_MAX}` on systems  
45367 XSI that do not support the XSI option or exceeds `{_XOPEN_PATH_MAX}` on XSI  
45368 systems, or has a pathname component that is longer than  
45369 XSI `{_POSIX_NAME_MAX}` on systems that do not support the XSI option or  
45370 longer than `{_XOPEN_NAME_MAX}` on XSI systems. A call to `mq_unlink()`  
45371 with a `name` argument that contains the same message queue name as was  
45372 previously used in a successful `mq_open()` call shall not give an  
45373 [ENAMETOOLONG] error.

45374 **EXAMPLES**

45375 None.

45376 **APPLICATION USAGE**

45377 None.

45378 **RATIONALE**

45379 None.

**FUTURE DIRECTIONS**

A future version might require the *mq\_open()* and *mq\_unlink()* functions to have semantics similar to normal file system operations.

**SEE ALSO**

*mq\_close()*, *mq\_open()*, *msgctl()*, *msgget()*, *msgrcv()*, *msgsnd()*

XBD <mqqueue.h>

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**Issue 6**

The *mq\_unlink()* function is marked as part of the Message Passing option.

The Open Group Corrigendum U021/5 is applied, clarifying that upon unsuccessful completion, the named message queue is unchanged by this function.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Message Passing option.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #077 is applied, changing [ENAMETOOLONG] from a “shall fail” to a “may fail” error .

Austin Group Interpretation 1003.1-2001 #141 is applied.

45398 **NAME**

45399       rand48 — generate uniformly distributed pseudo-random signed long integers

45400 **SYNOPSIS**

```
45401 XSI      #include <stdlib.h>
45402          long rand48(void);
```

45403 **DESCRIPTION**45404       Refer to *rand48()*.

45405 **NAME**

45406 msgctl — XSI message control operations

45407 **SYNOPSIS**

```
45408 XSI    #include <sys/msg.h>
45409    int msgctl(int msqid, int cmd, struct msqid_ds *buf);
```

45410 **DESCRIPTION**

45411 The *msgctl()* function operates on XSI message queues (see XBD [Section 3.226](#), on page 69). It is  
 45412 unspecified whether this function interoperates with the realtime interprocess communication  
 45413 facilities defined in [Section 2.8](#) (on page 503).

45414 The *msgctl()* function shall provide message control operations as specified by *cmd*. The  
 45415 following values for *cmd*, and the message control operations they specify, are:

45416 **IPC\_STAT** Place the current value of each member of the **msqid\_ds** data structure  
 45417 associated with *msqid* into the structure pointed to by *buf*. The contents of this  
 45418 structure are defined in **<sys/msg.h>**.

45419 **IPC\_SET** Set the value of the following members of the **msqid\_ds** data structure  
 45420 associated with *msqid* to the corresponding value found in the structure  
 45421 pointed to by *buf*:

```
45422 msg_perm.uid
45423 msg_perm.gid
45424 msg_perm.mode
45425 msg_qbytes
```

45426 Also, the *msg\_ctime* timestamp shall be set to the current time, as described in  
 45427 [Section 2.7.1](#) (on page 502).

45428 IPC\_SET can only be executed by a process with appropriate privileges or that  
 45429 has an effective user ID equal to the value of **msg\_perm.cuid** or  
 45430 **msg\_perm.uid** in the **msqid\_ds** data structure associated with *msqid*. Only a  
 45431 process with appropriate privileges can raise the value of **msg\_qbytes**.

45432 **IPC\_RMID** Remove the message queue identifier specified by *msqid* from the system and  
 45433 destroy the message queue and **msqid\_ds** data structure associated with it.  
 45434 IPC\_RMID can only be executed by a process with appropriate privileges or  
 45435 one that has an effective user ID equal to the value of **msg\_perm.cuid** or  
 45436 **msg\_perm.uid** in the **msqid\_ds** data structure associated with *msqid*.

45437 **RETURN VALUE**

45438 Upon successful completion, *msgctl()* shall return 0; otherwise, it shall return -1 and set *errno* to  
 45439 indicate the error.

45440 **ERRORS**

45441 The *msgctl()* function shall fail if:

45442 [EACCES] The argument *cmd* is IPC\_STAT and the calling process does not have read  
 45443 permission; see [Section 2.7](#) (on page 501).

45444 [EINVAL] The value of *msqid* is not a valid message queue identifier; or the value of *cmd*  
 45445 is not a valid command.

45446 [EPERM] The argument *cmd* is IPC\_RMID or IPC\_SET and the effective user ID of the  
 45447 calling process is not equal to that of a process with appropriate privileges and  
 45448 it is not equal to the value of **msg\_perm.cuid** or **msg\_perm.uid** in the data



45449 structure associated with *msqid*.

45450 [EPERM] The argument *cmd* is IPC\_SET, an attempt is being made to increase to the  
45451 value of **msg\_qbytes**, and the effective user ID of the calling process does not  
45452 have appropriate privileges.

#### 45453 EXAMPLES

45454 None.

#### 45455 APPLICATION USAGE

45456 The POSIX Realtime Extension defines alternative interfaces for interprocess communication  
45457 (IPC). Application developers who need to use IPC should design their applications so that  
45458 modules using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to  
45459 use the alternative interfaces.

#### 45460 RATIONALE

45461 None.

#### 45462 FUTURE DIRECTIONS

45463 None.

#### 45464 SEE ALSO

45465 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 503), [mq\\_close\(\)](#), [mq\\_getattr\(\)](#), [mq\\_notify\(\)](#),  
45466 [mq\\_open\(\)](#), [mq\\_receive\(\)](#), [mq\\_send\(\)](#), [mq\\_setattr\(\)](#), [mq\\_unlink\(\)](#), [msgget\(\)](#), [msgrcv\(\)](#), [msgsnd\(\)](#)

45467 XBD [Section 3.226](#) (on page 69), [<sys/msg.h>](#)

#### 45468 CHANGE HISTORY

45469 First released in Issue 2. Derived from Issue 2 of the SVID.

#### 45470 Issue 5

45471 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE  
45472 DIRECTIONS to a new APPLICATION USAGE section.

#### 45473 Issue 7

45474 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0395 [345] is applied.

**NAME**

msgget — get the XSI message queue identifier

**SYNOPSIS**

```
XSI    #include <sys/msg.h>
      int msgget(key_t key, int msgflg);
```

**DESCRIPTION**

The `msgget()` function operates on XSI message queues (see XBD [Section 3.226](#), on page 69). It is unspecified whether this function interoperates with the realtime interprocess communication facilities defined in [Section 2.8](#) (on page 503).

The `msgget()` function shall return the message queue identifier associated with the argument `key`.

A message queue identifier, associated message queue, and data structure (see `<sys/msg.h>`), shall be created for the argument `key` if one of the following is true:

- The argument `key` is equal to `IPC_PRIVATE`.
- The argument `key` does not already have a message queue identifier associated with it, and (`msgflg & IPC_CREAT`) is non-zero.

Upon creation, the data structure associated with the new message queue identifier shall be initialized as follows:

- `msg_perm.cuid`, `msg_perm.uid`, `msg_perm.cgid`, and `msg_perm.gid` shall be set to the effective user ID and effective group ID, respectively, of the calling process.
- The low-order 9 bits of `msg_perm.mode` shall be set to the low-order 9 bits of `msgflg`.
- `msg_qnum`, `msg_lspid`, `msg_lrpid`, `msg_stime`, and `msg_rtime` shall be set to 0.
- `msg_ctime` shall be set to the current time, as described in [Section 2.7.1](#) (on page 502).
- `msg_qbytes` shall be set to the system limit.

**RETURN VALUE**

Upon successful completion, `msgget()` shall return a non-negative integer, namely a message queue identifier. Otherwise, it shall return `-1` and set `errno` to indicate the error.

**ERRORS**

The `msgget()` function shall fail if:

- |          |                                                                                                                                                                                                                                 |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [EACCES] | A message queue identifier exists for the argument <code>key</code> , but operation permission as specified by the low-order 9 bits of <code>msgflg</code> would not be granted; see <a href="#">Section 2.7</a> (on page 501). |
| [EEXIST] | A message queue identifier exists for the argument <code>key</code> but <code>((msgflg &amp; IPC_CREAT) &amp;&amp; (msgflg &amp; IPC_EXCL))</code> is non-zero.                                                                 |
| [ENOENT] | A message queue identifier does not exist for the argument <code>key</code> and <code>(msgflg &amp; IPC_CREAT)</code> is 0.                                                                                                     |
| [ENOSPC] | A message queue identifier is to be created but the system-imposed limit on the maximum number of allowed message queue identifiers system-wide would be exceeded.                                                              |

45514 **EXAMPLES**

45515 None.

45516 **APPLICATION USAGE**

45517 The POSIX Realtime Extension defines alternative interfaces for interprocess communication  
 45518 (IPC). Application developers who need to use IPC should design their applications so that  
 45519 modules using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to  
 45520 use the alternative interfaces.

45521 **RATIONALE**

45522 None.

45523 **FUTURE DIRECTIONS**

45524 None.

45525 **SEE ALSO**

45526 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 503), [ftok\(\)](#), [mq\\_close\(\)](#), [mq\\_getattr\(\)](#), [mq\\_notify\(\)](#),  
 45527 [mq\\_open\(\)](#), [mq\\_receive\(\)](#), [mq\\_send\(\)](#), [mq\\_setattr\(\)](#), [mq\\_unlink\(\)](#), [msgctl\(\)](#), [msgrcv\(\)](#), [msgsnd\(\)](#)

45528 XBD [Section 3.226](#) (on page 69), [<sys/msg.h>](#)45529 **CHANGE HISTORY**

45530 First released in Issue 2. Derived from Issue 2 of the SVID.

45531 **Issue 5**

45532 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE  
 45533 DIRECTIONS to a new APPLICATION USAGE section.

45534 **Issue 7**

45535 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0396 [345] and XSH/TC1-2008/0397  
 45536 [344] are applied.

**NAME**

msgrcv — XSI message receive operation

**SYNOPSIS**

```
XSI    #include <sys/msg.h>

      ssize_t msgrcv(int msqid, void *msgp, size_t msgsz, long msgtyp,
      int msgflg);
```

**DESCRIPTION**

The *msgrcv()* function operates on XSI message queues (see XBD [Section 3.226](#), on page 69). It is unspecified whether this function interoperates with the realtime interprocess communication facilities defined in [Section 2.8](#) (on page 503).

The *msgrcv()* function shall read a message from the queue associated with the message queue identifier specified by *msqid* and place it in the user-defined buffer pointed to by *msgp*.

The application shall ensure that the argument *msgp* points to a user-defined buffer that contains first a field of type **long** specifying the type of the message, and then a data portion that holds the data bytes of the message. The structure below is an example of what this user-defined buffer might look like:

```
struct mymsg {
    long    mtype;    /* Message type. */
    char    mtext[1]; /* Message text. */
}
```

The structure member *mtype* is the received message's type as specified by the sending process.

The structure member *mtext* is the text of the message.

The argument *msgsz* specifies the size in bytes of *mtext*. The received message shall be truncated to *msgsz* bytes if it is larger than *msgsz* and (*msgflg* & MSG\_NOERROR) is non-zero. The truncated part of the message shall be lost and no indication of the truncation shall be given to the calling process.

If the value of *msgsz* is greater than {SSIZE\_MAX}, the result is implementation-defined.

The argument *msgtyp* specifies the type of message requested as follows:

- If *msgtyp* is 0, the first message on the queue shall be received.
- If *msgtyp* is greater than 0, the first message of type *msgtyp* shall be received.
- If *msgtyp* is less than 0, the first message of the lowest type that is less than or equal to the absolute value of *msgtyp* shall be received.

The argument *msgflg* specifies the action to be taken if a message of the desired type is not on the queue. These are as follows:

- If (*msgflg* & IPC\_NOWAIT) is non-zero, the calling thread shall return immediately with a return value of -1 and *errno* set to [ENOMSG].
- If (*msgflg* & IPC\_NOWAIT) is 0, the calling thread shall suspend execution until one of the following occurs:
  - A message of the desired type is placed on the queue.
  - The message queue identifier *msqid* is removed from the system; when this occurs, *errno* shall be set to [EIDRM] and -1 shall be returned.

45578 — The calling thread receives a signal that is to be caught; in this case a message is not  
 45579 received and the calling thread resumes execution in the manner prescribed in  
 45580 *sigaction()*.

45581 Upon successful completion, the following actions are taken with respect to the data structure  
 45582 associated with *msqid*:

- 45583 • **msg\_qnum** shall be decremented by 1.
- 45584 • **msg\_lrpid** shall be set to the process ID of the calling process.
- 45585 • **msg\_rtime** shall be set to the current time, as described in [Section 2.7.1](#) (on page 502).

#### 45586 RETURN VALUE

45587 Upon successful completion, *msgrcv()* shall return a value equal to the number of bytes actually  
 45588 placed into the buffer *mtext*. Otherwise, no message shall be received, *msgrcv()* shall return `-1`,  
 45589 and *errno* shall be set to indicate the error.

#### 45590 ERRORS

45591 The *msgrcv()* function shall fail if:

- |       |          |                                                                                                         |
|-------|----------|---------------------------------------------------------------------------------------------------------|
| 45592 | [E2BIG]  | The value of <i>mtext</i> is greater than <i>msgsz</i> and ( <i>msgflg</i> & MSG_NOERROR) is 0.         |
| 45593 | [EACCES] | Operation permission is denied to the calling process; see <a href="#">Section 2.7</a> (on page 501).   |
| 45594 |          |                                                                                                         |
| 45595 | [EIDRM]  | The message queue identifier <i>msqid</i> is removed from the system.                                   |
| 45596 | [EINTR]  | The <i>msgrcv()</i> function was interrupted by a signal.                                               |
| 45597 | [EINVAL] | <i>msqid</i> is not a valid message queue identifier.                                                   |
| 45598 | [ENOMSG] | The queue does not contain a message of the desired type and ( <i>msgflg</i> & IPC_NOWAIT) is non-zero. |
| 45599 |          |                                                                                                         |

#### 45600 EXAMPLES

##### 45601 Receiving a Message

45602 The following example receives the first message on the queue (based on the value of the *msgtyp*  
 45603 argument, 0). The queue is identified by the *msqid* argument (assuming that the value has  
 45604 previously been set). This call specifies that an error should be reported if no message is  
 45605 available, but not if the message is too large. The message size is calculated directly using the  
 45606 *sizeof* operator.

```

45607 #include <sys/msg.h>
45608 ...
45609 int result;
45610 int msqid;
45611 struct message {
45612     long type;
45613     char text[20];
45614 } msg;
45615 long msgtyp = 0;
45616 ...
45617 result = msgrcv(msqid, (void *) &msg, sizeof(msg.text),
45618                 msgtyp, MSG_NOERROR | IPC_NOWAIT);
  
```

**APPLICATION USAGE**

The POSIX Realtime Extension defines alternative interfaces for interprocess communication (IPC). Application developers who need to use IPC should design their applications so that modules using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the alternative interfaces.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 2.7](#) (on page 501), [Section 2.8](#) (on page 503), [mq\\_close\(\)](#), [mq\\_getattr\(\)](#), [mq\\_notify\(\)](#), [mq\\_open\(\)](#), [mq\\_receive\(\)](#), [mq\\_send\(\)](#), [mq\\_setattr\(\)](#), [mq\\_unlink\(\)](#), [msgctl\(\)](#), [msgget\(\)](#), [msgsnd\(\)](#), [sigaction\(\)](#)

XBD [Section 3.226](#) (on page 69), [<sys/msg.h>](#)

**CHANGE HISTORY**

First released in Issue 2. Derived from Issue 2 of the SVID.

**Issue 5**

The type of the return value is changed from **int** to **ssize\_t**, and a warning is added to the DESCRIPTION about values of *msgsz* larger than {SSIZE\_MAX}.

The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE DIRECTIONS to the APPLICATION USAGE section.

**Issue 6**

The normative text is updated to avoid use of the term “must” for application requirements.

**Issue 7**

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0398 [345] and XSH/TC1-2008/0399 [421] are applied.

**NAME**

msgsnd — XSI message send operation

**SYNOPSIS**

```
XSI    #include <sys/msg.h>
int msgsnd(int msqid, const void *msgp, size_t msgsz, int msgflg);
```

**DESCRIPTION**

The `msgsnd()` function operates on XSI message queues (see XBD [Section 3.226](#), on page 69). It is unspecified whether this function interoperates with the realtime interprocess communication facilities defined in [Section 2.8](#) (on page 503).

The `msgsnd()` function shall send a message to the queue associated with the message queue identifier specified by `msqid`.

The application shall ensure that the argument `msgp` points to a user-defined buffer that contains first a field of type **long** specifying the type of the message, and then a data portion that holds the data bytes of the message. The structure below is an example of what this user-defined buffer might look like:

```
struct mymsg {
    long    mtype;        /* Message type. */
    char    mtext[1];     /* Message text. */
}
```

The structure member `mtype` is a non-zero positive type **long** that can be used by the receiving process for message selection.

The structure member `mtext` is any text of length `msgsz` bytes. The argument `msgsz` can range from 0 to a system-imposed maximum.

The argument `msgflg` specifies the action to be taken if one or more of the following is true:

- The number of bytes already on the queue is equal to **msg\_qbytes**; see `<sys/msg.h>`.
- The total number of messages on all queues system-wide is equal to the system-imposed limit.

These actions are as follows:

- If `(msgflg & IPC_NOWAIT)` is non-zero, the message shall not be sent and the calling thread shall return immediately.
- If `(msgflg & IPC_NOWAIT)` is 0, the calling thread shall suspend execution until one of the following occurs:
  - The condition responsible for the suspension no longer exists, in which case the message is sent.
  - The message queue identifier `msqid` is removed from the system; when this occurs, `errno` shall be set to [EIDRM] and `-1` shall be returned.
  - The calling thread receives a signal that is to be caught; in this case the message is not sent and the calling thread resumes execution in the manner prescribed in [sigaction\(\)](#).

Upon successful completion, the following actions are taken with respect to the data structure associated with *msqid*; see `<sys/msg.h>`:

- **msg\_qnum** shall be incremented by 1.
- **msg\_lspid** shall be set to the process ID of the calling process.
- **msg\_stime** shall be set to the current time, as described in [Section 2.7.1](#) (on page 502).

#### RETURN VALUE

Upon successful completion, *msgsnd()* shall return 0; otherwise, no message shall be sent, *msgsnd()* shall return -1, and *errno* shall be set to indicate the error.

#### ERRORS

The *msgsnd()* function shall fail if:

- |          |                                                                                                                                                                                       |
|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [EACCES] | Operation permission is denied to the calling process; see <a href="#">Section 2.7</a> (on page 501).                                                                                 |
| [EAGAIN] | The message cannot be sent for one of the reasons cited above and ( <i>msgflg</i> & <i>IPC_NOWAIT</i> ) is non-zero.                                                                  |
| [EIDRM]  | The message queue identifier <i>msqid</i> is removed from the system.                                                                                                                 |
| [EINTR]  | The <i>msgsnd()</i> function was interrupted by a signal.                                                                                                                             |
| [EINVAL] | The value of <i>msqid</i> is not a valid message queue identifier, or the value of <i>mtyp</i> is less than 1; or the value of <i>msgsz</i> is greater than the system-imposed limit. |

#### EXAMPLES

##### Sending a Message

The following example sends a message to the queue identified by the *msqid* argument (assuming that value has previously been set). This call specifies that an error should be reported if no message is available. The message size is calculated directly using the *sizeof* operator.

```
#include <sys/msg.h>
...
int result;
int msqid;
struct message {
    long type;
    char text[20];
} msg;

msg.type = 1;
strcpy(msg.text, "This is message 1");
...
result = msgsnd(msqid, (void *) &msg, sizeof(msg.text), IPC_NOWAIT);
```

#### APPLICATION USAGE

The POSIX Realtime Extension defines alternative interfaces for interprocess communication (IPC). Application developers who need to use IPC should design their applications so that modules using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the alternative interfaces.



45725 **RATIONALE**

45726 None.

45727 **FUTURE DIRECTIONS**

45728 None.

45729 **SEE ALSO**

45730 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 503), [mq\\_close\(\)](#), [mq\\_getattr\(\)](#), [mq\\_notify\(\)](#),  
45731 [mq\\_open\(\)](#), [mq\\_receive\(\)](#), [mq\\_send\(\)](#), [mq\\_setattr\(\)](#), [mq\\_unlink\(\)](#), [msgctl\(\)](#), [msgget\(\)](#), [msgrcv\(\)](#),  
45732 [sigaction\(\)](#)

45733 [XBD Section 3.226](#) (on page 69), [<sys/msg.h>](#)45734 **CHANGE HISTORY**

45735 First released in Issue 2. Derived from Issue 2 of the SVID.

45736 **Issue 5**

45737 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE  
45738 DIRECTIONS to a new APPLICATION USAGE section.

45739 **Issue 6**

45740 The normative text is updated to avoid use of the term “must” for application requirements.

45741 **Issue 7**

45742 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0400 [345] and XSH/TC1-2008/0401  
45743 [359] are applied.

**NAME**

msync — synchronize memory with physical storage

**SYNOPSIS**

```
#include <sys/mman.h>

int msync(void *addr, size_t len, int flags);
```

**DESCRIPTION**

The *msync()* function shall write all modified data to permanent storage locations, if any, in those whole pages containing any part of the address space of the process starting at address *addr* and continuing for *len* bytes. If no such storage exists, *msync()* need not have any effect. If requested, the *msync()* function shall then invalidate cached copies of data.

The implementation may require that *addr* be a multiple of the page size as returned by *sysconf()*.

For mappings to files, the *msync()* function shall ensure that all write operations are completed as defined for synchronized I/O data integrity completion. It is unspecified whether the implementation also writes out other file attributes. When the *msync()* function is called on MAP\_PRIVATE mappings, any modified data shall not be written to the underlying object and shall not cause such data to be made visible to other processes. It is unspecified whether data in MAP\_PRIVATE mappings has any permanent storage locations. The effect of *msync()* on a shared memory object or a typed memory object is unspecified. The behavior of this function is unspecified if the mapping was not established by a call to *mmap()*.

The *flags* argument is constructed from the bitwise-inclusive OR of one or more of the following flags defined in the `<sys/mman.h>` header:

| Symbolic Constant | Description                  |
|-------------------|------------------------------|
| MS_ASYNC          | Perform asynchronous writes. |
| MS_SYNC           | Perform synchronous writes.  |
| MS_INVALIDATE     | Invalidate cached data.      |

When MS\_ASYNC is specified, *msync()* shall return immediately once all the write operations are initiated or queued for servicing; when MS\_SYNC is specified, *msync()* shall not return until all write operations are completed as defined for synchronized I/O data integrity completion. Either MS\_ASYNC or MS\_SYNC shall be specified, but not both.

When MS\_INVALIDATE is specified, *msync()* shall invalidate all cached copies of mapped data that are inconsistent with the permanent storage locations such that subsequent references shall obtain data that was consistent with the permanent storage locations sometime between the call to *msync()* and the first subsequent memory reference to the data.

If *msync()* causes any write to a file, the file's last data modification and last file status change timestamps shall be marked for update.

**RETURN VALUE**

Upon successful completion, *msync()* shall return 0; otherwise, it shall return -1 and set *errno* to indicate the error.

**ERRORS**

The *msync()* function shall fail if:

[EBUSY] Some or all of the addresses in the range starting at *addr* and continuing for *len* bytes are locked, and MS\_INVALIDATE is specified.

45787 [EINVAL] The value of *flags* is invalid.

45788 [ENOMEM] The addresses in the range starting at *addr* and continuing for *len* bytes are

45789 outside the range allowed for the address space of a process or specify one or

45790 more pages that are not mapped.

45791 The *msync()* function may fail if:

45792 [EINVAL] The value of *addr* is not a multiple of the page size as returned by *sysconf()*.

#### 45793 EXAMPLES

45794 None.

#### 45795 APPLICATION USAGE

45796 The *msync()* function is only supported if the Synchronized Input and Output option is

45797 supported, and thus need not be available on all implementations.

45798 The *msync()* function should be used by programs that require a memory object to be in a

45799 known state; for example, in building transaction facilities.

45800 Normal system activity can cause pages to be written to disk. Therefore, there are no guarantees

45801 that *msync()* is the only control over when pages are or are not written to disk.

#### 45802 RATIONALE

45803 The *msync()* function writes out data in a mapped region to the permanent storage for the

45804 underlying object. The call to *msync()* ensures data integrity of the file.

45805 After the data is written out, any cached data may be invalidated if the MS\_INVALIDATE flag

45806 was specified. This is useful on systems that do not support read/write consistency.

#### 45807 FUTURE DIRECTIONS

45808 None.

#### 45809 SEE ALSO

45810 *mmap()*, *sysconf()*

45811 XBD <sys/mman.h>

#### 45812 CHANGE HISTORY

45813 First released in Issue 4, Version 2.

#### 45814 Issue 5

45815 Moved from X/OPEN UNIX extension to BASE.

45816 Aligned with *msync()* in the POSIX Realtime Extension as follows:

- 45817 • The DESCRIPTION is extensively reworded.
- 45818 • [EBUSY] and a new form of [EINVAL] are added as mandatory error conditions.

#### 45819 Issue 6

45820 The *msync()* function is marked as part of the Memory Mapped Files and Synchronized Input

45821 and Output options.

45822 The following changes are made for alignment with the ISO POSIX-1: 1996 standard:

- 45823 • The [EBUSY] mandatory error condition is added.

45824 The following new requirements on POSIX implementations derive from alignment with the

45825 Single UNIX Specification:

- 45826                   • The DESCRIPTION is updated to state that implementations require *addr* to be a multiple  
45827                   of the page size.
- 45828                   • The second [EINVAL] error condition is made mandatory.
- 45829                   The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding reference to  
45830                   typed memory objects.
- 45831   **Issue 7**
- 45832                   Austin Group Interpretation 1003.1-2001 #078 is applied, clarifying page alignment  
45833                   requirements.
- 45834                   SD5-XSH-ERN-110 is applied.
- 45835                   The *msync()* function is marked as part of the Synchronized Input and Output option or XSI  
45836                   option as the Memory Mapped Files is moved to the Base.
- 45837                   Changes are made related to support for finegrained timestamps.

45838 **NAME**

45839       munlock — unlock a range of process address space

45840 **SYNOPSIS**

```
45841 MLR       #include <sys/mman.h>  
45842       int munlock(const void *addr, size_t len);
```

45843 **DESCRIPTION**45844       Refer to *mlock()*.

45845 **NAME**

45846       munlockall — unlock the address space of a process

45847 **SYNOPSIS**

```
45848 ML      #include <sys/mman.h>
45849      int munlockall(void);
```

45850 **DESCRIPTION**45851       Refer to *mlockall()*.

45852 **NAME**45853 `munmap` — unmap pages of memory45854 **SYNOPSIS**45855 `#include <sys/mman.h>`45856 `int munmap(void *addr, size_t len);`45857 **DESCRIPTION**

45858 The `munmap()` function shall remove any mappings for those entire pages containing any part of  
 45859 the address space of the process starting at `addr` and continuing for `len` bytes. Further references  
 45860 to these pages shall result in the generation of a SIGSEGV signal to the process. If there are no  
 45861 mappings in the specified address range, then `munmap()` has no effect.

45862 The implementation may require that `addr` be a multiple of the page size as returned by  
 45863 `sysconf()`.

45864 If a mapping to be removed was private, any modifications made in this address range shall be  
 45865 discarded.

45866 ML|MLR Any memory locks (see `mlock()` and `mlockall()`) associated with this address range shall be  
 45867 removed, as if by an appropriate call to `munlock()`.

45868 TYM If a mapping removed from a typed memory object causes the corresponding address range of  
 45869 the memory pool to be inaccessible by any process in the system except through allocatable  
 45870 mappings (that is, mappings of typed memory objects opened with the  
 45871 POSIX\_TYPED\_MEM\_MAP\_ALLOCATABLE flag), then that range of the memory pool shall  
 45872 become deallocated and may become available to satisfy future typed memory allocation  
 45873 requests.

45874 A mapping removed from a typed memory object opened with the  
 45875 POSIX\_TYPED\_MEM\_MAP\_ALLOCATABLE flag shall not affect in any way the availability of  
 45876 that typed memory for allocation.

45877 The behavior of this function is unspecified if the mapping was not established by a call to  
 45878 `mmap()`.

45879 **RETURN VALUE**

45880 Upon successful completion, `munmap()` shall return 0; otherwise, it shall return -1 and set `errno`  
 45881 to indicate the error.

45882 **ERRORS**

45883 The `munmap()` function shall fail if:

45884 [EINVAL] Addresses in the range `[addr,addr+len)` are outside the valid range for the  
 45885 address space of a process.

45886 [EINVAL] The `len` argument is 0.

45887 The `munmap()` function may fail if:

45888 [EINVAL] The `addr` argument is not a multiple of the page size as returned by `sysconf()`.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

The *munmap()* function corresponds to SVR4, just as the *mmap()* function does.

It is possible that an application has applied process memory locking to a region that contains shared memory. If this has occurred, the *munmap()* call ignores those locks and, if necessary, causes those locks to be removed.

Most implementations require that *addr* is a multiple of the page size as returned by *sysconf()*.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*mlock()*, *mlockall()*, *mmap()*, *posix\_typed\_mem\_open()*, *sysconf()*

XBD <sys/mman.h>

**CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

Aligned with *munmap()* in the POSIX Realtime Extension as follows:

- The DESCRIPTION is extensively reworded.
- The SIGBUS error is no longer permitted to be generated.

**Issue 6**

The *munmap()* function is marked as part of the Memory Mapped Files and Shared Memory Objects option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to state that implementations require *addr* to be a multiple of the page size.
- The [EINVAL] error conditions are added.

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- Semantics for typed memory objects are added to the DESCRIPTION.
- The *posix\_typed\_mem\_open()* function is added to the SEE ALSO section.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/36 is applied, changing the margin code in the SYNOPSIS from MF|SHM to MC3 (notation for MF|SHM|TYM).

**Issue 7**

Austin Group Interpretation 1003.1-2001 #078 is applied, clarifying page alignment requirements.

The *munmap()* function is moved from the Memory Mapped Files option to the Base.



45928 **NAME**

45929 nan, nanf, nanl — return quiet NaN

45930 **SYNOPSIS**

```
45931 #include <math.h>
45932 double nan(const char *tagp);
45933 float nanf(const char *tagp);
45934 long double nanl(const char *tagp);
```

45935 **DESCRIPTION**

45936 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 45937 conflict between the requirements described here and the ISO C standard is unintentional. This  
 45938 volume of POSIX.1-2008 defers to the ISO C standard.

45939 The function call *nan("n-char-sequence")* shall be equivalent to:

```
45940 strtod("NAN(n-char-sequence)", (char **) NULL);
```

45941 The function call *nan("")* shall be equivalent to:

```
45942 strtod("NAN()", (char **) NULL)
```

45943 If *tagp* does not point to an *n-char* sequence or an empty string, the function call shall be  
 45944 equivalent to:

```
45945 strtod("NAN", (char **) NULL)
```

45946 Function calls to *nanf()* and *nanl()* are equivalent to the corresponding function calls to *strtof()*  
 45947 and *strtold()*.

45948 **RETURN VALUE**

45949 These functions shall return a quiet NaN, if available, with content indicated through *tagp*.

45950 If the implementation does not support quiet NaNs, these functions shall return zero.

45951 **ERRORS**

45952 No errors are defined.

45953 **EXAMPLES**

45954 None.

45955 **APPLICATION USAGE**

45956 None.

45957 **RATIONALE**

45958 None.

45959 **FUTURE DIRECTIONS**

45960 None.

45961 **SEE ALSO**

45962 *strtod()*

45963 XBD *<math.h>*

45964 **CHANGE HISTORY**

45965 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

**NAME**

nanosleep — high resolution sleep

**SYNOPSIS**

```
#include <time.h>

int nanosleep(const struct timespec *rqtp, struct timespec *rmtp);
```

**DESCRIPTION**

The *nanosleep()* function shall cause the current thread to be suspended from execution until either the time interval specified by the *rqtp* argument has elapsed or a signal is delivered to the calling thread, and its action is to invoke a signal-catching function or to terminate the process. The suspension time may be longer than requested because the argument value is rounded up to an integer multiple of the sleep resolution or because of the scheduling of other activity by the system. But, except for the case of being interrupted by a signal, the suspension time shall not be less than the time specified by *rqtp*, as measured by the system clock `CLOCK_REALTIME`.

The use of the *nanosleep()* function has no effect on the action or blockage of any signal.

**RETURN VALUE**

If the *nanosleep()* function returns because the requested time has elapsed, its return value shall be zero.

If the *nanosleep()* function returns because it has been interrupted by a signal, it shall return a value of `-1` and set *errno* to indicate the interruption. If the *rmtp* argument is non-NULL, the **timespec** structure referenced by it is updated to contain the amount of time remaining in the interval (the requested time minus the time actually slept). The *rqtp* and *rmtp* arguments can point to the same object. If the *rmtp* argument is NULL, the remaining time is not returned.

If *nanosleep()* fails, it shall return a value of `-1` and set *errno* to indicate the error.

**ERRORS**

The *nanosleep()* function shall fail if:

[EINTR] The *nanosleep()* function was interrupted by a signal.

[EINVAL] The *rqtp* argument specified a nanosecond value less than zero or greater than or equal to 1 000 million.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

It is common to suspend execution of a thread for an interval in order to poll the status of a non-interrupting function. A large number of actual needs can be met with a simple extension to *sleep()* that provides finer resolution.

In the POSIX.1-1990 standard and SVR4, it is possible to implement such a routine, but the frequency of wakeup is limited by the resolution of the *alarm()* and *sleep()* functions. In 4.3 BSD, it is possible to write such a routine using no static storage and reserving no system facilities. Although it is possible to write a function with similar functionality to *sleep()* using the remainder of the *timer\_\**() functions, such a function requires the use of signals and the reservation of some signal number. This volume of POSIX.1-2008 requires that *nanosleep()* be non-intrusive of the signals function.

The *nanosleep()* function shall return a value of 0 on success and `-1` on failure or if interrupted.

46010 This latter case is different from *sleep()*. This was done because the remaining time is returned  
46011 via an argument structure pointer, *rmtpt*, instead of as the return value.

#### 46012 FUTURE DIRECTIONS

46013 None.

#### 46014 SEE ALSO

46015 *clock\_nanosleep()*, *sleep()*

46016 XBD <time.h>

#### 46017 CHANGE HISTORY

46018 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

#### 46019 Issue 6

46020 The *nanosleep()* function is marked as part of the Timers option.

46021 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
46022 implementation does not support the Timers option.

46023 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/37 is applied, updating the SEE ALSO  
46024 section to include the *clock\_nanosleep()* function.

46025 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/63 is applied, correcting text in the  
46026 RATIONALE section.

#### 46027 Issue 7

46028 SD5-XBD-ERN-33 is applied.

46029 The *nanosleep()* function is moved from the Timers option to the Base.

46030 **NAME**

46031 nearbyint, nearbyintf, nearbyintl — floating-point rounding functions

46032 **SYNOPSIS**

```
46033 #include <math.h>
46034 double nearbyint(double x);
46035 float nearbyintf(float x);
46036 long double nearbyintl(long double x);
```

46037 **DESCRIPTION**

46038 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
46039 conflict between the requirements described here and the ISO C standard is unintentional. This  
46040 volume of POSIX.1-2008 defers to the ISO C standard.

46041 These functions shall round their argument to an integer value in floating-point format, using  
46042 the current rounding direction and without raising the inexact floating-point exception.

46043 **RETURN VALUE**

46044 MX Upon successful completion, these functions shall return the rounded integer value. The result  
46045 shall have the same sign as  $x$ .

46046 MX If  $x$  is NaN, a NaN shall be returned.

46047 If  $x$  is  $\pm 0$ ,  $\pm 0$  shall be returned.

46048 If  $x$  is  $\pm \text{Inf}$ ,  $x$  shall be returned.

46049 **ERRORS**

46050 No errors are defined.

46051 **EXAMPLES**

46052 None.

46053 **APPLICATION USAGE**

46054 The integral value returned by these functions need not be expressible as an `intmax_t`. The  
46055 return value should be tested before assigning it to an integer type to avoid the undefined  
46056 results of an integer overflow.

46057 **RATIONALE**

46058 None.

46059 **FUTURE DIRECTIONS**

46060 None.

46061 **SEE ALSO**

46062 [`feclearexcept\(\)`](#), [`fetestexcept\(\)`](#)

46063 XBD [Section 4.20](#) (on page 117), [`<math.h>`](#)

46064 **CHANGE HISTORY**

46065 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

46066 **Issue 7**

46067 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0402 [346,428] is applied.

**NAME**

newlocale — create or modify a locale object

**SYNOPSIS**

```
CX      #include <locale.h>

      locale_t newlocale(int category_mask, const char *locale,
      locale_t base);
```

**DESCRIPTION**

The *newlocale()* function shall create a new locale object or modify an existing one. If the *base* argument is **(locale\_t)0**, a new locale object shall be created. It is unspecified whether the locale object pointed to by *base* shall be modified, or freed and a new locale object created.

The *category\_mask* argument specifies the locale categories to be set or modified. Values for *category\_mask* shall be constructed by a bitwise-inclusive OR of the symbolic constants *LC\_CTYPE\_MASK*, *LC\_NUMERIC\_MASK*, *LC\_TIME\_MASK*, *LC\_COLLATE\_MASK*, *LC\_MONETARY\_MASK*, and *LC\_MESSAGES\_MASK*, or any of the implementation-defined mask values defined in **<locale.h>**.

For each category with the corresponding bit set in *category\_mask* the data from the locale named by *locale* shall be used. In the case of modifying an existing locale object, the data from the locale named by *locale* shall replace the existing data within the locale object. If a completely new locale object is created, the data for all sections not requested by *category\_mask* shall be taken from the default locale.

The following preset values of *locale* are defined for all settings of *category\_mask*:

|         |                                                                                                                                                                                                                                                           |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| "POSIX" | Specifies the minimal environment for C-language translation called the POSIX locale.                                                                                                                                                                     |
| "C"     | Equivalent to "POSIX".                                                                                                                                                                                                                                    |
| " "     | Specifies an implementation-defined native environment. This corresponds to the value of the associated environment variables, <i>LC_*</i> and <i>LANG</i> ; see XBD <a href="#">Chapter 7</a> (on page 135) and <a href="#">Chapter 8</a> (on page 173). |

If the *base* argument is not **(locale\_t)0** and the *newlocale()* function call succeeds, the contents of *base* are unspecified. Applications shall ensure that they stop using *base* as a locale object before calling *newlocale()*. If the function call fails and the *base* argument is not **(locale\_t)0**, the contents of *base* shall remain valid and unchanged.

The behavior is undefined if the *base* argument is the special locale object *LC\_GLOBAL\_LOCALE*, or is not a valid locale object handle and is not **(locale\_t)0**.

**RETURN VALUE**

Upon successful completion, the *newlocale()* function shall return a handle which the caller may use on subsequent calls to *duplocale()*, *freelocale()*, and other functions taking a **locale\_t** argument.

Upon failure, the *newlocale()* function shall return **(locale\_t)0** and set *errno* to indicate the error.

**ERRORS**

The *newlocale()* function shall fail if:

|          |                                                                                           |
|----------|-------------------------------------------------------------------------------------------|
| [ENOMEM] | There is not enough memory available to create the locale object or load the locale data. |
|----------|-------------------------------------------------------------------------------------------|

46110 [EINVAL] The *category\_mask* contains a bit that does not correspond to a valid category.  
46111 [ENOENT] For any of the categories in *category\_mask*, the locale data is not available.  
46112 The *newlocale()* function may fail if:  
46113 [EINVAL] The *locale* argument is not a valid string pointer.

## 46114 EXAMPLES

### 46115 Constructing a Locale Object from Different Locales

46116 The following example shows the construction of a locale where the *LC\_CTYPE* category data  
46117 comes from a locale *loc1* and the *LC\_TIME* category data from a locale *loc2*:

```
46118 #include <locale.h>
46119 ...
46120 locale_t loc, new_loc;
46121 /* Get the "loc1" data. */
46122 loc = newlocale (LC_CTYPE_MASK, "loc1", (locale_t)0);
46123 if (loc == (locale_t) 0)
46124     abort ();
46125 /* Get the "loc2" data. */
46126 new_loc = newlocale (LC_TIME_MASK, "loc2", loc);
46127 if (new_loc != (locale_t) 0)
46128     /* We don't abort if this fails. In this case this
46129        simply used to unchanged locale object. */
46130     loc = new_loc;
46131 ...
```

### 46132 Freeing up a Locale Object

46133 The following example shows a code fragment to free a locale object created by *newlocale()*:

```
46134 #include <locale.h>
46135 ...
46136 /* Every locale object allocated with newlocale() should be
46137    * freed using freelocale():
46138    */
46139 locale_t loc;
46140 /* Get the locale. */
46141 loc = newlocale (LC_CTYPE_MASK | LC_TIME_MASK, "locname", (locale_t)0);
46142 /* ... Use the locale object ... */
46143 ...
46144 /* Free the locale object resources. */
46145 freelocale (loc);
```

**APPLICATION USAGE**

Handles for locale objects created by the *newlocale()* function should either be released by a corresponding call to *freelocale()*, or be used as a base locale to another *newlocale()* call.

The special locale object LC\_GLOBAL\_LOCALE must not be passed for the *base* argument, even when returned by the *uselocale()* function.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*duplocale()*, *freelocale()*, *uselocale()*

XBD [Chapter 7](#) (on page 135), [Chapter 8](#) (on page 173), [<locale.h>](#)

**CHANGE HISTORY**

First released in Issue 7.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0403 [227], XSH/TC1-2008/0404 [283], XSH/TC1-2008/0405 [295], and XSH/TC1-2008/0406 [227] are applied.

**NAME**

nextafter, nextafterf, nextafterl, nexttoward, nexttowardf, nexttowardl — next representable floating-point number

**SYNOPSIS**

```
#include <math.h>

double nextafter(double x, double y);
float nextafterf(float x, float y);
long double nextafterl(long double x, long double y);
double nexttoward(double x, long double y);
float nexttowardf(float x, long double y);
long double nexttowardl(long double x, long double y);
```

**DESCRIPTION**

**CX** The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *nextafter()*, *nextafterf()*, and *nextafterl()* functions shall compute the next representable floating-point value following *x* in the direction of *y*. Thus, if *y* is less than *x*, *nextafter()* shall return the largest representable floating-point number less than *x*. The *nextafter()*, *nextafterf()*, and *nextafterl()* functions shall return *y* if *x* equals *y*.

The *nexttoward()*, *nexttowardf()*, and *nexttowardl()* functions shall be equivalent to the corresponding *nextafter()* functions, except that the second parameter shall have type **long double** and the functions shall return *y* converted to the type of the function if *x* equals *y*.

An application wishing to check for error situations should set *errno* to zero and call *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-zero, an error has occurred.

**RETURN VALUE**

Upon successful completion, these functions shall return the next representable floating-point value following *x* in the direction of *y*.

If *x*=*y*, *y* (of the type *x*) shall be returned.

If *x* is finite and the correct function value would overflow, a range error shall occur and  $\pm\text{HUGE\_VAL}$ ,  $\pm\text{HUGE\_VALF}$ , and  $\pm\text{HUGE\_VALL}$  (with the same sign as *x*) shall be returned as appropriate for the return type of the function.

**MX** If *x* or *y* is NaN, a NaN shall be returned.

**MX** If *x*!=*y* and the correct function value is subnormal, zero, or underflows, a range error shall occur, and

**MX** the correct function value (if representable) or

**MX** 0.0 shall be returned.

**ERRORS**

These functions shall fail if:

**Range Error** The correct value overflows.

If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero, then *errno* shall be set to [ERANGE]. If the integer expression (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the overflow floating-point exception shall be raised.



46207 MX **Range Error** The correct value is subnormal or underflows.  
 46208 If the integer expression *(math\_errhandling & MATH\_ERRNO)* is non-zero,  
 46209 then *errno* shall be set to [ERANGE]. If the integer expression  
 46210 *(math\_errhandling & MATH\_ERREXCEPT)* is non-zero, then the underflow  
 46211 floating-point exception shall be raised.

**EXAMPLES**

46212  
 46213 None.

**APPLICATION USAGE**

46215 On error, the expressions *(math\_errhandling & MATH\_ERRNO)* and *(math\_errhandling &*  
 46216 *MATH\_ERREXCEPT)* are independent of each other, but at least one of them must be non-zero.

46217 When **<tgmath.h>** is included, note that the return type of *nextafter()* depends on the generic  
 46218 typing deduced from both arguments, while the return type of *nexttoward()* depends only on the  
 46219 generic typing of the first argument.

**RATIONALE**

46220  
 46221 None.

**FUTURE DIRECTIONS**

46222  
 46223 None.

**SEE ALSO**

46225 *feclearexcept()*, *fetestexcept()*

46226 XBD Section 4.20 (on page 117), **<math.h>**, **<tgmath.h>**

**CHANGE HISTORY**

46228 First released in Issue 4, Version 2.

**Issue 5**

46229  
 46230 Moved from X/OPEN UNIX extension to BASE.

**Issue 6**

46231 The *nextafter()* function is no longer marked as an extension.

46233 The *nextafterf()*, *nextafterl()*, *nexttoward()*, *nexttowardf()*, and *nexttowardl()* functions are added  
 46234 for alignment with the ISO/IEC 9899:1999 standard.

46235 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
 46236 revised to align with the ISO/IEC 9899:1999 standard.

46237 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
 46238 marked.

**Issue 7**

46240 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0407 [68] and XSH/TC1-2008/0408  
 46241 [357] are applied.

## 46242 NAME

46243 nftw — walk a file tree

## 46244 SYNOPSIS

```
46245 XSI      #include <ftw.h>
46246
46246      int nftw(const char *path, int (*fn)(const char *,
46247          const struct stat *, int, struct FTW *), int fd_limit, int flags);
```

## 46248 DESCRIPTION

46249 The *nftw()* function shall recursively descend the directory hierarchy rooted in *path*. The *nftw()*  
 46250 function has a similar effect to *ftw()* except that it takes an additional argument *flags*, which is a  
 46251 bitwise-inclusive OR of zero or more of the following flags:

46252 FTW\_CHDIR If set, *nftw()* shall change the current working directory to each directory as it  
 46253 reports files in that directory. If clear, *nftw()* shall not change the current  
 46254 working directory.

46255 FTW\_DEPTH If set, *nftw()* shall report all files in a directory before reporting the directory  
 46256 itself. If clear, *nftw()* shall report any directory before reporting the files in that  
 46257 directory.

46258 FTW\_MOUNT If set, *nftw()* shall only report files in the same file system as *path*. If clear,  
 46259 *nftw()* shall report all files encountered during the walk.

46260 FTW\_PHYS If set, *nftw()* shall perform a physical walk and shall not follow symbolic links.

46261 If FTW\_PHYS is clear and FTW\_DEPTH is set, *nftw()* shall follow links instead of reporting  
 46262 them, but shall not report any directory that would be a descendant of itself. If FTW\_PHYS is  
 46263 clear and FTW\_DEPTH is clear, *nftw()* shall follow links instead of reporting them, but shall not  
 46264 report the contents of any directory that would be a descendant of itself.

46265 At each file it encounters, *nftw()* shall call the user-supplied function *fn* with four arguments:

- 46266 • The first argument is the pathname of the object.
- 46267 • The second argument is a pointer to the **stat** buffer containing information on the object,  
 46268 filled in as if *fstatat()*, *stat()*, or *lstat()* had been called to retrieve the information.
- 46269 • The third argument is an integer giving additional information. Its value is one of the  
 46270 following:

46271 FTW\_D The object is a directory.

46272 FTW\_DNR The object is a directory that cannot be read. The *fn* function shall not be  
 46273 called for any of its descendants.

46274 FTW\_DP The object is a directory and subdirectories have been visited. (This condition  
 46275 shall only occur if the FTW\_DEPTH flag is included in *flags*.)

46276 FTW\_F The object is a non-directory file.

46277 FTW\_NS The *stat()* function failed on the object because of lack of appropriate  
 46278 permission. The **stat** buffer passed to *fn* is undefined. Failure of *stat()* for any  
 46279 other reason is considered an error and *nftw()* shall return  $-1$ .

46280 FTW\_SL The object is a symbolic link. (This condition shall only occur if the  
 46281 FTW\_PHYS flag is included in *flags*.)

46282 FTW\_SLN The object is a symbolic link that does not name an existing file. (This  
46283 condition shall only occur if the FTW\_PHYS flag is not included in *flags*.)

- 46284 • The fourth argument is a pointer to an **FTW** structure. The value of **base** is the offset of the  
46285 object's filename in the pathname passed as the first argument to *fn*. The value of **level**  
46286 indicates depth relative to the root of the walk, where the root level is 0.

46287 The results are unspecified if the application-supplied *fn* function does not preserve the current  
46288 working directory.

46289 The argument *fd\_limit* sets the maximum number of file descriptors that shall be used by *nftw()*  
46290 while traversing the file tree. At most one file descriptor shall be used for each directory level.

46291 The *nftw()* function need not be thread-safe.

## 46292 RETURN VALUE

46293 The *nftw()* function shall continue until the first of the following conditions occurs:

- 46294 • An invocation of *fn* shall return a non-zero value, in which case *nftw()* shall return that  
46295 value.
- 46296 • The *nftw()* function detects an error other than [EACCES] (see FTW\_DNR and FTW\_NS  
46297 above), in which case *nftw()* shall return -1 and set *errno* to indicate the error.
- 46298 • The tree is exhausted, in which case *nftw()* shall return 0.

## 46299 ERRORS

46300 The *nftw()* function shall fail if:

- 46301 [EACCES] Search permission is denied for any component of *path* or read permission is  
46302 denied for *path*, or *fn* returns -1 and does not reset *errno*.
- 46303 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*  
46304 argument.
- 46305 [ENAMETOOLONG]  
46306 The length of a component of a pathname is longer than {NAME\_MAX}.
- 46307 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.
- 46308 [ENOTDIR] A component of *path* names an existing file that is neither a directory nor a  
46309 symbolic link to a directory.
- 46310 [EOVERFLOW] A field in the **stat** structure cannot be represented correctly in the current  
46311 programming environment for one or more files found in the file hierarchy.

46312 The *nftw()* function may fail if:

- 46313 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
46314 resolution of the *path* argument.
- 46315 [EMFILE] All file descriptors available to the process are currently open.
- 46316 [ENAMETOOLONG]  
46317 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
46318 symbolic link produced an intermediate result with a length that exceeds  
46319 {PATH\_MAX}.
- 46320 [ENFILE] Too many files are currently open in the system.

46321 In addition, *errno* may be set if the function pointed to by *fn* causes *errno* to be set.

**EXAMPLES**

The following program traverses the directory tree under the path named in its first command-line argument, or under the current directory if no argument is supplied. It displays various information about each file. The second command-line argument can be used to specify characters that control the value assigned to the *flags* argument when calling *nftw()*.

```
#include <ftw.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>

static int
display_info(const char *fpath, const struct stat *sb,
             int tflag, struct FTW *ftwbuf)
{
    printf("%-3s %2d %7jd    %-40s %d %s\n",
           (tflag == FTW_D) ? "d" : (tflag == FTW_DNR) ? "dnr" :
           (tflag == FTW_DP) ? "dp" : (tflag == FTW_F) ?
           (S_ISBLK(sb->st_mode) ? "f b" :
            S_ISCHR(sb->st_mode) ? "f c" :
            S_ISFIFO(sb->st_mode) ? "f p" :
            S_ISREG(sb->st_mode) ? "f r" :
            S_ISSOCK(sb->st_mode) ? "f s" : "f ?") :
           (tflag == FTW_NS) ? "ns" : (tflag == FTW_SL) ? "sl" :
           (tflag == FTW_SLN) ? "sln" : "?",
           ftwbuf->level, (intmax_t) sb->st_size,
           fpath, ftwbuf->base, fpath + ftwbuf->base);
    return 0;          /* To tell nftw() to continue */
}

int
main(int argc, char *argv[])
{
    int flags = 0;

    if (argc > 2 && strchr(argv[2], 'd') != NULL)
        flags |= FTW_DEPTH;
    if (argc > 2 && strchr(argv[2], 'p') != NULL)
        flags |= FTW_PHYS;

    if (nftw((argc < 2) ? "." : argv[1], display_info, 20, flags) == -1)
    {
        perror("nftw");
        exit(EXIT_FAILURE);
    }

    exit(EXIT_SUCCESS);
}
```

**APPLICATION USAGE**

The *nftw()* function may allocate dynamic storage during its operation. If *nftw()* is forcibly terminated, such as by *longjmp()* or *siglongjmp()* being executed by the function pointed to by *fn* or an interrupt routine, *nftw()* does not have a chance to free that storage, so it remains permanently allocated. A safe way to handle interrupts is to store the fact that an interrupt has

46370 occurred, and arrange to have the function pointed to by *fn* return a non-zero value at its next  
 46371 invocation.

#### 46372 RATIONALE

46373 None.

#### 46374 FUTURE DIRECTIONS

46375 None.

#### 46376 SEE ALSO

46377 *fdopendir()*, *fstatat()*, *readdir()*

46378 XBD <ftw.h>

#### 46379 CHANGE HISTORY

46380 First released in Issue 4, Version 2.

#### 46381 Issue 5

46382 Moved from X/OPEN UNIX extension to BASE.

46383 In the DESCRIPTION, the definition of the *depth* argument is clarified.

#### 46384 Issue 6

46385 The Open Group Base Resolution bwg97-003 is applied.

46386 The ERRORS section is updated as follows:

- 46387 • The wording of the mandatory [ELOOP] error condition is updated.
- 46388 • A second optional [ELOOP] error condition is added.
- 46389 • The [Eoverflow] mandatory error condition is added.

46390 Text is added to the DESCRIPTION to say that the *nftw()* function need not be reentrant and that  
 46391 the results are unspecified if the application-supplied *fn* function does not preserve the current  
 46392 working directory.

46393 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/64 is applied, changing the argument  
 46394 *depth* to *fd\_limit* throughout and changing “to a maximum of 5 levels deep” to “using a  
 46395 maximum of 5 file descriptors” in the EXAMPLES section.

#### 46396 Issue 7

46397 Austin Group Interpretations 1003.1-2001 #143 and #156 are applied.

46398 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

46399 SD5-XBD-ERN-61 is applied.

46400 APPLICATION USAGE is added and the EXAMPLES section is replaced with a new example.

46401 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0409 [403], XSH/TC1-2008/0410 [324],  
 46402 and XSH/TC1-2008/0411 [403] are applied.

46403 **NAME**

46404 **nice** — change the nice value of a process

46405 **SYNOPSIS**

46406 XSI `#include <unistd.h>`  
46407 `int nice(int incr);`

46408 **DESCRIPTION**

46409 The *nice()* function shall add the value of *incr* to the nice value of the calling process. A nice  
46410 value of a process is a non-negative number for which a more positive value shall result in less  
46411 favorable scheduling.

46412 A maximum nice value of 2\*[NZERO]–1 and a minimum nice value of 0 shall be imposed by the  
46413 system. Requests for values above or below these limits shall result in the nice value being set to  
46414 the corresponding limit. Only a process with appropriate privileges can lower the nice value.

46415 PS|TPS Calling the *nice()* function has no effect on the priority of processes or threads with policy  
46416 SCHED\_FIFO or SCHED\_RR. The effect on processes or threads with other scheduling policies  
46417 is implementation-defined.

46418 The nice value set with *nice()* shall be applied to the process. If the process is multi-threaded, the  
46419 nice value shall affect all system scope threads in the process.

46420 As –1 is a permissible return value in a successful situation, an application wishing to check for  
46421 error situations should set *errno* to 0, then call *nice()*, and if it returns –1, check to see whether  
46422 *errno* is non-zero.

46423 **RETURN VALUE**

46424 Upon successful completion, *nice()* shall return the new nice value –[NZERO]. Otherwise, –1  
46425 shall be returned, the nice value of the process shall not be changed, and *errno* shall be set to  
46426 indicate the error.

46427 **ERRORS**

46428 The *nice()* function shall fail if:

46429 [EPERM] The *incr* argument is negative and the calling process does not have  
46430 appropriate privileges.

46431 **EXAMPLES**

46432 **Changing the Nice Value**

46433 The following example adds the value of the *incr* argument, –20, to the nice value of the calling  
46434 process.

46435 `#include <unistd.h>`  
46436 `...`  
46437 `int incr = -20;`  
46438 `int ret;`  
46439 `ret = nice(incr);`

46440 **APPLICATION USAGE**

46441 None.

46442 **RATIONALE**

46443 None.

46444 **FUTURE DIRECTIONS**

46445 None.

46446 **SEE ALSO**46447 *exec*, *getpriority()*46448 XBD *<limits.h>*, *<unistd.h>*46449 **CHANGE HISTORY**

46450 First released in Issue 1. Derived from Issue 1 of the SVID.

46451 **Issue 5**46452 A statement is added to the description indicating the effects of this function on the different  
46453 scheduling policies and multi-threaded processes.

## NAME

nl\_langinfo, nl\_langinfo\_l — language information

## SYNOPSIS

```
#include <langinfo.h>

char *nl_langinfo(nl_item item);
char *nl_langinfo_l(nl_item item, locale_t locale);
```

## DESCRIPTION

The *nl\_langinfo()* and *nl\_langinfo\_l()* functions shall return a pointer to a string containing information relevant to the particular language or cultural area defined in the current locale, or in the locale represented by *locale*, respectively (see <langinfo.h>). The manifest constant names and values of *item* are defined in <langinfo.h>. For example:

```
nl_langinfo(ABDAY_1)
```

would return a pointer to the string "Dom" if the identified language was Portuguese, and "Sun" if the identified language was English.

```
nl_langinfo_l(ABDAY_1, loc)
```

would return a pointer to the string "Dom" if the identified language of the locale represented by *loc* was Portuguese, and "Sun" if the identified language of the locale represented by *loc* was English.

The *nl\_langinfo()* function need not be thread-safe.

The behavior is undefined if the *locale* argument to *nl\_langinfo\_l()* is the special locale object LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

## RETURN VALUE

In a locale where *langinfo* data is not defined, these functions shall return a pointer to the corresponding string in the POSIX locale. In all locales, these functions shall return a pointer to an empty string if *item* contains an invalid setting.

The application shall not modify the string returned. The pointer returned by *nl\_langinfo()* might be invalidated or the string content might be overwritten by a subsequent call to *nl\_langinfo()* in any thread or to *nl\_langinfo\_l()* in the same thread or the initial thread, by subsequent calls to *setlocale()* with a category corresponding to the category of *item* (see <langinfo.h>) or the category LC\_ALL, or by subsequent calls to *uselocale()* which change the category corresponding to the category of *item*. The pointer returned by *nl\_langinfo\_l()* might be invalidated or the string content might be overwritten by a subsequent call to *nl\_langinfo\_l()* in the same thread or to *nl\_langinfo()* in any thread, or by subsequent calls to *freelocale()* or *newlocale()* which free or modify the locale object that was passed to *nl\_langinfo\_l()*. The returned pointer and the string content might also be invalidated if the calling thread is terminated.

## ERRORS

No errors are defined.



46492 **EXAMPLES**46493 **Getting Date and Time Formatting Information**

46494 The following example returns a pointer to a string containing date and time formatting  
 46495 information, as defined in the *LC\_TIME* category of the current locale.

```
46496 #include <time.h>
46497 #include <langinfo.h>
46498 ...
46499 strftime(datestring, sizeof(datestring), nl_langinfo(D_T_FMT), tm);
46500 ...
```

46501 **APPLICATION USAGE**

46502 The array pointed to by the return value should not be modified by the program, but may be  
 46503 modified by further calls to these functions.

46504 **RATIONALE**

46505 The possible interactions between internal data used by *nl\_langinfo()* and *nl\_langinfo\_l()* are  
 46506 complicated by the fact that *nl\_langinfo\_l()* must be thread-safe but *nl\_langinfo()* need not be.  
 46507 The various implementation choices are:

- 46508 1. *nl\_langinfo\_l()* and *nl\_langinfo()* use separate buffers, or at least one of them does not use  
 46509 an internal string buffer. In this case there are no interactions.
- 46510 2. *nl\_langinfo\_l()* and *nl\_langinfo()* share an internal per-thread buffer. There can be  
 46511 interactions, but only in the same thread.
- 46512 3. *nl\_langinfo\_l()* uses an internal per-thread buffer, and *nl\_langinfo()* uses (in all threads)  
 46513 the same buffer that *nl\_langinfo\_l()* uses in the initial thread. There can be interactions,  
 46514 but only when *nl\_langinfo\_l()* is called in the initial thread.

46515 **FUTURE DIRECTIONS**

46516 None.

46517 **SEE ALSO**

46518 *setlocale()*, *uselocale()*

46519 XBD Chapter 7 (on page 135), *<langinfo.h>*, *<locale.h>*, *<nl\_types.h>*

46520 **CHANGE HISTORY**

46521 First released in Issue 2.

46522 **Issue 5**

46523 The last paragraph of the DESCRIPTION is moved from the APPLICATION USAGE section.

46524 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

46525 **Issue 7**

46526 Austin Group Interpretation 1003.1-2001 #156 is applied.

46527 The *nl\_langinfo()* function is moved from the XSI option to the Base.

46528 The *nl\_langinfo\_l()* function is added from The Open Group Technical Standard, 2006, Extended  
 46529 API Set Part 4.

46530 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0412 [302], XSH/TC1-2008/0413 [75],  
 46531 XSH/TC1-2008/0414 [283], XSH/TC1-2008/0415 [75,402], XSH/TC1-2008/0416 [283], and  
 46532 XSH/TC1-2008/0417 [402] are applied.

46533 **NAME**

46534       nrand48 — generate uniformly distributed pseudo-random non-negative long integers

46535 **SYNOPSIS**

46536 XSI       #include &lt;stdlib.h&gt;

46537       long nrand48(unsigned short xsubi[3]);

46538 **DESCRIPTION**46539       Refer to *drand48()*.

46540 **NAME**

46541       ntohl, ntohs — convert values between host and network byte order

46542 **SYNOPSIS**

46543       #include &lt;arpa/inet.h&gt;

46544       uint32\_t ntohl(uint32\_t *netlong*);46545       uint16\_t ntohs(uint16\_t *netshort*);46546 **DESCRIPTION**46547       Refer to *htonl()*.

## 46548 NAME

46549 open, openat — open file |

## 46550 SYNOPSIS

46551 OH `#include <sys/stat.h>`46552 `#include <fcntl.h>`46553 `int open(const char *path, int oflag, ...);`46554 `int openat(int fd, const char *path, int oflag, ...);`

## 46555 DESCRIPTION

46556 The `open()` function shall establish the connection between a file and a file descriptor. It shall  
 46557 create an open file description that refers to a file and a file descriptor that refers to that open file  
 46558 description. The file descriptor is used by other I/O functions to refer to that file. The *path*  
 46559 argument points to a pathname naming the file.

46560 The `open()` function shall return a file descriptor for the named file, allocated as described in |  
 46561 [Section 2.14](#) (on page 549). The open file description is new, and therefore the file descriptor |  
 46562 shall not share it with any other process in the system. The FD\_CLOEXEC file descriptor flag  
 46563 associated with the new file descriptor shall be cleared unless the O\_CLOEXEC flag is set in  
 46564 *oflag*.

46565 The file offset used to mark the current position within the file shall be set to the beginning of  
 46566 the file.

46567 The file status flags and file access modes of the open file description shall be set according to  
 46568 the value of *oflag*.

46569 Values for *oflag* are constructed by a bitwise-inclusive OR of flags from the following list, defined  
 46570 in `<fcntl.h>`. Applications shall specify exactly one of the first five values (file access modes)  
 46571 below in the value of *oflag*:

46572 O\_EXEC Open for execute only (non-directory files). The result is unspecified if  
 46573 this flag is applied to a directory.

46574 O\_RDONLY Open for reading only.

46575 O\_RDWR Open for reading and writing. The result is undefined if this flag is  
 46576 applied to a FIFO.

46577 O\_SEARCH Open directory for search only. The result is unspecified if this flag is  
 46578 applied to a non-directory file.

46579 O\_WRONLY Open for writing only.

46580 Any combination of the following may be used:

46581 O\_APPEND If set, the file offset shall be set to the end of the file prior to each write.

46582 O\_CLOEXEC If set, the FD\_CLOEXEC flag for the new file descriptor shall be set.

46583 O\_CREAT If the file exists, this flag has no effect except as noted under O\_EXCL |  
 46584 below. Otherwise, if O\_DIRECTORY is not set the file shall be created as a |  
 46585 regular file; the user ID of the file shall be set to the effective user ID of the |  
 46586 process; the group ID of the file shall be set to the group ID of the file's |  
 46587 parent directory or to the effective group ID of the process; and the access |  
 46588 permission bits (see `<sys/stat.h>`) of the file mode shall be set to the value |  
 46589 of the argument following the *oflag* argument taken as type `mode_t`  
 46590 modified as follows: a bitwise AND is performed on the file-mode bits  
 46591 and the corresponding bits in the complement of the process' file mode

|       |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------|-----|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 46592 |     |             | creation mask. Thus, all bits in the file mode whose corresponding bit in the file mode creation mask is set are cleared. When bits other than the file permission bits are set, the effect is unspecified. The argument following the <i>oflag</i> argument does not affect whether the file is open for reading, writing, or for both. Implementations shall provide a way to initialize the file's group ID to the group ID of the parent directory. Implementations may, but need not, provide an implementation-defined way to initialize the file's group ID to the effective group ID of the calling process. |
| 46593 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46594 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46595 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46596 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46597 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46598 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46599 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46600 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46601 |     | O_DIRECTORY | If <i>path</i> resolves to a non-directory file, fail and set <i>errno</i> to [ENOTDIR].                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 46602 | SIO | O_DSYNC     | Write I/O operations on the file descriptor shall complete as defined by synchronized I/O data integrity completion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 46603 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46604 |     | O_EXCL      | If O_CREAT and O_EXCL are set, <i>open()</i> shall fail if the file exists. The check for the existence of the file and the creation of the file if it does not exist shall be atomic with respect to other threads executing <i>open()</i> naming the same filename in the same directory with O_EXCL and O_CREAT set. If O_EXCL and O_CREAT are set, and <i>path</i> names a symbolic link, <i>open()</i> shall fail and set <i>errno</i> to [EEXIST], regardless of the contents of the symbolic link. If O_EXCL is set and O_CREAT is not set, the result is undefined.                                          |
| 46605 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46606 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46607 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46608 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46609 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46610 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46611 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46612 |     | O_NOCTTY    | If set and <i>path</i> identifies a terminal device, <i>open()</i> shall not cause the terminal device to become the controlling terminal for the process. If <i>path</i> does not identify a terminal device, O_NOCTTY shall be ignored.                                                                                                                                                                                                                                                                                                                                                                            |
| 46613 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46614 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46615 |     | O_NOFOLLOW  | If <i>path</i> names a symbolic link, fail and set <i>errno</i> to [ELOOP].                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 46616 |     | O_NONBLOCK  | When opening a FIFO with O_RDONLY or O_WRONLY set:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 46617 |     |             | <ul style="list-style-type: none"> <li>• If O_NONBLOCK is set, an <i>open()</i> for reading-only shall return without delay. An <i>open()</i> for writing-only shall return an error if no process currently has the file open for reading.</li> </ul>                                                                                                                                                                                                                                                                                                                                                               |
| 46618 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46619 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46620 |     |             | <ul style="list-style-type: none"> <li>• If O_NONBLOCK is clear, an <i>open()</i> for reading-only shall block the calling thread until a thread opens the file for writing. An <i>open()</i> for writing-only shall block the calling thread until a thread opens the file for reading.</li> </ul>                                                                                                                                                                                                                                                                                                                  |
| 46621 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46622 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46623 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46624 |     |             | When opening a block special or character special file that supports non-blocking opens:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 46625 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46626 |     |             | <ul style="list-style-type: none"> <li>• If O_NONBLOCK is set, the <i>open()</i> function shall return without blocking for the device to be ready or available. Subsequent behavior of the device is device-specific.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                    |
| 46627 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46628 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46629 |     |             | <ul style="list-style-type: none"> <li>• If O_NONBLOCK is clear, the <i>open()</i> function shall block the calling thread until the device is ready or available before returning.</li> </ul>                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 46630 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46631 |     |             | Otherwise, the O_NONBLOCK flag shall not cause an error, but it is unspecified whether the file status flags will include the O_NONBLOCK flag.                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 46632 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46633 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46634 | SIO | O_RSYNC     | Read I/O operations on the file descriptor shall complete at the same level of integrity as specified by the O_DSYNC and O_SYNC flags. If both O_DSYNC and O_RSYNC are set in <i>oflag</i> , all I/O operations on the file                                                                                                                                                                                                                                                                                                                                                                                          |
| 46635 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 46636 |     |             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

|       |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|-------|-----------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 46637 |           |            | descriptor shall complete as defined by synchronized I/O data integrity completion. If both O_SYNC and O_RSYNC are set in flags, all I/O operations on the file descriptor shall complete as defined by synchronized I/O file integrity completion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 46638 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46639 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46640 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46641 | XSI   SIO | O_SYNC     | Write I/O operations on the file descriptor shall complete as defined by synchronized I/O file integrity completion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 46642 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46643 | XSI       |            | The O_SYNC flag shall be supported for regular files, even if the Synchronized Input and Output option is not supported.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 46644 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46645 |           | O_TRUNC    | If the file exists and is a regular file, and the file is successfully opened O_RDWR or O_WRONLY, its length shall be truncated to 0, and the mode and owner shall be unchanged. It shall have no effect on FIFO special files or terminal device files. Its effect on other file types is implementation-defined. The result of using O_TRUNC without either O_RDWR or O_WRONLY is undefined.                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 46646 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46647 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46648 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46649 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46650 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46651 |           | O_TTY_INIT | If <i>path</i> identifies a terminal device other than a pseudo-terminal, the device is not already open in any process, and either O_TTY_INIT is set in <i>oflag</i> or O_TTY_INIT has the value zero, <i>open()</i> shall set any non-standard <b>termios</b> structure terminal parameters to a state that provides conforming behavior; see XBD Section 11.2 (on page 205). It is unspecified whether O_TTY_INIT has any effect if the device is already open in any process. If <i>path</i> identifies the slave side of a pseudo-terminal that is not already open in any process, <i>open()</i> shall set any non-standard <b>termios</b> structure terminal parameters to a state that provides conforming behavior, regardless of whether O_TTY_INIT is set. If <i>path</i> does not identify a terminal device, O_TTY_INIT shall be ignored. |
| 46652 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46653 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46654 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46655 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46656 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46657 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46658 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46659 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46660 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46661 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46662 |           |            | If O_CREAT and O_DIRECTORY are set and the requested access mode is neither O_WRONLY +                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 46663 |           |            | nor O_RDWR, the result is unspecified. +                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 46664 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46665 |           |            | If O_CREAT is set and the file did not previously exist, upon successful completion, <i>open()</i> shall mark for update the last data access, last data modification, and last file status change timestamps of the file and the last data modification and last file status change timestamps of the parent directory.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 46666 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46667 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46668 |           |            | If O_TRUNC is set and the file did previously exist, upon successful completion, <i>open()</i> shall mark for update the last data modification and last file status change timestamps of the file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 46669 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46670 | SIO       |            | If both the O_SYNC and O_DSYNC flags are set, the effect is as if only the O_SYNC flag was set.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46671 | OB XSR    |            | If <i>path</i> refers to a STREAMS file, <i>oflag</i> may be constructed from O_NONBLOCK OR'ed with either O_RDONLY, O_WRONLY, or O_RDWR. Other flag values are not applicable to STREAMS devices and shall have no effect on them. The value O_NONBLOCK affects the operation of STREAMS drivers and certain functions applied to file descriptors associated with STREAMS files. For STREAMS drivers, the implementation of O_NONBLOCK is device-specific.                                                                                                                                                                                                                                                                                                                                                                                           |
| 46672 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46673 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46674 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46675 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46676 |           |            | The application shall ensure that it specifies the O_TTY_INIT flag on the first open of a terminal device since system boot or since the device was closed by the process that last had it open. The application need not specify the O_TTY_INIT flag when opening pseudo-terminals. If <i>path</i> names the master side of a pseudo-terminal device, then it is unspecified whether <i>open()</i> locks the slave side so that it cannot be opened. Conforming applications shall call <i>unlockpt()</i> before opening the slave side.                                                                                                                                                                                                                                                                                                              |
| 46677 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46678 | XSI       |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46679 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46680 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 46681 |           |            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

46682 The largest value that can be represented correctly in an object of type **off\_t** shall be established  
 46683 as the offset maximum in the open file description.

46684 The *openat()* function shall be equivalent to the *open()* function except in the case where *path*  
 46685 specifies a relative path. In this case the file to be opened is determined relative to the directory  
 46686 associated with the file descriptor *fd* instead of the current working directory. If the access mode  
 46687 of the open file description associated with the file descriptor is not O\_SEARCH, the function  
 46688 shall check whether directory searches are permitted using the current permissions of the  
 46689 directory underlying the file descriptor. If the access mode is O\_SEARCH, the function shall not  
 46690 perform the check.

46691 The *oflag* parameter and the optional fourth parameter correspond exactly to the parameters of  
 46692 *open()*.

46693 If *openat()* is passed the special value AT\_FDCWD in the *fd* parameter, the current working  
 46694 directory shall be used and the behavior shall be identical to a call to *open()*.

#### 46695 RETURN VALUE

46696 Upon successful completion, these functions shall open the file and return a non-negative  
 46697 integer representing the file descriptor. Otherwise, these functions shall return -1 and set *errno*  
 46698 to indicate the error. If -1 is returned, no files shall be created or modified.

#### 46699 ERRORS

46700 These functions shall fail if:

46701 [EACCES] Search permission is denied on a component of the path prefix, or the file  
 46702 exists and the permissions specified by *oflag* are denied, or the file does not  
 46703 exist and write permission is denied for the parent directory of the file to be  
 46704 created, or O\_TRUNC is specified and write permission is denied.

46705 [EEXIST] O\_CREAT and O\_EXCL are set, and the named file exists.

46706 [EINTR] A signal was caught during *open()*.

46707 SIO [EINVAL] The implementation does not support synchronized I/O for this file.

46708 OB XSR [EIO] The *path* argument names a STREAMS file and a hangup or error occurred  
 46709 during the *open()*.

46710 [EISDIR] The named file is a directory and *oflag* includes O\_WRONLY or O\_RDWR, or  
 46711 includes O\_CREAT without O\_DIRECTORY.

46712 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*  
 46713 argument, or O\_NOFOLLOW was specified and the *path* argument names a  
 46714 symbolic link.

46715 [EMFILE] All file descriptors available to the process are currently open.

46716 [ENAMETOOLONG]  
 46717 The length of a component of a pathname is longer than {NAME\_MAX}.

46718 [ENFILE] The maximum allowable number of files is currently open in the system.

46719 [ENOENT] O\_CREAT is not set and a component of *path* does not name an existing file, or  
 46720 O\_CREAT is set and a component of the path prefix of *path* does not name an  
 46721 existing file, or *path* points to an empty string.

46722 [ENOENT] or [ENOTDIR]  
 46723 O\_CREAT is set, and the *path* argument contains at least one non-*<slash>*  
 46724 character and ends with one or more trailing *<slash>* characters. If *path*



|       |        |                |                                                                                                |
|-------|--------|----------------|------------------------------------------------------------------------------------------------|
| 46725 |        |                | without the trailing <slash> characters would name an existing file, an                        |
| 46726 |        |                | [ENOENT] error shall not occur.                                                                |
| 46727 | OB XSR | [ENOSR]        | The <i>path</i> argument names a STREAMS-based file and the system is unable to                |
| 46728 |        |                | allocate a STREAM.                                                                             |
| 46729 |        | [ENOSPC]       | The directory or file system that would contain the new file cannot be                         |
| 46730 |        |                | expanded, the file does not exist, and O_CREAT is specified.                                   |
| 46731 |        | [ENOTDIR]      | A component of the path prefix names an existing file that is neither a                        |
| 46732 |        |                | directory nor a symbolic link to a directory; or O_CREAT and O_EXCL are not                    |
| 46733 |        |                | specified, the <i>path</i> argument contains at least one non-<slash> character and            |
| 46734 |        |                | ends with one or more trailing <slash> characters, and the last pathname                       |
| 46735 |        |                | component names an existing file that is neither a directory nor a symbolic                    |
| 46736 |        |                | link to a directory; or O_DIRECTORY was specified and the <i>path</i> argument                 |
| 46737 |        |                | resolves to a non-directory file.                                                              |
| 46738 |        | [ENXIO]        | O_NONBLOCK is set, the named file is a FIFO, O_WRONLY is set, and no                           |
| 46739 |        |                | process has the file open for reading.                                                         |
| 46740 |        | [ENXIO]        | The named file is a character special or block special file, and the device                    |
| 46741 |        |                | associated with this special file does not exist.                                              |
| 46742 |        | [EOVERFLOW]    | The named file is a regular file and the size of the file cannot be represented                |
| 46743 |        |                | correctly in an object of type <b>off_t</b> .                                                  |
| 46744 |        | [EROFS]        | The named file resides on a read-only file system and either O_WRONLY,                         |
| 46745 |        |                | O_RDWR, O_CREAT (if the file does not exist), or O_TRUNC is set in the <i>oflag</i>            |
| 46746 |        |                | argument.                                                                                      |
| 46747 |        |                | The <i>openat()</i> function shall fail if:                                                    |
| 46748 |        | [EACCES]       | The access mode of the open file description associated with <i>fd</i> is not                  |
| 46749 |        |                | O_SEARCH and the permissions of the directory underlying <i>fd</i> do not permit               |
| 46750 |        |                | directory searches.                                                                            |
| 46751 |        | [EBADF]        | The <i>path</i> argument does not specify an absolute path and the <i>fd</i> argument is       |
| 46752 |        |                | neither AT_FDCWD nor a valid file descriptor open for reading or searching.                    |
| 46753 |        | [ENOTDIR]      | The <i>path</i> argument is not an absolute path and <i>fd</i> is a file descriptor associated |
| 46754 |        |                | with a non-directory file.                                                                     |
| 46755 |        |                | These functions may fail if:                                                                   |
| 46756 | XSI    | [EAGAIN]       | The <i>path</i> argument names the slave side of a pseudo-terminal device that is              |
| 46757 |        |                | locked.                                                                                        |
| 46758 |        | [EINVAL]       | The value of the <i>oflag</i> argument is not valid.                                           |
| 46759 |        | [ELOOP]        | More than {SYMLOOP_MAX} symbolic links were encountered during                                 |
| 46760 |        |                | resolution of the <i>path</i> argument.                                                        |
| 46761 |        | [ENAMETOOLONG] |                                                                                                |
| 46762 |        |                | The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a                       |
| 46763 |        |                | symbolic link produced an intermediate result with a length that exceeds                       |
| 46764 |        |                | {PATH_MAX}.                                                                                    |
| 46765 | OB XSR | [ENOMEM]       | The <i>path</i> argument names a STREAMS file and the system is unable to allocate             |
| 46766 |        |                | resources.                                                                                     |



46767 [EOPNOTSUPP] The *path* argument names a socket. +

46768 [ETXTBSY] The file is a pure procedure (shared text) file that is being executed and *oflag* is

46769 O\_WRONLY or O\_RDWR.

## 46770 EXAMPLES

### 46771 Opening a File for Writing by the Owner

46772 The following example opens the file **/tmp/file**, either by creating it (if it does not already exist),

46773 or by truncating its length to 0 (if it does exist). In the former case, if the call creates a new file,

46774 the access permission bits in the file mode of the file are set to permit reading and writing by the

46775 owner, and to permit reading only by group members and others.

46776 If the call to *open()* is successful, the file is opened for writing.

```
46777 #include <fcntl.h>
46778 ...
46779 int fd;
46780 mode_t mode = S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH;
46781 char *pathname = "/tmp/file";
46782 ...
46783 fd = open(pathname, O_WRONLY | O_CREAT | O_TRUNC, mode);
46784 ...
```

### 46785 Opening a File Using an Existence Check

46786 The following example uses the *open()* function to try to create the **LOCKFILE** file and open it

46787 for writing. Since the *open()* function specifies the O\_EXCL flag, the call fails if the file already

46788 exists. In that case, the program assumes that someone else is updating the password file and

46789 exits.

```
46790 #include <fcntl.h>
46791 #include <stdio.h>
46792 #include <stdlib.h>
46793 #define LOCKFILE "/etc/ptmp"
46794 ...
46795 int pfd; /* Integer for file descriptor returned by open() call. */
46796 ...
46797 if ((pfd = open(LOCKFILE, O_WRONLY | O_CREAT | O_EXCL,
46798     S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)) == -1)
46799 {
46800     fprintf(stderr, "Cannot open /etc/ptmp. Try again later.\n");
46801     exit(1);
46802 }
46803 ...
```

## Opening a File for Writing

The following example opens a file for writing, creating the file if it does not already exist. If the file does exist, the system truncates the file to zero bytes.

```
#include <fcntl.h>
#include <stdio.h>
#include <stdlib.h>

#define LOCKFILE "/etc/ptmp"
...
int pfd;
char pathname[PATH_MAX+1];
...
if ((pfd = open(pathname, O_WRONLY | O_CREAT | O_TRUNC,
    S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH)) == -1)
{
    perror("Cannot open output file\n"); exit(1);
}
...
```

## APPLICATION USAGE

POSIX.1-2008 does not require that terminal parameters be automatically set to any state on first open, nor that they be reset after the last close. It is possible for a non-conforming application to leave a terminal device in a state where the next process to use that device finds it in a non-conforming state, but has no way of determining this. To ensure that the device is set to a conforming initial state, applications which perform a first open of a terminal (other than a pseudo-terminal) should do so using the `O_TTY_INIT` flag to set the parameters associated with the terminal to a conforming state.

Except as specified in this volume of POSIX.1-2008, the flags allowed in *oflag* are not mutually-exclusive and any number of them may be used simultaneously. Not all combinations of flags make sense. For example, using `O_SEARCH | O_CREAT` will successfully open a pre-existing directory for searching, but if there is no existing file by that name, then it is unspecified whether a regular file will be created. Likewise, if a non-directory file descriptor is successfully returned, it is unspecified whether that descriptor will have execute permissions as if by `O_EXEC` (note that it is unspecified whether `O_EXEC` and `O_SEARCH` have the same value).

## RATIONALE

Some implementations permit opening FIFOs with `O_RDWR`. Since FIFOs could be implemented in other ways, and since two file descriptors can be used to the same effect, this possibility is left as undefined.

See [\*getgroups\(\)\*](#) about the group of a newly created file.

The use of `open()` to create a regular file is preferable to the use of `creat()`, because the latter is redundant and included only for historical reasons.

The use of the `O_TRUNC` flag on FIFOs and directories (pipes cannot be `open()`-ed) must be permissible without unexpected side-effects (for example, `creat()` on a FIFO must not remove data). Since terminal special files might have type-ahead data stored in the buffer, `O_TRUNC` should not affect their content, particularly if a program that normally opens a regular file should open the current controlling terminal instead. Other file types, particularly implementation-defined ones, are left implementation-defined.

POSIX.1-2008 permits `[EACCES]` to be returned for conditions other than those explicitly listed.

The `O_NOCTTY` flag was added to allow applications to avoid unintentionally acquiring a controlling terminal as a side-effect of opening a terminal file. This volume of POSIX.1-2008 does not specify how a controlling terminal is acquired, but it allows an implementation to provide this on `open()` if the `O_NOCTTY` flag is not set and other conditions specified in XBD Chapter 11 (on page 199) are met.

In historical implementations the value of `O_RDONLY` is zero. Because of that, it is not possible to detect the presence of `O_RDONLY` and another option. Future implementations should encode `O_RDONLY` and `O_WRONLY` as bit flags so that:

```
O_RDONLY | O_WRONLY == O_RDWR
```

`O_EXEC` and `O_SEARCH` are specified as two of the five file access modes. Since `O_EXEC` does not apply to directories, and `O_SEARCH` only applies to directories, their values need not be distinct. Since `O_RDONLY` has historically had the value zero, implementations are not able to distinguish between `O_SEARCH` and `O_SEARCH | O_RDONLY`, and similarly for `O_EXEC`.

In general, the `open()` function follows the symbolic link if *path* names a symbolic link. However, the `open()` function, when called with `O_CREAT` and `O_EXCL`, is required to fail with `[EEXIST]` if *path* names an existing symbolic link, even if the symbolic link refers to a nonexistent file. This behavior is required so that privileged applications can create a new file in a known location without the possibility that a symbolic link might cause the file to be created in a different location.

For example, a privileged application that must create a file with a predictable name in a user-writable directory, such as the user's home directory, could be compromised if the user creates a symbolic link with that name that refers to a nonexistent file in a system directory. If the user can influence the contents of a file, the user could compromise the system by creating a new system configuration or spool file that would then be interpreted by the system. The test for a symbolic link which refers to a nonexistent file must be atomic with the creation of a new file.

In addition, the `open()` function refuses to open non-directories if the `O_DIRECTORY` flag is set. This avoids race conditions whereby a user might compromise the system by substituting a hard link to a sensitive file (e.g., a device or a FIFO) while a privileged application is running, where opening a file even for read access might have undesirable side-effects.

In addition, the `open()` function does not follow symbolic links if the `O_NOFOLLOW` flag is set. This avoids race conditions whereby a user might compromise the system by substituting a symbolic link to a sensitive file (e.g., a device) while a privileged application is running, where opening a file even for read access might have undesirable side-effects.

The POSIX.1-1990 standard required that the group ID of a newly created file be set to the group ID of its parent directory or to the effective group ID of the creating process. FIPS 151-2 required that implementations provide a way to have the group ID be set to the group ID of the containing directory, but did not prohibit implementations also supporting a way to set the group ID to the effective group ID of the creating process. Conforming applications should not assume which group ID will be used. If it matters, an application can use `chown()` to set the group ID after the file is created, or determine under what conditions the implementation will set the desired group ID.

The purpose of the `openat()` function is to enable opening files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to `open()`, resulting in unspecified behavior. By opening a file descriptor for the target directory and using the `openat()` function it can be guaranteed that the opened file is located relative to the desired directory. Some implementations use the `openat()` function for other purposes as well. In some cases, if the *oflag* parameter has the

O\_XATTR bit set, the returned file descriptor provides access to extended attributes. This functionality is not standardized here.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*chmod(), close(), creat(), dirfd(), dup(), exec, fcntl(), fdopendir(), link(), lseek(), mkdtemp(), mknod(), read(), symlink(), umask(), unlockpt(), write()*

XBD Chapter 11 (on page 199), *<fcntl.h>*, *<sys/stat.h>*, *<sys/types.h>*

#### CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

##### Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

Large File Summit extensions are added.

##### Issue 6

In the SYNOPSIS, the optional include of the *<sys/types.h>* header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include *<sys/types.h>* has been removed. Although *<sys/types.h>* was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- In the DESCRIPTION, O\_CREAT is amended to state that the group ID of the file is set to the group ID of the file's parent directory or to the effective group ID of the process. This is a FIPS requirement.
- In the DESCRIPTION, text is added to indicate setting of the offset maximum in the open file description. This change is to support large files.
- In the ERRORS section, the [EOVERFLOW] condition is added. This change is to support large files.
- The [ENXIO] mandatory error condition is added.
- The [EINVAL], [ENAMETOOLONG], and [ETXTBSY] optional error conditions are added.

The DESCRIPTION and ERRORS sections are updated so that items related to the optional XSI STREAMS Option Group are marked.

The following changes were made to align with the IEEE P1003.1a draft standard:

- An explanation is added of the effect of the O\_CREAT and O\_EXCL flags when the path refers to a symbolic link.
- The [ELOOP] optional error condition is added.

The normative text is updated to avoid use of the term “must” for application requirements.

The DESCRIPTION of O\_EXCL is updated in response to IEEE PASC Interpretation 1003.1c #48.

46935 **Issue 7**

- 46936 Austin Group Interpretations 1003.1-2001 #113 and #143 are applied.
- 46937 Austin Group Interpretation 1003.1-2001 #144 is applied, adding the O\_TTY\_INIT flag.
- 46938 Austin Group Interpretation 1003.1-2001 #171 is applied, adding support to set the
- 46939 FD\_CLOEXEC flag atomically at *open()*, and adding the F\_DUPFD\_CLOEXEC flag.
- 46940 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.
- 46941 This page is revised and the *openat()* function is added from The Open Group Technical
- 46942 Standard, 2006, Extended API Set Part 2.
- 46943 Functionality relating to the XSI STREAMS option is marked obsolescent.
- 46944 Changes are made related to support for finegrained timestamps.
- 46945 Changes are made to allow a directory to be opened for searching.
- 46946 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0418 [292], XSH/TC1-2008/0419 [141],
- 46947 XSH/TC1-2008/0420 [461], XSH/TC1-2008/0421 [390], XSH/TC1-2008/0422 [146],
- 46948 XSH/TC1-2008/0423 [324], XSH/TC1-2008/0424 [292], XSH/TC1-2008/0425 [278],
- 46949 XSH/TC1-2008/0426 [278], XSH/TC1-2008/0427 [291], and XSH/TC1-2008/0428 [307] are
- 46950 applied.

46951 **NAME**

46952 open\_memstream, open\_wmemstream — open a dynamic memory buffer stream

46953 **SYNOPSIS**

```

46954 CX    #include <stdio.h>
46955
46955     FILE *open_memstream(char **bufp, size_t *sizep);
46956
46956     #include <wchar.h>
46957
46957     FILE *open_wmemstream(wchar_t **bufp, size_t *sizep);

```

46958 **DESCRIPTION**

46959 The *open\_memstream()* and *open\_wmemstream()* functions shall create an I/O stream associated  
 46960 with a dynamically allocated memory buffer. The stream shall be opened for writing and shall  
 46961 be seekable.

46962 The stream associated with a call to *open\_memstream()* shall be byte-oriented.

46963 The stream associated with a call to *open\_wmemstream()* shall be wide-oriented.

46964 The stream shall maintain a current position in the allocated buffer and a current buffer length.  
 46965 The position shall be initially set to zero (the start of the buffer). Each write to the stream shall  
 46966 start at the current position and move this position by the number of successfully written bytes  
 46967 for *open\_memstream()* or the number of successfully written wide characters for  
 46968 *open\_wmemstream()*. The length shall be initially set to zero. If a write moves the position to a  
 46969 value larger than the current length, the current length shall be set to this position. In this case a  
 46970 null character for *open\_memstream()* or a null wide character for *open\_wmemstream()* shall be  
 46971 appended to the current buffer. For both functions the terminating null is not included in the  
 46972 calculation of the buffer length.

46973 After a successful *fflush()* or *fclose()*, the pointer referenced by *bufp* shall contain the address of  
 46974 the buffer, and the variable pointed to by *sizep* shall contain the smaller of the current buffer  
 46975 length and the number of bytes for *open\_memstream()*, or the number of wide characters for  
 46976 *open\_wmemstream()*, between the beginning of the buffer and the current file position indicator.

46977 After a successful *fflush()* the pointer referenced by *bufp* and the variable referenced by *sizep*  
 46978 remain valid only until the next write operation on the stream or a call to *fclose()*.

46979 After a successful *fclose()*, the pointer referenced by *bufp* can be passed to *free()*. +

46980 **RETURN VALUE**

46981 Upon successful completion, these functions shall return a pointer to the object controlling the  
 46982 stream. Otherwise, a null pointer shall be returned, and *errno* shall be set to indicate the error.

46983 **ERRORS**

46984 These functions shall fail if: +

46985 [EMFILE] {STREAM\_MAX} streams are currently open in the calling process. +

46986 These functions may fail if:

46987 [EINVAL] *bufp* or *sizep* are NULL.

46988 [EMFILE] {FOPEN\_MAX} streams are currently open in the calling process.

46989 [ENOMEM] Memory for the stream or the buffer could not be allocated.

**EXAMPLES**

```
#include <stdio.h>
#include <stdlib.h>

int
main (void)
{
    FILE *stream;
    char *buf;
    size_t len;
    off_t eob;

    stream = open_memstream (&buf, &len);
    if (stream == NULL)
        /* handle error */ ;
    fprintf (stream, "hello my world");
    fflush (stream);
    printf ("buf=%s, len=%zu\n", buf, len);
    eob = ftello(stream);
    fseeko (stream, 0, SEEK_SET);
    fprintf (stream, "good-bye");
    fseeko (stream, eob, SEEK_SET);
    fclose (stream);
    printf ("buf=%s, len=%zu\n", buf, len);
    free (buf);
    return 0;
}
```

This program produces the following output:

```
buf=hello my world, len=14
buf=good-bye world, len=14
```

**APPLICATION USAGE**

The buffer created by these functions should be freed by the application after closing the stream, by means of a call to *free()*.

**RATIONALE**

These functions are similar to *fmemopen()* except that the memory is always allocated dynamically by the function, and the stream is opened only for output.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*fclose()*, *fdopen()*, *fflush()*, *fmemopen()*, *fopen()*, *free()*, *freopen()*

XBD *<stdio.h>*, *<wchar.h>*

**CHANGE HISTORY**

First released in Issue 7.

47031 **NAME**

47032       openat — open file relative to directory file descriptor

47033 **SYNOPSIS**

47034       #include &lt;fcntl.h&gt;

47035       int openat(int *fd*, const char \**path*, int *oflag*, ...);47036 **DESCRIPTION**47037       Refer to *open()*.



47038 **NAME**

47039        opendir — open directory associated with file descriptor

47040 **SYNOPSIS**

47041        #include &lt;dirent.h&gt;

47042        DIR \*opendir(const char \*dirname);

47043 **DESCRIPTION**47044        Refer to *fdopendir()*.

47045 **NAME**

47046       openlog — open a connection to the logging facility

47047 **SYNOPSIS**

47048 XSI       #include &lt;syslog.h&gt;

47049       void openlog(const char \*ident, int logopt, int facility);

47050 **DESCRIPTION**47051       Refer to *closelog()*.

47052 **NAME**

47053       optarg, opterr, optind, optopt — options parsing variables

47054 **SYNOPSIS**

47055       #include &lt;unistd.h&gt;

47056       extern char \*optarg;

47057       extern int opterr, optind, optopt;

47058 **DESCRIPTION**47059       Refer to *getopt()*.

47060 **NAME**

47061 pathconf — get configurable pathname variables

47062 **SYNOPSIS**

47063 #include &lt;unistd.h&gt;

47064 long pathconf(const char \*path, int name);

47065 **DESCRIPTION**47066 Refer to *fpathconf()*.

47067 **NAME**

47068 pause — suspend the thread until a signal is received

47069 **SYNOPSIS**

47070 #include &lt;unistd.h&gt;

47071 int pause(void);

47072 **DESCRIPTION**47073 The *pause()* function shall suspend the calling thread until delivery of a signal whose action is  
47074 either to execute a signal-catching function or to terminate the process.47075 If the action is to terminate the process, *pause()* shall not return.47076 If the action is to execute a signal-catching function, *pause()* shall return after the signal-catching  
47077 function returns.47078 **RETURN VALUE**47079 Since *pause()* suspends thread execution indefinitely unless interrupted by a signal, there is no  
47080 successful completion return value. A value of -1 shall be returned and *errno* set to indicate the  
47081 error.47082 **ERRORS**47083 The *pause()* function shall fail if:47084 [EINTR] A signal is caught by the calling process and control is returned from the  
47085 signal-catching function.47086 **EXAMPLES**

47087 None.

47088 **APPLICATION USAGE**47089 Many common uses of *pause()* have timing windows. The scenario involves checking a  
47090 condition related to a signal and, if the signal has not occurred, calling *pause()*. When the signal  
47091 occurs between the check and the call to *pause()*, the process often blocks indefinitely. The  
47092 *sigprocmask()* and *sigsuspend()* functions can be used to avoid this type of problem.47093 **RATIONALE**

47094 None.

47095 **FUTURE DIRECTIONS**

47096 None.

47097 **SEE ALSO**47098 *sigsuspend()*

47099 XBD &lt;unistd.h&gt;

47100 **CHANGE HISTORY**

47101 First released in Issue 1. Derived from Issue 1 of the SVID.

47102 **Issue 5**

47103 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

47104 **Issue 6**

47105 The APPLICATION USAGE section is added.

**NAME**

pclose — close a pipe stream to or from a process

**SYNOPSIS**

```
#include <stdio.h>

int pclose(FILE *stream);
```

**DESCRIPTION**

The *pclose()* function shall close a stream that was opened by *popen()*, wait for the command to terminate, and return the termination status of the process that was running the command language interpreter. However, if a call caused the termination status to be unavailable to *pclose()*, then *pclose()* shall return  $-1$  with *errno* set to [ECHILD] to report this situation. This can happen if the application calls one of the following functions:

- *wait()*
- *waitpid()* with a *pid* argument less than or equal to 0 or equal to the process ID of the command line interpreter
- Any other function not defined in this volume of POSIX.1-2008 that could do one of the above

In any case, *pclose()* shall not return before the child process created by *popen()* has terminated.

If the command language interpreter cannot be executed, the child termination status returned by *pclose()* shall be as if the command language interpreter terminated using *exit(127)* or *\_exit(127)*.

The *pclose()* function shall not affect the termination status of any child of the calling process other than the one created by *popen()* for the associated stream.

If the argument *stream* to *pclose()* is not a pointer to a stream created by *popen()*, the result of *pclose()* is undefined.

If a thread is canceled during execution of *pclose()*, the behavior is undefined.

+

**RETURN VALUE**

Upon successful return, *pclose()* shall return the termination status of the command language interpreter. Otherwise, *pclose()* shall return  $-1$  and set *errno* to indicate the error.

**ERRORS**

The *pclose()* function shall fail if:

[ECHILD] The status of the child process could not be obtained, as described above.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

There is a requirement that *pclose()* not return before the child process terminates. This is intended to disallow implementations that return [EINTR] if a signal is received while waiting. If *pclose()* returned before the child terminated, there would be no way for the application to discover which child used to be associated with the stream, and it could not do the cleanup itself.

If the stream pointed to by *stream* was not created by *popen()*, historical implementations of

*pclose()* return `-1` without setting *errno*. To avoid requiring *pclose()* to set *errno* in this case, POSIX.1-2008 makes the behavior unspecified. An application should not use *pclose()* to close any stream that was not created by *popen()*.

Some historical implementations of *pclose()* either block or ignore the signals `SIGINT`, `SIGQUIT`, and `SIGHUP` while waiting for the child process to terminate. Since this behavior is not described for the *pclose()* function in POSIX.1-2008, such implementations are not conforming. Also, some historical implementations return `[EINTR]` if a signal is received, even though the child process has not terminated. Such implementations are also considered non-conforming.

Consider, for example, an application that uses:

```
popen("command", "r")
```

to start *command*, which is part of the same application. The parent writes a prompt to its standard output (presumably the terminal) and then reads from the *popen()*ed stream. The child reads the response from the user, does some transformation on the response (pathname expansion, perhaps) and writes the result to its standard output. The parent process reads the result from the pipe, does something with it, and prints another prompt. The cycle repeats. Assuming that both processes do appropriate buffer flushing, this would be expected to work.

To conform to POSIX.1-2008, *pclose()* must use *waitpid()*, or some similar function, instead of *wait()*.

The code sample below illustrates how the *pclose()* function might be implemented on a system conforming to POSIX.1-2008.

```
int pclose(FILE *stream)
{
    int stat;
    pid_t pid;

    pid = <pid for process created for stream by popen(>
    (void) fclose(stream);
    while (waitpid(pid, &stat, 0) == -1) {
        if (errno != EINTR){
            stat = -1;
            break;
        }
    }
    return(stat);
}
```

## FUTURE DIRECTIONS

None.

## SEE ALSO

*fork()*, *popen()*, *wait()*

XBD `<stdio.h>`

## CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

47189 **NAME**

47190 perror — write error messages to standard error

47191 **SYNOPSIS**

47192 #include &lt;stdio.h&gt;

47193 void perror(const char \*s);

47194 **DESCRIPTION**

47195 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 47196 conflict between the requirements described here and the ISO C standard is unintentional. This  
 47197 volume of POSIX.1-2008 defers to the ISO C standard.

47198 The *perror()* function shall map the error number accessed through the symbol *errno* to a  
 47199 language-dependent error message, which shall be written to the standard error stream as  
 47200 follows:

- 47201 • First (if *s* is not a null pointer and the character pointed to by *s* is not the null byte), the  
 47202 string pointed to by *s* followed by a <colon> and a <space>.
- 47203 • Then an error message string followed by a <newline>.

47204 The contents of the error message strings shall be the same as those returned by *strerror()* with  
 47205 argument *errno*.

47206 CX The *perror()* function shall mark for update the last data modification and last file status change  
 47207 timestamps of the file associated with the standard error stream at some time between its  
 47208 successful completion and *exit()*, *abort()*, or the completion of *fflush()* or *fclose()* on *stderr*.

47209 The *perror()* function shall not change the orientation of the standard error stream.

47210 On error, *perror()* shall set the error indicator for the stream to which *stderr* points, and shall set  
 47211 *errno* to indicate the error.

47212 Since no value is returned, an application wishing to check for error situations should call  
 47213 *clearerr(stderr)* before calling *perror()*, then if *ferror(stderr)* returns non-zero, the value of *errno*  
 47214 indicates which error occurred.

47215 **RETURN VALUE**47216 The *perror()* function shall not return a value.47217 **ERRORS**47218 CX Refer to *fputc()*.47219 **EXAMPLES**47220 **Printing an Error Message for a Function**

47221 The following example replaces *bufptr* with a buffer that is the necessary size. If an error occurs,  
 47222 the *perror()* function prints a message and the program exits.

```

47223 #include <stdio.h>
47224 #include <stdlib.h>
47225 ...
47226 char *bufptr;
47227 size_t szbuf;
47228 ...
47229 if ((bufptr = malloc(szbuf)) == NULL) {
47230     perror("malloc"); exit(2);
47231 }
```



47232 . . .

#### 47233 APPLICATION USAGE

47234 Application writers may prefer to use alternative interfaces instead of *perror()*, such as  
47235 *strerror\_r()* in combination with *fprintf()*.

#### 47236 RATIONALE

47237 None.

#### 47238 FUTURE DIRECTIONS

47239 None.

#### 47240 SEE ALSO

47241 *fprintf()*, *fputc()*, *psiginfo()*, *strerror()*

47242 XBD <stdio.h>

#### 47243 CHANGE HISTORY

47244 First released in Issue 1. Derived from Issue 1 of the SVID.

##### 47245 Issue 5

47246 A paragraph is added to the DESCRIPTION indicating that *perror()* does not change the  
47247 orientation of the standard error stream.

##### 47248 Issue 6

47249 Extensions beyond the ISO C standard are marked.

##### 47250 Issue 7

47251 SD5-XSH-ERN-95 is applied.

47252 Changes are made related to support for finegrained timestamps.

47253 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0429 [389,401], XSH/TC1-2008/0430  
47254 [389], and XSH/TC1-2008/0431 [389,401] are applied.

**NAME**

pipe — create an interprocess channel

**SYNOPSIS**

```
#include <unistd.h>

int pipe(int fildes[2]);
```

**DESCRIPTION**

The *pipe()* function shall create a pipe and place two file descriptors, one each into the arguments *fildes*[0] and *fildes*[1], that refer to the open file descriptions for the read and write ends of the pipe. The file descriptors shall be allocated as described in [Section 2.14](#) (on page 549). The O\_NONBLOCK and FD\_CLOEXEC flags shall be clear on both file descriptors. (The *fcntl()* function can be used to set both these flags.)

Data can be written to the file descriptor *fildes*[1] and read from the file descriptor *fildes*[0]. A read on the file descriptor *fildes*[0] shall access data written to the file descriptor *fildes*[1] on a first-in-first-out basis. It is unspecified whether *fildes*[0] is also open for writing and whether *fildes*[1] is also open for reading.

A process has the pipe open for reading (correspondingly writing) if it has a file descriptor open that refers to the read end, *fildes*[0] (write end, *fildes*[1]).

The pipe's user ID shall be set to the effective user ID of the calling process.

The pipe's group ID shall be set to the effective group ID of the calling process.

Upon successful completion, *pipe()* shall mark for update the last data access, last data modification, and last file status change timestamps of the pipe.

**RETURN VALUE**

Upon successful completion, 0 shall be returned; otherwise, -1 shall be returned and *errno* set to indicate the error, no file descriptors shall be allocated and the contents of *fildes* shall be left unmodified.

**ERRORS**

The *pipe()* function shall fail if:

- |          |                                                                                            |
|----------|--------------------------------------------------------------------------------------------|
| [EMFILE] | All, or all but one, of the file descriptors available to the process are currently open.  |
| [ENFILE] | The number of simultaneously open files in the system would exceed a system-imposed limit. |

**EXAMPLES****Using a Pipe to Pass Data Between a Parent Process and a Child Process**

The following example demonstrates the use of a pipe to transfer data between a parent process and a child process. Error handling is excluded, but otherwise this code demonstrates good practice when using pipes: after the *fork()* the two processes close the unused ends of the pipe before they commence transferring data.

```
#include <stdlib.h>
#include <unistd.h>
...

int fildes[2];
const int BSIZE = 100;
char buf[BSIZE];
```

```

47298     ssize_t nbytes;
47299     int status;

47300     status = pipe(fildes);
47301     if (status == -1 ) {
47302         /* an error occurred */
47303         ...
47304     }

47305     switch (fork()) {
47306     case -1: /* Handle error */
47307         break;

47308     case 0: /* Child - reads from pipe */
47309         close(fildes[1]); /* Write end is unused */
47310         nbytes = read(fildes[0], buf, BSIZE); /* Get data from pipe */
47311         /* At this point, a further read would see end-of-file ... */
47312         close(fildes[0]); /* Finished with pipe */
47313         exit(EXIT_SUCCESS);

47314     default: /* Parent - writes to pipe */
47315         close(fildes[0]); /* Read end is unused */
47316         write(fildes[1], "Hello world\n", 12); /* Write data on pipe */
47317         close(fildes[1]); /* Child will see EOF */
47318         exit(EXIT_SUCCESS);
47319     }

```

**APPLICATION USAGE**

None.

**RATIONALE**

The wording carefully avoids using the verb “to open” in order to avoid any implication of use of *open()*; see also *write()*.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 2.14](#) (on page 549), *fcntl()*, *read()*, *write()*

XBD [<fcntl.h>](#), [<unistd.h>](#)

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 6**

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION is updated to indicate that certain dispositions of *fildes[0]* and *fildes[1]* are unspecified.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/65 is applied, adding the example to the EXAMPLES section.

47339 **Issue 7**

47340 SD5-XSH-ERN-156 is applied, updating the DESCRIPTION to state the setting of the pipe's user  
47341 ID and group ID.

47342 Changes are made related to support for finegrained timestamps.

47343 **NAME**

47344 poll — input/output multiplexing

47345 **SYNOPSIS**

47346 #include &lt;poll.h&gt;

47347 int poll(struct pollfd *fds*[], nfds\_t *nfds*, int *timeout*);47348 **DESCRIPTION**

47349 The *poll()* function provides applications with a mechanism for multiplexing input/output over  
 47350 a set of file descriptors. For each member of the array pointed to by *fds*, *poll()* shall examine the  
 47351 given file descriptor for the event(s) specified in *events*. The number of **pollfd** structures in the  
 47352 *fds* array is specified by *nfds*. The *poll()* function shall identify those file descriptors on which an  
 47353 application can read or write data, or on which certain events have occurred.

47354 The *fds* argument specifies the file descriptors to be examined and the events of interest for each  
 47355 file descriptor. It is a pointer to an array with one member for each open file descriptor of  
 47356 interest. The array's members are **pollfd** structures within which *fd* specifies an open file  
 47357 descriptor and *events* and *revents* are bitmasks constructed by OR'ing a combination of the  
 47358 following event flags:

47359 **POLLIN** Data other than high-priority data may be read without blocking.

47360 OB XSR For STREAMS, this flag is set in *revents* even if the message is of zero length.  
 47361 This flag shall be equivalent to **POLLRDNORM** | **POLLRDBAND**.

47362 **POLLRDNORM** Normal data may be read without blocking.

47363 OB XSR For STREAMS, data on priority band 0 may be read without blocking. This  
 47364 flag is set in *revents* even if the message is of zero length.

47365 **POLLRDBAND** Priority data may be read without blocking.

47366 OB XSR For STREAMS, data on priority bands greater than 0 may be read without  
 47367 blocking. This flag is set in *revents* even if the message is of zero length.

47368 **POLLPRI** High-priority data may be read without blocking.

47369 OB XSR For STREAMS, this flag is set in *revents* even if the message is of zero length.

47370 **POLLOUT** Normal data may be written without blocking.

47371 OB XSR For STREAMS, data on priority band 0 may be written without blocking.

47372 **POLLWRNORM** Equivalent to **POLLOUT**.

47373 **POLLWRBAND** Priority data may be written.

47374 OB XSR For STREAMS, data on priority bands greater than 0 may be written without  
 47375 blocking. If any priority band has been written to on this STREAM, this event  
 47376 only examines bands that have been written to at least once.

47377 **POLLERR** An error has occurred on the device or stream. This flag is only valid in the  
 47378 *revents* bitmask; it shall be ignored in the *events* member.

47379 **POLLHUP** A device has been disconnected, or a pipe or FIFO has been closed by the last  
 47380 process that had it open for writing. Once set, the hangup state of a FIFO shall  
 47381 persist until some process opens the FIFO for writing or until all read-only file  
 47382 descriptors for the FIFO are closed. This event and **POLLOUT** are mutually-  
 47383 exclusive; a stream can never be writable if a hangup has occurred. However,  
 47384 this event and **POLLIN**, **POLLRDNORM**, **POLLRDBAND**, or **POLLPRI** are  
 47385 not mutually-exclusive. This flag is only valid in the *revents* bitmask; it shall be

|       |                     |                                                                                                                                       |
|-------|---------------------|---------------------------------------------------------------------------------------------------------------------------------------|
| 47386 |                     | ignored in the <i>events</i> member.                                                                                                  |
| 47387 | POLLNVAL            | The specified <i>fd</i> value is invalid. This flag is only valid in the <i>revents</i> member;                                       |
| 47388 |                     | it shall ignored in the <i>events</i> member.                                                                                         |
| 47389 |                     | The significance and semantics of normal, priority, and high-priority data are file and device-                                       |
| 47390 |                     | specific.                                                                                                                             |
| 47391 |                     | If the value of <i>fd</i> is less than 0, <i>events</i> shall be ignored, and <i>revents</i> shall be set to 0 in that entry on       |
| 47392 |                     | return from <i>poll()</i> .                                                                                                           |
| 47393 |                     | In each <b>pollfd</b> structure, <i>poll()</i> shall clear the <i>revents</i> member, except that where the application               |
| 47394 |                     | requested a report on a condition by setting one of the bits of <i>events</i> listed above, <i>poll()</i> shall set                   |
| 47395 |                     | the corresponding bit in <i>revents</i> if the requested condition is true. In addition, <i>poll()</i> shall set the                  |
| 47396 |                     | POLLHUP, POLLERR, and POLLNVAL flag in <i>revents</i> if the condition is true, even if the                                           |
| 47397 |                     | application did not set the corresponding bit in <i>events</i> .                                                                      |
| 47398 |                     | If none of the defined events have occurred on any selected file descriptor, <i>poll()</i> shall wait at                              |
| 47399 |                     | least <i>timeout</i> milliseconds for an event to occur on any of the selected file descriptors. If the value                         |
| 47400 |                     | of <i>timeout</i> is 0, <i>poll()</i> shall return immediately. If the value of <i>timeout</i> is -1, <i>poll()</i> shall block until |
| 47401 |                     | a requested event occurs or until the call is interrupted.                                                                            |
| 47402 |                     | Implementations may place limitations on the granularity of timeout intervals. If the requested                                       |
| 47403 |                     | timeout interval requires a finer granularity than the implementation supports, the actual                                            |
| 47404 |                     | timeout interval shall be rounded up to the next supported value.                                                                     |
| 47405 |                     | The <i>poll()</i> function shall not be affected by the O_NONBLOCK flag.                                                              |
| 47406 |                     | The <i>poll()</i> function shall support regular files, terminal and pseudo-terminal devices, FIFOs,                                  |
| 47407 | OB XSR              | pipes, sockets and <b>STREAMS-based files</b> . The behavior of <i>poll()</i> on elements of <i>fds</i> that refer to                 |
| 47408 |                     | other types of file is unspecified.                                                                                                   |
| 47409 |                     | Regular files shall always poll TRUE for reading and writing.                                                                         |
| 47410 |                     | A file descriptor for a socket that is listening for connections shall indicate that it is ready for                                  |
| 47411 |                     | reading, once connections are available. A file descriptor for a socket that is connecting                                            |
| 47412 |                     | asynchronously shall indicate that it is ready for writing, once a connection has been established.                                   |
| 47413 |                     | Provided the application does not perform any action that results in unspecified or undefined +                                       |
| 47414 |                     | behavior, the value of the <i>fd</i> and <i>events</i> members of each element of <i>fds</i> shall not be modified by +               |
| 47415 |                     | <i>poll()</i> .                                                                                                                       |
| 47416 | <b>RETURN VALUE</b> |                                                                                                                                       |
| 47417 |                     | Upon successful completion, <i>poll()</i> shall return a non-negative value. A positive value indicates                               |
| 47418 |                     | the total number of <b>pollfd</b> structures that have selected events (that is, those for which the                                  |
| 47419 |                     | <i>revents</i> member is non-zero). A value of 0 indicates that the call timed out and no file descriptors                            |
| 47420 |                     | have been selected. Upon failure, <i>poll()</i> shall return -1 and set <i>errno</i> to indicate the error.                           |
| 47421 | <b>ERRORS</b>       |                                                                                                                                       |
| 47422 |                     | The <i>poll()</i> function shall fail if:                                                                                             |
| 47423 | [EAGAIN]            | The allocation of internal data structures failed but a subsequent request may                                                        |
| 47424 |                     | succeed.                                                                                                                              |
| 47425 | [EINTR]             | A signal was caught during <i>poll()</i> .                                                                                            |
| 47426 | OB XSR [EINVAL]     | The <i>nfds</i> argument is greater than {OPEN_MAX}, or one of the <i>fd</i> members                                                  |
| 47427 |                     | refers to a <b>STREAM</b> or multiplexer that is linked (directly or indirectly)                                                      |
| 47428 |                     | <b>downstream from a multiplexer.</b>                                                                                                 |

## EXAMPLES

## Checking for Events on a Stream

The following example opens a pair of STREAMS devices and then waits for either one to become writable. This example proceeds as follows:

1. Sets the *timeout* parameter to 500 milliseconds.
2. Opens the STREAMS devices */dev/dev0* and */dev/dev1*, and then polls them, specifying POLLOUT and POLLWRBAND as the events of interest.

The STREAMS device names */dev/dev0* and */dev/dev1* are only examples of how STREAMS devices can be named; STREAMS naming conventions may vary among systems conforming to the POSIX.1-2008.

3. Uses the *ret* variable to determine whether an event has occurred on either of the two STREAMS. The *poll()* function is given 500 milliseconds to wait for an event to occur (if it has not occurred prior to the *poll()* call).
4. Checks the returned value of *ret*. If a positive value is returned, one of the following can be done:
  - a. Priority data can be written to the open STREAM on priority bands greater than 0, because the POLLWRBAND event occurred on the open STREAM (*fds[0]* or *fds[1]*).
  - b. Data can be written to the open STREAM on priority-band 0, because the POLLOUT event occurred on the open STREAM (*fds[0]* or *fds[1]*).
5. If the returned value is not a positive value, permission to write data to the open STREAM (on any priority band) is denied.
6. If the POLLHUP event occurs on the open STREAM (*fds[0]* or *fds[1]*), the device on the open STREAM has disconnected.

```
#include <stropts.h>
#include <poll.h>
...
struct pollfd fds[2];
int timeout_msecs = 500;
int ret;
int i;

/* Open STREAMS device. */
fds[0].fd = open("/dev/dev0", ...);
fds[1].fd = open("/dev/dev1", ...);
fds[0].events = POLLOUT | POLLWRBAND;
fds[1].events = POLLOUT | POLLWRBAND;

ret = poll(fds, 2, timeout_msecs);

if (ret > 0) {
    /* An event on one of the fds has occurred. */
    for (i=0; i<2; i++) {
        if (fds[i].revents & POLLWRBAND) {
            /* Priority data may be written on device number i. */
            ...
        }
    }
}
```

```
47472         if (fds[i].revents & POLLOUT) {
47473             /* Data may be written on device number i. */
47474             ...
47475         }
47476         if (fds[i].revents & POLLHUP) {
47477             /* A hangup has occurred on device number i. */
47478             ...
47479         }
47480     }
47481 }
```

#### 47482 APPLICATION USAGE

47483 None.

#### 47484 RATIONALE

47485 The POLLHUP event does not occur for FIFOs just because the FIFO is not open for writing. It  
47486 only occurs when the FIFO is closed by the last writer and persists until some process opens the  
47487 FIFO for writing or until all read-only file descriptors for the FIFO are closed.

#### 47488 FUTURE DIRECTIONS

47489 None.

#### 47490 SEE ALSO

47491 [Section 2.6](#) (on page 500), [getmsg\(\)](#), [pselect\(\)](#), [putmsg\(\)](#), [read\(\)](#), [write\(\)](#)

47492 XBD [<poll.h>](#), [<stropts.h>](#)

#### 47493 CHANGE HISTORY

47494 First released in Issue 4, Version 2.

##### 47495 Issue 5

47496 Moved from X/OPEN UNIX extension to BASE.

47497 The description of POLLWRBAND is updated.

##### 47498 Issue 6

47499 Text referring to sockets is added to the DESCRIPTION.

47500 Functionality relating to the XSI STREAMS Option Group is marked.

47501 The Open Group Corrigendum U055/3 is applied, updating the DESCRIPTION of  
47502 POLLWRBAND.

47503 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/66 is applied, correcting the spacing in  
47504 the EXAMPLES section.

##### 47505 Issue 7

47506 Austin Group Interpretation 1003.1-2001 #209 is applied, clarifying the POLLHUP event.

47507 The *poll()* function is moved from the XSI option to the Base.

47508 Functionality relating to the XSI STREAMS option is marked obsolescent.



**NAME**

popen — initiate pipe streams to or from a process

**SYNOPSIS**

```
#include <stdio.h>
```

```
FILE *popen(const char *command, const char *mode);
```

**DESCRIPTION**

The *popen()* function shall execute the command specified by the string *command*. It shall create a pipe between the calling program and the executed command, and shall return a pointer to a stream that can be used to either read from or write to the pipe.

The environment of the executed command shall be as if a child process were created within the *popen()* call using the *fork()* function, and the child invoked the *sh* utility using the call:

```
execl(shell_path, "sh", "-c", command, (char *)0);
```

where *shell\_path* is an unspecified pathname for the *sh* utility.

The *popen()* function shall ensure that any streams from previous *popen()* calls that remain open in the parent process are closed in the new child process.

The *mode* argument to *popen()* is a string that specifies I/O mode:

1. If *mode* is *r*, when the child process is started, its file descriptor *STDOUT\_FILENO* shall be the writable end of the pipe, and the file descriptor *fileno(stream)* in the calling process, where *stream* is the stream pointer returned by *popen()*, shall be the readable end of the pipe.
2. If *mode* is *w*, when the child process is started its file descriptor *STDIN\_FILENO* shall be the readable end of the pipe, and the file descriptor *fileno(stream)* in the calling process, where *stream* is the stream pointer returned by *popen()*, shall be the writable end of the pipe.
3. If *mode* is any other value, the result is unspecified.

After *popen()*, both the parent and the child process shall be capable of executing independently before either terminates.

Pipe streams are byte-oriented.

**RETURN VALUE**

Upon successful completion, *popen()* shall return a pointer to an open stream that can be used to read or write to the pipe. Otherwise, it shall return a null pointer and may set *errno* to indicate the error.

**ERRORS**

The *popen()* function shall fail if:

[EMFILE] {STREAM\_MAX} streams are currently open in the calling process.

The *popen()* function may fail if:

[EMFILE] {FOPEN\_MAX} streams are currently open in the calling process.

[EINVAL] The *mode* argument is invalid.

The *popen()* function may also set *errno* values as described by *fork()* or *pipe()*.

## EXAMPLES

## Using popen() to Obtain a List of Files from the ls Utility

The following example demonstrates the use of *popen()* and *pclose()* to execute the command *ls* in order to obtain a list of files in the current directory:

```
#include <stdio.h>
...

FILE *fp;
int status;
char path[PATH_MAX];

fp = popen("ls *", "r");
if (fp == NULL)
    /* Handle error */;

while (fgets(path, PATH_MAX, fp) != NULL)
    printf("%s", path);

status = pclose(fp);
if (status == -1) {
    /* Error reported by pclose() */
    ...
} else {
    /* Use macros described under wait() to inspect 'status' in order
       to determine success/failure of command executed by popen() */
    ...
}
```

## APPLICATION USAGE

Since open files are shared, a mode *r* command can be used as an input filter and a mode *w* command as an output filter.

Buffered reading before opening an input filter may leave the standard input of that filter mispositioned. Similar problems with an output filter may be prevented by careful buffer flushing; for example, with *fflush()*.

A stream opened by *popen()* should be closed by *pclose()*.

The behavior of *popen()* is specified for values of *mode* of *r* and *w*. Other modes such as *rb* and *wb* might be supported by specific implementations, but these would not be portable features. Note that historical implementations of *popen()* only check to see if the first character of *mode* is *r*. Thus, a *mode* of *robert the robot* would be treated as *mode r*, and a *mode* of *anything else* would be treated as *mode w*.

If the application calls *waitpid()* or *waitid()* with a *pid* argument greater than 0, and it still has a stream that was called with *popen()* open, it must ensure that *pid* does not refer to the process started by *popen()*.

To determine whether or not the environment specified in the Shell and Utilities volume of POSIX.1-2008 is present, use the function call:

```
sysconf(_SC_2_VERSION)
```

(See *sysconf()*).

**RATIONALE**

The *popen()* function should not be used by programs that have set user (or group) ID privileges. The *fork()* and *exec* family of functions (except *execlp()* and *execvp()*), should be used instead. This prevents any unforeseen manipulation of the environment of the user that could cause execution of commands not anticipated by the calling program.

If the original and *popen()*ed processes both intend to read or write or read and write a common file, and either will be using FILE-type C functions (*fread()*, *fwrite()*, and so on), the rules for sharing file handles must be observed (see [Section 2.5.1](#), on page 497).

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 2.5](#) (on page 495), *fork()*, *pclose()*, *pipe()*, *sysconf()*, *system()*, *wait()*, *waitid()*

XBD [<stdio.h>](#)

XCU *sh*

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

A statement is added to the DESCRIPTION indicating that pipe streams are byte-oriented.

**Issue 6**

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The optional [EMFILE] error condition is added.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/67 is applied, adding the example to the EXAMPLES section.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #029 is applied, clarifying the values for *mode* in the DESCRIPTION.

SD5-XSH-ERN-149 is applied, changing the {STREAM\_MAX} [EMFILE] error condition from a “may fail” to a “shall fail”.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0432 [14] is applied.

**NAME**

posix\_fadvise — file advisory information (**ADVANCED REALTIME**)

**SYNOPSIS**

```
#include <fcntl.h>

int posix_fadvise(int fd, off_t offset, off_t len, int advice);
```

**DESCRIPTION**

The *posix\_fadvise()* function shall advise the implementation on the expected behavior of the application with respect to the data in the file associated with the open file descriptor, *fd*, starting at *offset* and continuing for *len* bytes. The specified range need not currently exist in the file. If *len* is zero, all data following *offset* is specified. The implementation may use this information to optimize handling of the specified data. The *posix\_fadvise()* function shall have no effect on the semantics of other operations on the specified data, although it may affect the performance of other operations.

The advice to be applied to the data is specified by the *advice* parameter and may be one of the following values:

**POSIX\_FADV\_NORMAL**

Specifies that the application has no advice to give on its behavior with respect to the specified data. It is the default characteristic if no advice is given for an open file.

**POSIX\_FADV\_SEQUENTIAL**

Specifies that the application expects to access the specified data sequentially from lower offsets to higher offsets.

**POSIX\_FADV\_RANDOM**

Specifies that the application expects to access the specified data in a random order.

**POSIX\_FADV\_WILLNEED**

Specifies that the application expects to access the specified data in the near future.

**POSIX\_FADV\_DONTNEED**

Specifies that the application expects that it will not access the specified data in the near future.

**POSIX\_FADV\_NOREUSE**

Specifies that the application expects to access the specified data once and then not reuse it thereafter.

These values are defined in **<fcntl.h>**.

**RETURN VALUE**

Upon successful completion, *posix\_fadvise()* shall return zero; otherwise, an error number shall be returned to indicate the error.

**ERRORS**

The *posix\_fadvise()* function shall fail if:

- |          |                                                                                      |
|----------|--------------------------------------------------------------------------------------|
| [EBADF]  | The <i>fd</i> argument is not a valid file descriptor.                               |
| [EINVAL] | The value of <i>advice</i> is invalid, or the value of <i>len</i> is less than zero. |
| [ESPIPE] | The <i>fd</i> argument is associated with a pipe or FIFO.                            |

47660 **EXAMPLES**

47661 None.

47662 **APPLICATION USAGE**

47663 The *posix\_fadvise()* function is part of the Advisory Information option and need not be  
47664 provided on all implementations.

47665 **RATIONALE**

47666 None.

47667 **FUTURE DIRECTIONS**

47668 None.

47669 **SEE ALSO**47670 *posix\_madvise()*

47671 XBD &lt;fcntl.h&gt;

47672 **CHANGE HISTORY**

47673 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

47674 In the SYNOPSIS, the inclusion of &lt;sys/types.h&gt; is no longer required.

47675 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/68 is applied, changing the function  
47676 prototype in the SYNOPSIS section. The previous prototype was not large file-aware, and the  
47677 standard developers felt it acceptable to make this change before implementations of this  
47678 function become widespread.

47679 **Issue 7**

47680 Austin Group Interpretation 1003.1-2001 #024 is applied, changing the definition of the  
47681 [EINVAL] error.

## NAME

posix\_fallocate — file space control (ADVANCED REALTIME)

## SYNOPSIS

```
#include <fcntl.h>

int posix_fallocate(int fd, off_t offset, off_t len);
```

## DESCRIPTION

The *posix\_fallocate()* function shall ensure that any required storage for regular file data starting at *offset* and continuing for *len* bytes is allocated on the file system storage media. If *posix\_fallocate()* returns successfully, subsequent writes to the specified file data shall not fail due to the lack of free space on the file system storage media.

If the *offset+len* is beyond the current file size, then *posix\_fallocate()* shall adjust the file size to *offset+len*. Otherwise, the file size shall not be changed.

It is implementation-defined whether a previous *posix\_fadvise()* call influences allocation strategy.

Space allocated via *posix\_fallocate()* shall be freed by a successful call to *creat()* or *open()* that truncates the size of the file. Space allocated via *posix\_fallocate()* may be freed by a successful call to *ftruncate()* that reduces the file size to a size smaller than *offset+len*.

## RETURN VALUE

Upon successful completion, *posix\_fallocate()* shall return zero; otherwise, an error number shall be returned to indicate the error.

## ERRORS

The *posix\_fallocate()* function shall fail if:

- [EBADF] The *fd* argument is not a valid file descriptor.
  - [EBADF] The *fd* argument references a file that was opened without write permission.
  - [EFBIG] The value of *offset+len* is greater than the maximum file size.
  - [EINTR] A signal was caught during execution.
  - [EINVAL] The *len* argument is less than zero, or the *offset* argument is less than zero, or the underlying file system does not support this operation.
  - [EIO] An I/O error occurred while reading from or writing to a file system.
  - [ENODEV] The *fd* argument does not refer to a regular file.
  - [ENOSPC] There is insufficient free space remaining on the file system storage media.
  - [ESPIPE] The *fd* argument is associated with a pipe or FIFO.
- The *posix\_fallocate()* function may fail if:
- [EINVAL] The *len* argument is zero.

47716 **EXAMPLES**

47717 None.

47718 **APPLICATION USAGE**

47719 The *posix\_fallocate()* function is part of the Advisory Information option and need not be  
47720 provided on all implementations.

47721 **RATIONALE**

47722 None.

47723 **FUTURE DIRECTIONS**

47724 None.

47725 **SEE ALSO**47726 *creat()*, *ftruncate()*, *open()*, *unlink()*

47727 XBD &lt;fcntl.h&gt;

47728 **CHANGE HISTORY**

47729 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

47730 In the SYNOPSIS, the inclusion of &lt;sys/types.h&gt; is no longer required.

47731 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/69 is applied, changing the function  
47732 prototype in the SYNOPSIS section. The previous prototype was not large file-aware, and the  
47733 standard developers felt it acceptable to make this change before implementations of this  
47734 function become widespread.

47735 **Issue 7**

47736 Austin Group Interpretations 1003.1-2001 #022, #024, and #162 are applied, changing the  
47737 definition of the [EINVAL] error.

## NAME

posix\_madvise — memory advisory information and alignment control (**ADVANCED REALTIME**)

## SYNOPSIS

```
#include <sys/mman.h>

int posix_madvise(void *addr, size_t len, int advice);
```

## DESCRIPTION

The *posix\_madvise()* function shall advise the implementation on the expected behavior of the application with respect to the data in the memory starting at address *addr*, and continuing for *len* bytes. The implementation may use this information to optimize handling of the specified data. The *posix\_madvise()* function shall have no effect on the semantics of access to memory in the specified range, although it may affect the performance of access.

The implementation may require that *addr* be a multiple of the page size, which is the value returned by *sysconf()* when the name value *\_SC\_PAGESIZE* is used.

The advice to be applied to the memory range is specified by the *advice* parameter and may be one of the following values:

### POSIX\_MADV\_NORMAL

Specifies that the application has no advice to give on its behavior with respect to the specified range. It is the default characteristic if no advice is given for a range of memory.

### POSIX\_MADV\_SEQUENTIAL

Specifies that the application expects to access the specified range sequentially from lower addresses to higher addresses.

### POSIX\_MADV\_RANDOM

Specifies that the application expects to access the specified range in a random order.

### POSIX\_MADV\_WILLNEED

Specifies that the application expects to access the specified range in the near future.

### POSIX\_MADV\_DONTNEED

Specifies that the application expects that it will not access the specified range in the near future.

These values are defined in the **<sys/mman.h>** header.

## RETURN VALUE

Upon successful completion, *posix\_madvise()* shall return zero; otherwise, an error number shall be returned to indicate the error.

## ERRORS

The *posix\_madvise()* function shall fail if:

[EINVAL] The value of *advice* is invalid.

[ENOMEM] Addresses in the range starting at *addr* and continuing for *len* bytes are partly or completely outside the range allowed for the address space of the calling process.



47777 The *posix\_madvise()* function may fail if:

47778 [EINVAL] The value of *addr* is not a multiple of the value returned by *sysconf()* when the  
47779 name value *\_SC\_PAGESIZE* is used.

47780 [EINVAL] The value of *len* is zero.

#### 47781 EXAMPLES

47782 None.

#### 47783 APPLICATION USAGE

47784 The *posix\_madvise()* function is part of the Advisory Information option and need not be  
47785 provided on all implementations.

#### 47786 RATIONALE

47787 None.

#### 47788 FUTURE DIRECTIONS

47789 None.

#### 47790 SEE ALSO

47791 *mmap()*, *posix\_fadvise()*, *sysconf()*

47792 XBD <sys/mman.h>

#### 47793 CHANGE HISTORY

47794 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

47795 IEEE PASC Interpretation 1003.1 #102 is applied.

**NAME**

posix\_mem\_offset — find offset and length of a mapped typed memory block (**ADVANCED REALTIME**)

**SYNOPSIS**

```
#include <sys/mman.h>
```

```
int posix_mem_offset(const void *restrict addr, size_t len,
    off_t *restrict off, size_t *restrict contig_len,
    int *restrict fildes);
```

**DESCRIPTION**

The *posix\_mem\_offset()* function shall return in the variable pointed to by *off* a value that identifies the offset (or location), within a memory object, of the memory block currently mapped at *addr*. The function shall return in the variable pointed to by *fildes*, the descriptor used (via *mmap()*) to establish the mapping which contains *addr*. If that descriptor was closed since the mapping was established, the returned value of *fildes* shall be  $-1$ . The *len* argument specifies the length of the block of the memory object the user wishes the offset for; upon return, the value pointed to by *contig\_len* shall equal either *len*, or the length of the largest contiguous block of the memory object that is currently mapped to the calling process starting at *addr*, whichever is smaller.

If the memory object mapped at *addr* is a typed memory object, then if the *off* and *contig\_len* values obtained by calling *posix\_mem\_offset()* are used in a call to *mmap()* with a file descriptor that refers to the same memory pool as *fildes* (either through the same port or through a different port), and that was opened with neither the `POSIX_TYPED_MEM_ALLOCATE` nor the `POSIX_TYPED_MEM_ALLOCATE_CONTIG` flag, the typed memory area that is mapped shall be exactly the same area that was mapped at *addr* in the address space of the process that called *posix\_mem\_offset()*.

If the memory object specified by *fildes* is not a typed memory object, then the behavior of this function is implementation-defined.

**RETURN VALUE**

Upon successful completion, the *posix\_mem\_offset()* function shall return zero; otherwise, the corresponding error status value shall be returned.

**ERRORS**

The *posix\_mem\_offset()* function shall fail if:

[EACCES] The process has not mapped a memory object supported by this function at the given address *addr*.

This function shall not return an error code of [EINTR].

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

None.

47837 **FUTURE DIRECTIONS**

47838 None.

47839 **SEE ALSO**47840 *mmap()*, *posix\_typed\_mem\_open()*

47841 XBD &lt;sys/mman.h&gt;

47842 **CHANGE HISTORY**

47843 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

47844 **NAME**

47845       posix\_memalign — aligned memory allocation (**ADVANCED REALTIME**)

47846 **SYNOPSIS**

47847 ADV       #include <stdlib.h>  
47848       int posix\_memalign(void \*\*memptr, size\_t alignment, size\_t size);

47849 **DESCRIPTION**

47850       The *posix\_memalign()* function shall allocate *size* bytes aligned on a boundary specified by  
47851       *alignment*, and shall return a pointer to the allocated memory in *memptr*. The value of *alignment*  
47852       shall be a power of two multiple of *sizeof(void \*)*.

47853       Upon successful completion, the value pointed to by *memptr* shall be a multiple of *alignment*.

47854       If the size of the space requested is 0, the behavior is implementation-defined: either a null  
47855       pointer shall be returned in *memptr*, or the behavior shall be as if the size were some non-zero  
47856       value, except that the behavior is undefined if the the value returned in *memptr* is used to access  
47857       an object.

47858 CX       The *free()* function shall deallocate memory that has previously been allocated by  
47859       *posix\_memalign()*.

47860 **RETURN VALUE**

47861       Upon successful completion, *posix\_memalign()* shall return zero; otherwise, an error number  
47862       shall be returned to indicate the error and the contents of *memptr* shall either be left unmodified  
47863       or be set to a null pointer.

47864       If *size* is 0, either:

- 47865       • *posix\_memalign()* shall not attempt to allocate any space, in which case either an  
47866       implementation-defined error number shall be returned, or zero shall be returned with a  
47867       null pointer returned in *memptr*, or
- 47868       • *posix\_memalign()* shall attempt to allocate some space and, if the allocation succeeds, zero  
47869       shall be returned and a pointer to the allocated space shall be returned in *memptr*. The  
47870       application shall ensure that the pointer is not used to access an object.

47871 **ERRORS**

47872       The *posix\_memalign()* function shall fail if:

47873       [EINVAL]       The value of the alignment parameter is not a power of two multiple of  
47874       *sizeof(void \*)*.

47875       [ENOMEM]       There is insufficient memory available with the requested alignment.

47876 **EXAMPLES**

47877       The following example shows how applications can obtain consistent behavior on error by  
47878       setting *\*memptr* to be a null pointer before calling *posix\_memalign()*.

```
47879 void *ptr = NULL;
47880 ...
47881 //do some work, which might goto error
47882 if (posix_memalign(&ptr, align, size))
47883     goto error;
47884
47885 //do some more work, which might goto error
47886 ...
47887 error:
```

```
47887         free(ptr);
47888         //more cleanup;
```

#### 47889 APPLICATION USAGE

47890 The *posix\_memalign()* function is part of the Advisory Information option and need not be  
47891 provided on all implementations.

#### 47892 RATIONALE

47893 None.

#### 47894 FUTURE DIRECTIONS

47895 None.

#### 47896 SEE ALSO

47897 *free()*, *malloc()*

47898 XBD <stdlib.h>

#### 47899 CHANGE HISTORY

47900 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

47901 In the SYNOPSIS, the inclusion of <sys/types.h> is no longer required.

#### 47902 Issue 7

47903 Austin Group Interpretation 1003.1-2001 #058 is applied, clarifying the value of the *alignment*  
47904 argument in the DESCRIPTION.

47905 Austin Group Interpretation 1003.1-2001 #152 is applied, clarifying the behavior when the size of  
47906 the space requested is 0.

## NAME

`posix_openpt` — open a pseudo-terminal device

## SYNOPSIS

```
XSI      #include <stdlib.h>
         #include <fcntl.h>

         int posix_openpt(int oflag);
```

## DESCRIPTION

The `posix_openpt()` function shall establish a connection between a master device for a pseudo-terminal and a file descriptor. The file descriptor shall be allocated as described in [Section 2.14](#) (on page 549) and can be used by other I/O functions that refer to that pseudo-terminal.

The file status flags and file access modes of the open file description shall be set according to the value of *oflag*.

Values for *oflag* are constructed by a bitwise-inclusive OR of flags from the following list, defined in `<fcntl.h>`:

|        |                               |
|--------|-------------------------------|
| O_RDWR | Open for reading and writing. |
|--------|-------------------------------|

|          |                                                                                                                      |
|----------|----------------------------------------------------------------------------------------------------------------------|
| O_NOCTTY | If set <i>posix_openpt()</i> shall not cause the terminal device to become the controlling terminal for the process. |
|----------|----------------------------------------------------------------------------------------------------------------------|

The behavior of other values for the *oflag* argument is unspecified.

## RETURN VALUE

Upon successful completion, the `posix_openpt()` function shall open a file descriptor for a master pseudo-terminal device and return a non-negative integer representing the file descriptor. Otherwise, `-1` shall be returned and `errno` set to indicate the error.

## ERRORS

The *posix\_openpt()* function shall fail if:

[EMFILE] All file descriptors available to the process are currently open.

[ENFILE] The maximum allowable number of files is currently open in the system.

The `posix_openpt()` function may fail if:

[EINVAL] The value of *oflag* is not valid.

[EAGAIN] Out of pseudo-terminal resources.

|    |     |         |                           |
|----|-----|---------|---------------------------|
| OB | XSR | [ENOSR] | Out of STREAMS resources. |
|----|-----|---------|---------------------------|

## EXAMPLES

## Opening a Pseudo-Terminal and Returning the Name of the Slave Device and a File Descriptor

```
#include <fcntl.h>
#include <stdio.h>

int masterfd, slavefd;
char *slavedevice;

masterfd = posix_openpt(O_RDWR|O_NOCTTY);

if (masterfd == -1
    || grantpt (masterfd) == -1
```

```
47947         || unlockpt (masterfd) == -1
47948         || (slavedevice = ptsname (masterfd)) == NULL)
47949         return -1;

47950     printf("slave device is: %s\n", slavedevice);

47951     slavefd = open(slavedevice, O_RDWR|O_NOCTTY);
47952     if (slavefd < 0)
47953         return -1;
```

#### 47954 APPLICATION USAGE

47955 This function is a method for portably obtaining a file descriptor of a master terminal device for  
47956 a pseudo-terminal. The *grantpt()* and *ptsname()* functions can be used to manipulate mode and  
47957 ownership permissions, and to obtain the name of the slave device, respectively.

#### 47958 RATIONALE

47959 The standard developers considered the matter of adding a special device for cloning master  
47960 pseudo-terminals: the */dev/ptmx* device. However, consensus could not be reached, and it was  
47961 felt that adding a new function would permit other implementations. The *posix\_openpt()*  
47962 function is designed to complement the *grantpt()*, *ptsname()*, and *unlockpt()* functions.

47963 On implementations supporting the */dev/ptmx* clone device, opening the master device of a  
47964 pseudo-terminal is simply:

```
47965     mfdp = open("/dev/ptmx", oflag );
47966     if (mfdp < 0)
47967         return -1;
```

#### 47968 FUTURE DIRECTIONS

47969 None.

#### 47970 SEE ALSO

47971 [Section 2.14](#) (on page 549), *grantpt()*, *open()*, *ptsname()*, *unlockpt()*

47972 XBD [<fcntl.h>](#), [<stdlib.h>](#)

#### 47973 CHANGE HISTORY

47974 First released in Issue 6.

#### 47975 Issue 7

47976 SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

47977 SD5-XSH-ERN-51 is applied, correcting an error in the EXAMPLES section.

## NAME

`posix_spawn`, `posix_spawnnp` — spawn a process (**ADVANCED REALTIME**)

## SYNOPSIS

```
SPN    #include <spawn.h>
```

```
int posix_spawn(pid_t *restrict pid, const char *restrict path,
    const posix_spawn_file_actions_t *file_actions,
    const posix_spawnattr_t *restrict attrp,
    char *const argv[restrict], char *const envp[restrict]);
int posix_spawnp(pid_t *restrict pid, const char *restrict file,
    const posix_spawn_file_actions_t *file_actions,
    const posix_spawnattr_t *restrict attrp,
    char *const argv[restrict], char *const envp[restrict]);
```

## DESCRIPTION

The *posix\_spawn()* and *posix\_spawnnp()* functions shall create a new process (child process) from the specified process image. The new process image shall be constructed from a regular executable file called the new process image file.

When a C program is executed as the result of this call, it shall be entered as a C-language function call as follows:

```
int main(int argc, char *argv[]);
```

where *argc* is the argument count and *argv* is an array of character pointers to the arguments themselves. In addition, the following variable:

```
extern char **environ;
```

shall be initialized as a pointer to an array of character pointers to the environment strings.

The argument *argv* is an array of character pointers to null-terminated strings. The last member of this array shall be a null pointer and is not counted in *argc*. These strings constitute the argument list available to the new process image. The value in *argv*[0] should point to a filename string that is associated with the process image being started by the *posix\_spawn()* or *posix\_spawnnp()* function.

The argument *envp* is an array of character pointers to null-terminated strings. These strings constitute the environment for the new process image. The environment array is terminated by a null pointer.

The number of bytes available for the combined argument and environment lists of the child process is {ARG\_MAX}. The implementation shall specify in the system documentation (see XBD [Chapter 2](#), on page 15) whether any list overhead, such as length words, null terminators, pointers, or alignment bytes, is included in this total.

The *path* argument to *posix\_spawn()* is a pathname that identifies the new process image file to execute.

The *file* parameter to *posix\_spawnnp()* shall be used to construct a pathname that identifies the new process image file. If the *file* parameter contains a <slash> character, the *file* parameter shall be used as the pathname for the new process image file. Otherwise, the path prefix for this file shall be obtained by a search of the directories passed as the environment variable *PATH* (see XBD Chapter 8, on page 173). If this environment variable is not defined, the results of the search are implementation-defined.

If *file\_actions* is a null pointer, then file descriptors open in the calling process shall remain open



in the child process, except for those whose close-on-exec flag FD\_CLOEXEC is set (see *fcntl()*). For those file descriptors that remain open, the child process shall not inherit any file locks, but all remaining attributes of the corresponding open file descriptions (see *fcntl()*), shall remain unchanged.

If *file\_actions* is not NULL, then the file descriptors open in the child process shall be those open in the calling process as modified by the spawn file actions object pointed to by *file\_actions* and the FD\_CLOEXEC flag of each remaining open file descriptor after the spawn file actions have been processed. The effective order of processing the spawn file actions shall be:

1. The set of open file descriptors for the child process shall initially be the same set as is open for the calling process. The child process shall not inherit any file locks, but all remaining attributes of the corresponding open file descriptions (see *fcntl()*), shall remain unchanged.
2. The signal mask, signal default actions, and the effective user and group IDs for the child process shall be changed as specified in the attributes object referenced by *attrp*.
3. The file actions specified by the spawn file actions object shall be performed in the order in which they were added to the spawn file actions object.
4. Any file descriptor that has its FD\_CLOEXEC flag set (see *fcntl()*) shall be closed.

If file descriptor 0, 1, or 2 would otherwise be closed in the new process image created by *posix\_spawn()* or *posix\_spawnnp()*, implementations may open an unspecified file for the file descriptor in the new process image. If a standard utility or a conforming application is executed with file descriptor 0 not open for reading or with file descriptor 1 or 2 not open for writing, the environment in which the utility or application is executed shall be deemed non-conforming, and consequently the utility or application might not behave as described in this standard.

The **posix\_spawnattr\_t** spawn attributes object type is defined in **<spawn.h>**. It shall contain at least the attributes defined below.

If the POSIX\_SPAWN\_SETPGROUP flag is set in the *spawn-flags* attribute of the object referenced by *attrp*, and the *spawn-pgroup* attribute of the same object is non-zero, then the child's process group shall be as specified in the *spawn-pgroup* attribute of the object referenced by *attrp*.

As a special case, if the POSIX\_SPAWN\_SETPGROUP flag is set in the *spawn-flags* attribute of the object referenced by *attrp*, and the *spawn-pgroup* attribute of the same object is set to zero, then the child shall be in a new process group with a process group ID equal to its process ID.

If the POSIX\_SPAWN\_SETPGROUP flag is not set in the *spawn-flags* attribute of the object referenced by *attrp*, the new child process shall inherit the parent's process group.

PS If the POSIX\_SPAWN\_SETSCHEDPARAM flag is set in the *spawn-flags* attribute of the object referenced by *attrp*, but POSIX\_SPAWN\_SETSCHEDULER is not set, the new process image shall initially have the scheduling policy of the calling process with the scheduling parameters specified in the *spawn-schedparam* attribute of the object referenced by *attrp*.

If the POSIX\_SPAWN\_SETSCHEDULER flag is set in the *spawn-flags* attribute of the object referenced by *attrp* (regardless of the setting of the POSIX\_SPAWN\_SETSCHEDPARAM flag), the new process image shall initially have the scheduling policy specified in the *spawn-schedpolicy* attribute of the object referenced by *attrp* and the scheduling parameters specified in the *spawn-schedparam* attribute of the same object.

The POSIX\_SPAWN\_RESETPID flag in the *spawn-flags* attribute of the object referenced by *attrp* governs the effective user ID of the child process. If this flag is not set, the child process shall inherit the effective user ID of the parent process. If this flag is set, the effective user ID of the

48067  
48068  
48069

48070  
48071  
48072  
48073  
48074  
48075

48076  
48077  
48078

48079  
48080  
48081  
48082

48083  
48084

48085  
48086  
48087  
48088

48089  
48090  
48091  
48092  
48093

48094

48095  
48096  
48097  
48098  
48099

48100  
48101

## 48102

48103  
48104  
48105  
48106  
48107  
48108

## 48109

48110

[EINVAL] The value specified by *file\_actions* or *attrp* is invalid.

If this error occurs after the calling process successfully returns from the *posix\_spawn()* or *posix\_spawnnp()* function, the child process may exit with exit status 127.

If *posix\_spawn()* or *posix\_spawnnp()* fail for any of the reasons that would cause *fork()* or one of the *exec* family of functions to fail, an error value shall be returned as described by *fork()* and *exec*, respectively (or, if the error occurs after the calling process successfully returns, the child process shall exit with exit status 127).

If POSIX\_SPAWN\_SETPGROUP is set in the *spawn\_flags* attribute of the object referenced by *attrp*, and *posix\_spawn()* or *posix\_spawnnp()* fails while changing the child's process group, an error value shall be returned as described by *setpgid()* (or, if the error occurs after the calling process successfully returns, the child process shall exit with exit status 127).

PS If POSIX\_SPAWN\_SETSCHEDPARAM is set and POSIX\_SPAWN\_SETSCHEDULER is not set in the *spawn\_flags* attribute of the object referenced by *attrp*, then if *posix\_spawn()* or *posix\_spawnnp()* fails for any of the reasons that would cause *sched\_setparam()* to fail, an error value shall be returned as described by *sched\_setparam()* (or, if the error occurs after the calling process successfully returns, the child process shall exit with exit status 127).

If POSIX\_SPAWN\_SETSCHEDULER is set in the *spawn\_flags* attribute of the object referenced by *attrp*, and if *posix\_spawn()* or *posix\_spawnnp()* fails for any of the reasons that would cause *sched\_setscheduler()* to fail, an error value shall be returned as described by *sched\_setscheduler()* (or, if the error occurs after the calling process successfully returns, the child process shall exit with exit status 127).

If the *file\_actions* argument is not NULL, and specifies any *close*, *dup2*, or *open* actions to be performed, and if *posix\_spawn()* or *posix\_spawnnp()* fails for any of the reasons that would cause *close()*, *dup2()*, or *open()* to fail, an error value shall be returned as described by *close()*, *dup2()*, and *open()*, respectively (or, if the error occurs after the calling process successfully returns, the child process shall exit with exit status 127). An open file action may, by itself, result in any of the errors described by *close()* or *dup2()*, in addition to those described by *open()*.

## EXAMPLES

None.

## APPLICATION USAGE

These functions are part of the Spawn option and need not be provided on all implementations.

See also the APPLICATION USAGE section for *exec*.

## RATIONALE

The *posix\_spawn()* function and its close relation *posix\_spawnnp()* have been introduced to overcome the following perceived difficulties with *fork()*: the *fork()* function is difficult or impossible to implement without swapping or dynamic address translation.

- Swapping is generally too slow for a realtime environment.
- Dynamic address translation is not available everywhere that POSIX might be useful.
- Processes are too useful to simply option out of POSIX whenever it must run without address translation or other MMU services.

Thus, POSIX needs process creation and file execution primitives that can be efficiently implemented without address translation or other MMU services.

The *posix\_spawn()* function is implementable as a library routine, but both *posix\_spawn()* and *posix\_spawnnp()* are designed as kernel operations. Also, although they may be an efficient

replacement for many *fork()/exec* pairs, their goal is to provide useful process creation primitives for systems that have difficulty with *fork()*, not to provide drop-in replacements for *fork()/exec*.

This view of the role of *posix\_spawn()* and *posix\_spawnnp()* influenced the design of their API. It does not attempt to provide the full functionality of *fork()/exec* in which arbitrary user-specified operations of any sort are permitted between the creation of the child process and the execution of the new process image; any attempt to reach that level would need to provide a programming language as parameters. Instead, *posix\_spawn()* and *posix\_spawnnp()* are process creation primitives like the *Start\_Process* and *Start\_Process\_Search* Ada language bindings package *POSIX\_Process\_Primitives* and also like those in many operating systems that are not UNIX systems, but with some POSIX-specific additions.

To achieve its coverage goals, *posix\_spawn()* and *posix\_spawnnp()* have control of six types of inheritance: file descriptors, process group ID, user and group ID, signal mask, scheduling, and whether each signal ignored in the parent will remain ignored in the child, or be reset to its default action in the child.

Control of file descriptors is required to allow an independently written child process image to access data streams opened by and even generated or read by the parent process without being specifically coded to know which parent files and file descriptors are to be used. Control of the process group ID is required to control how the job control of the child process relates to that of the parent.

Control of the signal mask and signal defaulting is sufficient to support the implementation of *system()*. Although support for *system()* is not explicitly one of the goals for *posix\_spawn()* and *posix\_spawnnp()*, it is covered under the “at least 50%” coverage goal.

The intention is that the normal file descriptor inheritance across *fork()*, the subsequent effect of the specified spawn file actions, and the normal file descriptor inheritance across one of the *exec* family of functions should fully specify open file inheritance. The implementation need make no decisions regarding the set of open file descriptors when the child process image begins execution, those decisions having already been made by the caller and expressed as the set of open file descriptors and their *FD\_CLOEXEC* flags at the time of the call and the spawn file actions object specified in the call. We have been assured that in cases where the POSIX *Start\_Process* Ada primitives have been implemented in a library, this method of controlling file descriptor inheritance may be implemented very easily.

We can identify several problems with *posix\_spawn()* and *posix\_spawnnp()*, but there does not appear to be a solution that introduces fewer problems. Environment modification for child process attributes not specifiable via the *attrp* or *file\_actions* arguments must be done in the parent process, and since the parent generally wants to save its context, it is more costly than similar functionality with *fork()/exec*. It is also complicated to modify the environment of a multi-threaded process temporarily, since all threads must agree when it is safe for the environment to be changed. However, this cost is only borne by those invocations of *posix\_spawn()* and *posix\_spawnnp()* that use the additional functionality. Since extensive modifications are not the usual case, and are particularly unlikely in time-critical code, keeping much of the environment control out of *posix\_spawn()* and *posix\_spawnnp()* is appropriate design.

The *posix\_spawn()* and *posix\_spawnnp()* functions do not have all the power of *fork()/exec*. This is to be expected. The *fork()* function is a wonderfully powerful operation. We do not expect to duplicate its functionality in a simple, fast function with no special hardware requirements. It is worth noting that *posix\_spawn()* and *posix\_spawnnp()* are very similar to the process creation operations on many operating systems that are not UNIX systems.

## Requirements

The requirements for *posix\_spawn()* and *posix\_spawnnp()* are:

- They must be implementable without an MMU or unusual hardware.
- They must be compatible with existing POSIX standards.

Additional goals are:

- They should be efficiently implementable.
- They should be able to replace at least 50% of typical executions of *fork()*.
- A system with *posix\_spawn()* and *posix\_spawnnp()* and without *fork()* should be useful, at least for realtime applications.
- A system with *fork()* and the *exec* family should be able to implement *posix\_spawn()* and *posix\_spawnnp()* as library routines.

## Two-Syntax

POSIX *exec* has several calling sequences with approximately the same functionality. These appear to be required for compatibility with existing practice. Since the existing practice for the *posix\_spawn\*()* functions is otherwise substantially unlike POSIX, we feel that simplicity outweighs compatibility. There are, therefore, only two names for the *posix\_spawn\*()* functions.

The parameter list does not differ between *posix\_spawn()* and *posix\_spawnnp()*; *posix\_spawnnp()* interprets the second parameter more elaborately than *posix\_spawn()*.

## Compatibility with POSIX.5 (Ada)

The *Start\_Process* and *Start\_Process\_Search* procedures from the *POSIX\_Process\_Primitives* package from the Ada language binding to POSIX.1 encapsulate *fork()* and *exec* functionality in a manner similar to that of *posix\_spawn()* and *posix\_spawnnp()*. Originally, in keeping with our simplicity goal, the standard developers had limited the capabilities of *posix\_spawn()* and *posix\_spawnnp()* to a subset of the capabilities of *Start\_Process* and *Start\_Process\_Search*; certain non-default capabilities were not supported. However, based on suggestions by the ballot group to improve file descriptor mapping or drop it, and on the advice of an Ada Language Bindings working group member, the standard developers decided that *posix\_spawn()* and *posix\_spawnnp()* should be sufficiently powerful to implement *Start\_Process* and *Start\_Process\_Search*. The rationale is that if the Ada language binding to such a primitive had already been approved as an IEEE standard, there can be little justification for not approving the functionally-equivalent parts of a C binding. The only three capabilities provided by *posix\_spawn()* and *posix\_spawnnp()* that are not provided by *Start\_Process* and *Start\_Process\_Search* are optionally specifying the child's process group ID, the set of signals to be reset to default signal handling in the child process, and the child's scheduling policy and parameters.

For the Ada language binding for *Start\_Process* to be implemented with *posix\_spawn()*, that binding would need to explicitly pass an empty signal mask and the parent's environment to *posix\_spawn()* whenever the caller of *Start\_Process* allowed these arguments to default, since *posix\_spawn()* does not provide such defaults. The ability of *Start\_Process* to mask user-specified signals during its execution is functionally unique to the Ada language binding and must be dealt with in the binding separately from the call to *posix\_spawn()*.



## 48242

48243  
48244  
48245

## 48246

48247  
48248  
48249  
48250  
48251  
48252  
48253  
48254  
48255

48256  
48257  
48258  
48259  
48260  
48261

## 48262

48263  
48264  
48265  
48266  
48267

48268  
48269  
48270  
48271  
48272

48273  
48274  
48275  
48276  
48277  
48278  
48279  
48280

48281  
48282  
48283  
48284  
48285  
48286  
48287

identify this class of error, nor does it provide any detailed information on the nature of the failure. Note that a kernel implementation of *posix\_spawn()* or *posix\_spawnnp()* is permitted (and encouraged) to return any possible error as the function value, thus providing more detailed failure information to the parent process.

Thus, no special macros are available to isolate asynchronous *posix\_spawn()* or *posix\_spawnnp()* errors. Instead, errors detected by the *posix\_spawn()* or *posix\_spawnnp()* operations in the context of the child process before the new process image executes are reported by setting the child's exit status to 127. The calling process may use the WIFEXITED and WEXITSTATUS macros on the *stat\_val* stored by the *wait()* or *waitpid()* functions to detect spawn failures to the extent that other status values with which the child process image may exit (before the parent can conclusively determine that the child process image has begun execution) are distinct from exit status 127.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*alarm()*, *chmod()*, *close()*, *dup()*, *exec*, *exit()*, *fcntl()*, *fork()*, *fstatat()*, *kill()*, *open()*, *posix\_spawn\_file\_actions\_addclose()*, *posix\_spawn\_file\_actions\_adddup2()*, *posix\_spawn\_file\_actions\_destroy()*, *posix\_spawnattr\_destroy()*, *posix\_spawnattr\_getsigdefault()*, *posix\_spawnattr\_getflags()*, *posix\_spawnattr\_getpgroup()*, *posix\_spawnattr\_getschedparam()*, *posix\_spawnattr\_getschedpolicy()*, *posix\_spawnattr\_getsigmask()*, *sched\_setparam()*, *sched\_setscheduler()*, *setpgid()*, *setuid()*, *times()*, *wait()*, *waitid()*

XBD Chapter 8 (on page 173), [`<spawn.h>`](#)

#### CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

IEEE PASC Interpretation 1003.1 #103 is applied, noting that the signal default actions are changed as well as the signal mask in step 2.

IEEE PASC Interpretation 1003.1 #132 is applied.

#### Issue 7

Functionality relating to the Threads option is moved to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0433 [291], XSH/TC1-2008/0434 [173], and XSH/TC1-2008/0435 [173] are applied.

## NAME

posix\_spawn\_file\_actions\_addclose, posix\_spawn\_file\_actions\_addopen — add close or open action to spawn file actions object (**ADVANCED REALTIME**)

## SYNOPSIS

```
#include <spawn.h>

int posix_spawn_file_actions_addclose(posix_spawn_file_actions_t
    *file_actions, int fildes);
int posix_spawn_file_actions_addopen(posix_spawn_file_actions_t
    *restrict file_actions, int fildes,
    const char *restrict path, int oflag, mode_t mode);
```

## DESCRIPTION

These functions shall add or delete a close or open action to a spawn file actions object.

A spawn file actions object is of type **posix\_spawn\_file\_actions\_t** (defined in **<spawn.h>**) and is used to specify a series of actions to be performed by a *posix\_spawn()* or *posix\_spawnnp()* operation in order to arrive at the set of open file descriptors for the child process given the set of open file descriptors of the parent. POSIX.1-2008 does not define comparison or assignment operators for the type **posix\_spawn\_file\_actions\_t**.

A spawn file actions object, when passed to *posix\_spawn()* or *posix\_spawnnp()*, shall specify how the set of open file descriptors in the calling process is transformed into a set of potentially open file descriptors for the spawned process. This transformation shall be as if the specified sequence of actions was performed exactly once, in the context of the spawned process (prior to execution of the new process image), in the order in which the actions were added to the object; additionally, when the new process image is executed, any file descriptor (from this new set) which has its FD\_CLOEXEC flag set shall be closed (see *posix\_spawn()*).

The *posix\_spawn\_file\_actions\_addclose()* function shall add a *close* action to the object referenced by *file\_actions* that shall cause the file descriptor *fildes* to be closed (as if *close(fildes)* had been called) when a new process is spawned using this file actions object.

The *posix\_spawn\_file\_actions\_addopen()* function shall add an *open* action to the object referenced by *file\_actions* that shall cause the file named by *path* to be opened (as if *open(path, oflag, mode)* had been called, and the returned file descriptor, if not *fildes*, had been changed to *fildes*) when a new process is spawned using this file actions object. If *fildes* was already an open file descriptor, it shall be closed before the new file is opened.

The string described by *path* shall be copied by the *posix\_spawn\_file\_actions\_addopen()* function.

## RETURN VALUE

Upon successful completion, these functions shall return zero; otherwise, an error number shall be returned to indicate the error.

## ERRORS

The *posix\_spawn\_file\_actions\_addopen()* function shall fail if:

[EBADF]        The value specified by *fildes* is negative or greater than or equal to {OPEN\_MAX}.

The *posix\_spawn\_file\_actions\_addclose()* function shall fail if:

[EBADF]        The value specified by *fildes* is negative.



48361 These functions may fail if:

48362 [EINVAL] The value specified by *file\_actions* is invalid.

48363 [ENOMEM] Insufficient memory exists to add to the spawn file actions object.

48364 It shall not be considered an error for the *fildev* argument passed to these functions to specify a  
 48365 file descriptor for which the specified operation could not be performed at the time of the call.  
 48366 Any such error will be detected when the associated file actions object is later used during a  
 48367 *posix\_spawn()* or *posix\_spawnnp()* operation.

#### 48368 EXAMPLES

48369 None.

#### 48370 APPLICATION USAGE

48371 These functions are part of the Spawn option and need not be provided on all implementations.

48372 Implementations may use file descriptors that must be inherited into child processes for the  
 48373 child process to remain conforming, such as for message catalog or tracing purposes. Therefore,  
 48374 an application that calls *posix\_spawn\_file\_actions\_addclose()* with an arbitrary integer risks non-  
 48375 conforming behavior, and this function can only portably be used to close file descriptor values  
 48376 that the application has obtained through explicit actions, or for the three file descriptors  
 48377 corresponding to the standard file streams. In order to avoid a race condition of leaking an  
 48378 unintended file descriptor into a child process, an application should consider opening all file  
 48379 descriptors with the FD\_CLOEXEC bit set unless the file descriptor is intended to be inherited  
 48380 across *exec*.

#### 48381 RATIONALE

48382 A spawn file actions object may be initialized to contain an ordered sequence of *close()*, *dup2()*,  
 48383 and *open()* operations to be used by *posix\_spawn()* or *posix\_spawnnp()* to arrive at the set of open  
 48384 file descriptors inherited by the spawned process from the set of open file descriptors in the  
 48385 parent at the time of the *posix\_spawn()* or *posix\_spawnnp()* call. It had been suggested that the  
 48386 *close()* and *dup2()* operations alone are sufficient to rearrange file descriptors, and that files  
 48387 which need to be opened for use by the spawned process can be handled either by having the  
 48388 calling process open them before the *posix\_spawn()* or *posix\_spawnnp()* call (and close them after),  
 48389 or by passing pathnames to the spawned process (in *argv*) so that it may open them itself. The  
 48390 standard developers recommend that applications use one of these two methods when practical,  
 48391 since detailed error status on a failed open operation is always available to the application this  
 48392 way. However, the standard developers feel that allowing a spawn file actions object to specify  
 48393 open operations is still appropriate because:

- 48394 1. It is consistent with equivalent POSIX.5 (Ada) functionality.
- 48395 2. It supports the I/O redirection paradigm commonly employed by POSIX programs  
 48396 designed to be invoked from a shell. When such a program is the child process, it may not  
 48397 be designed to open files on its own.
- 48398 3. It allows file opens that might otherwise fail or violate file ownership/access rights if  
 48399 executed by the parent process.

48400 Regarding 2. above, note that the spawn open file action provides to *posix\_spawn()* and  
 48401 *posix\_spawnnp()* the same capability that the shell redirection operators provide to *system()*, only  
 48402 without the intervening execution of a shell; for example:

```
48403 system ("myprog <file1 3<file2");
```

48404 Regarding 3. above, note that if the calling process needs to open one or more files for access by  
 48405 the spawned process, but has insufficient spare file descriptors, then the open action is necessary

to allow the *open()* to occur in the context of the child process after other file descriptors have been closed (that must remain open in the parent).

Additionally, if a parent is executed from a file having a “set-user-id” mode bit set and the POSIX\_SPAWN\_RESETPID flag is set in the spawn attributes, a file created within the parent process will (possibly incorrectly) have the parent’s effective user ID as its owner, whereas a file created via an *open()* action during *posix\_spawn()* or *posix\_spawnnp()* will have the parent’s real ID as its owner; and an open by the parent process may successfully open a file to which the real user should not have access or fail to open a file to which the real user should have access.

### File Descriptor Mapping

The standard developers had originally proposed using an array which specified the mapping of child file descriptors back to those of the parent. It was pointed out by the ballot group that it is not possible to reshuffle file descriptors arbitrarily in a library implementation of *posix\_spawn()* or *posix\_spawnnp()* without provision for one or more spare file descriptor entries (which simply may not be available). Such an array requires that an implementation develop a complex strategy to achieve the desired mapping without inadvertently closing the wrong file descriptor at the wrong time.

It was noted by a member of the Ada Language Bindings working group that the approved Ada Language *Start\_Process* family of POSIX process primitives use a caller-specified set of file actions to alter the normal *fork()/exec* semantics for inheritance of file descriptors in a very flexible way, yet no such problems exist because the burden of determining how to achieve the final file descriptor mapping is completely on the application. Furthermore, although the file actions interface appears frightening at first glance, it is actually quite simple to implement in either a library or the kernel.

The *posix\_spawn\_file\_actions\_addclose()* function is not required to check whether the file descriptor is less than {OPEN\_MAX} because on some implementations {OPEN\_MAX} reflects the RLIMIT\_NOFILE soft limit and therefore calling *setrlimit()* to reduce this limit can result in an {OPEN\_MAX} value less than or equal to an already open file descriptor. Applications need to be able to close such file descriptors on spawn. On implementations where {OPEN\_MAX} does not change, it is recommended that *posix\_spawn\_file\_actions\_addclose()* should return [EBADF] if *fd* is greater than or equal to {OPEN\_MAX}.

### FUTURE DIRECTIONS

None.

### SEE ALSO

*close()*, *dup()*, *open()*, *posix\_spawn()*, *posix\_spawn\_file\_actions\_adddup2()*,  
*posix\_spawn\_file\_actions\_destroy()*

XBD <spawn.h>

### CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

IEEE PASC Interpretation 1003.1 #105 is applied, adding a note to the DESCRIPTION that the string pointed to by *path* is copied by the *posix\_spawn\_file\_actions\_addopen()* function.

### Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0436 [418], XSH/TC1-2008/0437 [149], XSH/TC1-2008/0438 [291], and XSH/TC1-2008/0439 [418] are applied.

**NAME**

posix\_spawn\_file\_actions\_adddup2 — add dup2 action to spawn file actions object  
(ADVANCED REALTIME)

**SYNOPSIS**

```
#include <spawn.h>

int posix_spawn_file_actions_adddup2(posix_spawn_file_actions_t
    *file_actions, int fildes, int newfildes);
```

**DESCRIPTION**

The *posix\_spawn\_file\_actions\_adddup2()* function shall add a *dup2()* action to the object referenced by *file\_actions* that shall cause the file descriptor *fildes* to be duplicated as *newfildes* (as if *dup2(fildes, newfildes)* had been called) when a new process is spawned using this file actions object.

A spawn file actions object is as defined in *posix\_spawn\_file\_actions\_addclose()*.

**RETURN VALUE**

Upon successful completion, the *posix\_spawn\_file\_actions\_adddup2()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

**ERRORS**

The *posix\_spawn\_file\_actions\_adddup2()* function shall fail if:

[EBADF] The value specified by *fildes* or *newfildes* is negative or greater than or equal to {OPEN\_MAX}.

[ENOMEM] Insufficient memory exists to add to the spawn file actions object.

The *posix\_spawn\_file\_actions\_adddup2()* function may fail if:

[EINVAL] The value specified by *file\_actions* is invalid.

It shall not be considered an error for the *fildes* argument passed to the *posix\_spawn\_file\_actions\_adddup2()* function to specify a file descriptor for which the specified operation could not be performed at the time of the call. Any such error will be detected when the associated file actions object is later used during a *posix\_spawn()* or *posix\_spawnnp()* operation.

**EXAMPLES**

None.

**APPLICATION USAGE**

The *posix\_spawn\_file\_actions\_adddup2()* function is part of the Spawn option and need not be provided on all implementations.

Implementations may use file descriptors that must be inherited into child processes for the child process to remain conforming, such as for message catalog or tracing purposes. Therefore, an application that calls *posix\_spawn\_file\_actions\_adddup2()* with an arbitrary integer for *newfildes* risks non-conforming behavior, and this function can only portably be used to overwrite file descriptor values that the application has obtained through explicit actions, or for the three file descriptors corresponding to the standard file streams. In order to avoid a race condition of leaking an unintended file descriptor into a child process, an application should consider opening all file descriptors with the FD\_CLOEXEC bit set unless the file descriptor is intended to be inherited across *exec*.

48491 **RATIONALE**48492 Refer to the RATIONALE section in *posix\_spawn\_file\_actions\_addclose()*.48493 **FUTURE DIRECTIONS**

48494 None.

48495 **SEE ALSO**48496 *dup()*, *posix\_spawn()*, *posix\_spawn\_file\_actions\_addclose()*, *posix\_spawn\_file\_actions\_destroy()*

48497 XBD &lt;spawn.h&gt;

48498 **CHANGE HISTORY**

48499 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

48500 IEEE PASC Interpretation 1003.1 #104 is applied, noting that the [EBADF] error can apply to the  
48501 *newfildes* argument in addition to *fildes*.48502 **Issue 7**

48503 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0440 [149] is applied.

48504 **NAME**

48505       posix\_spawn\_file\_actions\_addopen — add open action to spawn file actions object  
48506       (ADVANCED REALTIME)

48507 **SYNOPSIS**

```
48508 SPN   #include <spawn.h>  
48509       int posix_spawn_file_actions_addopen(posix_spawn_file_actions_t  
48510       *restrict file_actions, int fildes,  
48511       const char *restrict path, int oflag, mode_t mode);
```

48512 **DESCRIPTION**

48513       Refer to *posix\_spawn\_file\_actions\_addclose()*.

## NAME

posix\_spawn\_file\_actions\_destroy, posix\_spawn\_file\_actions\_init — destroy and initialize spawn file actions object (ADVANCED REALTIME)

## SYNOPSIS

```
SPN    #include <spawn.h>

int posix_spawn_file_actions_destroy(posix_spawn_file_actions_t
    *file_actions);
int posix_spawn_file_actions_init(posix_spawn_file_actions_t
    *file_actions);
```

## DESCRIPTION

The *posix\_spawn\_file\_actions\_destroy()* function shall destroy the object referenced by *file\_actions*; the object becomes, in effect, uninitialized. An implementation may cause *posix\_spawn\_file\_actions\_destroy()* to set the object referenced by *file\_actions* to an invalid value. A destroyed spawn file actions object can be reinitialized using *posix\_spawn\_file\_actions\_init()*; the results of otherwise referencing the object after it has been destroyed are undefined.

The *posix\_spawn\_file\_actions\_init()* function shall initialize the object referenced by *file\_actions* to contain no file actions for *posix\_spawn()* or *posix\_spawnnp()* to perform.

A spawn file actions object is as defined in *posix\_spawn\_file\_actions\_addclose()*.

The effect of initializing an already initialized spawn file actions object is undefined.

## RETURN VALUE

Upon successful completion, these functions shall return zero; otherwise, an error number shall be returned to indicate the error.

## ERRORS

The *posix\_spawn\_file\_actions\_init()* function shall fail if:

[ENOMEM]      Insufficient memory exists to initialize the spawn file actions object.

The *posix\_spawn\_file\_actions\_destroy()* function may fail if:

[EINVAL]      The value specified by *file\_actions* is invalid.

## EXAMPLES

None.

## APPLICATION USAGE

These functions are part of the Spawn option and need not be provided on all implementations.

## RATIONALE

Refer to the RATIONALE section in *posix\_spawn\_file\_actions\_addclose()*.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*posix\_spawn()*, *posix\_spawn\_file\_actions\_addclose()*

XBD <spawn.h>

**CHANGE HISTORY**

- 48552 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.
- 48553
- 48554 In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

## NAME

posix\_spawnattr\_destroy, posix\_spawnattr\_init — destroy and initialize spawn attributes object (ADVANCED REALTIME)

## SYNOPSIS

```
#include <spawn.h>

int posix_spawnattr_destroy(posix_spawnattr_t *attr);
int posix_spawnattr_init(posix_spawnattr_t *attr);
```

## DESCRIPTION

The *posix\_spawnattr\_destroy()* function shall destroy a spawn attributes object. A destroyed *attr* attributes object can be reinitialized using *posix\_spawnattr\_init()*; the results of otherwise referencing the object after it has been destroyed are undefined. An implementation may cause *posix\_spawnattr\_destroy()* to set the object referenced by *attr* to an invalid value.

The *posix\_spawnattr\_init()* function shall initialize a spawn attributes object *attr* with the default value for all of the individual attributes used by the implementation. Results are undefined if *posix\_spawnattr\_init()* is called specifying an already initialized *attr* attributes object.

A spawn attributes object is of type **posix\_spawnattr\_t** (defined in **<spawn.h>**) and is used to specify the inheritance of process attributes across a spawn operation. POSIX.1-2008 does not define comparison or assignment operators for the type **posix\_spawnattr\_t**.

Each implementation shall document the individual attributes it uses and their default values unless these values are defined by POSIX.1-2008. Attributes not defined by POSIX.1-2008, their default values, and the names of the associated functions to get and set those attribute values are implementation-defined.

The resulting spawn attributes object (possibly modified by setting individual attribute values), is used to modify the behavior of *posix\_spawn()* or *posix\_spawnnp()*. After a spawn attributes object has been used to spawn a process by a call to a *posix\_spawn()* or *posix\_spawnnp()*, any function affecting the attributes object (including destruction) shall not affect any process that has been spawned in this way.

## RETURN VALUE

Upon successful completion, *posix\_spawnattr\_destroy()* and *posix\_spawnattr\_init()* shall return zero; otherwise, an error number shall be returned to indicate the error.

## ERRORS

The *posix\_spawnattr\_init()* function shall fail if:

[ENOMEM]      Insufficient memory exists to initialize the spawn attributes object.

The *posix\_spawnattr\_destroy()* function may fail if:

[EINVAL]      The value specified by *attr* is invalid.

## EXAMPLES

None.

## APPLICATION USAGE

These functions are part of the Spawn option and need not be provided on all implementations.

## RATIONALE

The original spawn interface proposed in POSIX.1-2008 defined the attributes that specify the inheritance of process attributes across a spawn operation as a structure. In order to be able to separate optional individual attributes under their appropriate options (that is, the *spawn-schedparam* and *spawn-schedpolicy* attributes depending upon the Process Scheduling option), and



also for extensibility and consistency with the newer POSIX interfaces, the attributes interface has been changed to an opaque data type. This interface now consists of the type **posix\_spawnattr\_t**, representing a spawn attributes object, together with associated functions to initialize or destroy the attributes object, and to set or get each individual attribute. Although the new object-oriented interface is more verbose than the original structure, it is simple to use, more extensible, and easy to implement.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*posix\_spawn()*, *posix\_spawnattr\_getsigdefault()*, *posix\_spawnattr\_getflags()*,  
*posix\_spawnattr\_getpgroup()*, *posix\_spawnattr\_getschedparam()*, *posix\_spawnattr\_getschedpolicy()*,  
*posix\_spawnattr\_getsigmask()*

XBD <spawn.h>

**CHANGE HISTORY**

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

IEEE PASC Interpretation 1003.1 #106 is applied, noting that the effect of initializing an already initialized spawn attributes option is undefined.

**NAME**

posix\_spawnattr\_getflags, posix\_spawnattr\_setflags — get and set the spawn-flags attribute of a spawn attributes object (**ADVANCED REALTIME**)

**SYNOPSIS**

```
#include <spawn.h>

int posix_spawnattr_getflags(const posix_spawnattr_t *restrict attr,
    short *restrict flags);
int posix_spawnattr_setflags(posix_spawnattr_t *attr, short flags);
```

**DESCRIPTION**

The *posix\_spawnattr\_getflags()* function shall obtain the value of the *spawn-flags* attribute from the attributes object referenced by *attr*.

The *posix\_spawnattr\_setflags()* function shall set the *spawn-flags* attribute in an initialized attributes object referenced by *attr*.

The *spawn-flags* attribute is used to indicate which process attributes are to be changed in the new process image when invoking *posix\_spawn()* or *posix\_spawnnp()*. It is the bitwise-inclusive OR of zero or more of the following flags:

```
POSIX_SPAWN_RESETIDS
POSIX_SPAWN_SETPGROUP
POSIX_SPAWN_SETSIGDEF
POSIX_SPAWN_SETSIGMASK
POSIX_SPAWN_SETSCHEDPARAM
POSIX_SPAWN_SETSCHEDULER
```

These flags are defined in **<spawn.h>**. The default value of this attribute shall be as if no flags were set.

**RETURN VALUE**

Upon successful completion, *posix\_spawnattr\_getflags()* shall return zero and store the value of the *spawn-flags* attribute of *attr* into the object referenced by the *flags* parameter; otherwise, an error number shall be returned to indicate the error.

Upon successful completion, *posix\_spawnattr\_setflags()* shall return zero; otherwise, an error number shall be returned to indicate the error.

**ERRORS**

These functions may fail if:

[EINVAL] The value specified by *attr* is invalid.

The *posix\_spawnattr\_setflags()* function may fail if:

[EINVAL] The value of the attribute being set is not valid.

**48651 EXAMPLES**

48652       None.

**48653 APPLICATION USAGE**

48654       These functions are part of the Spawn option and need not be provided on all implementations.

**48655 RATIONALE**

48656       None.

**48657 FUTURE DIRECTIONS**

48658       None.

**48659 SEE ALSO**

48660       *posix\_spawn()*, *posix\_spawnattr\_destroy()*, *posix\_spawnattr\_getsigdefault()*,  
48661       *posix\_spawnattr\_getpgroup()*, *posix\_spawnattr\_getschedparam()*, *posix\_spawnattr\_getschedpolicy()*,  
48662       *posix\_spawnattr\_getsigmask()*

48663       XBD <spawn.h>

**48664 CHANGE HISTORY**

48665       First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

**NAME**

posix\_spawnattr\_getpgroup, posix\_spawnattr\_setpgroup — get and set the spawn-pgroup attribute of a spawn attributes object (**ADVANCED REALTIME**)

**SYNOPSIS**

```
#include <spawn.h>

int posix_spawnattr_getpgroup(const posix_spawnattr_t *restrict attr,
    pid_t *restrict pgroup);
int posix_spawnattr_setpgroup(posix_spawnattr_t *attr, pid_t pgroup);
```

**DESCRIPTION**

The *posix\_spawnattr\_getpgroup()* function shall obtain the value of the *spawn-pgroup* attribute from the attributes object referenced by *attr*.

The *posix\_spawnattr\_setpgroup()* function shall set the *spawn-pgroup* attribute in an initialized attributes object referenced by *attr*.

The *spawn-pgroup* attribute represents the process group to be joined by the new process image in a spawn operation (if `POSIX_SPAWN_SETPGROUP` is set in the *spawn-flags* attribute). The default value of this attribute shall be zero.

**RETURN VALUE**

Upon successful completion, *posix\_spawnattr\_getpgroup()* shall return zero and store the value of the *spawn-pgroup* attribute of *attr* into the object referenced by the *pgroup* parameter; otherwise, an error number shall be returned to indicate the error.

Upon successful completion, *posix\_spawnattr\_setpgroup()* shall return zero; otherwise, an error number shall be returned to indicate the error.

**ERRORS**

These functions may fail if:

[EINVAL] The value specified by *attr* is invalid.

The *posix\_spawnattr\_setpgroup()* function may fail if:

[EINVAL] The value of the attribute being set is not valid.

**EXAMPLES**

None.

**APPLICATION USAGE**

These functions are part of the Spawn option and need not be provided on all implementations.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*posix\_spawn()*, *posix\_spawnattr\_destroy()*, *posix\_spawnattr\_getsigdefault()*,  
*posix\_spawnattr\_getflags()*, *posix\_spawnattr\_getschedparam()*, *posix\_spawnattr\_getschedpolicy()*,  
*posix\_spawnattr\_getsigmask()*

XBD `<spawn.h>`

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

48708 **NAME**

48709 posix\_spawnattr\_getschedparam, posix\_spawnattr\_setschedparam — get and set the spawn-  
 48710 schedparam attribute of a spawn attributes object (**ADVANCED REALTIME**)

48711 **SYNOPSIS**

```
48712 SPN PS #include <spawn.h>
48713         #include <sched.h>

48714         int posix_spawnattr_getschedparam(const posix_spawnattr_t
48715             *restrict attr, struct sched_param *restrict schedparam);
48716         int posix_spawnattr_setschedparam(posix_spawnattr_t *restrict attr,
48717             const struct sched_param *restrict schedparam);
```

48718 **DESCRIPTION**

48719 The *posix\_spawnattr\_getschedparam()* function shall obtain the value of the *spawn-schedparam*  
 48720 attribute from the attributes object referenced by *attr*.

48721 The *posix\_spawnattr\_setschedparam()* function shall set the *spawn-schedparam* attribute in an  
 48722 initialized attributes object referenced by *attr*.

48723 The *spawn-schedparam* attribute represents the scheduling parameters to be assigned to the new  
 48724 process image in a spawn operation (if POSIX\_SPAWN\_SETSCHEDULER or  
 48725 POSIX\_SPAWN\_SETSCHEDPARAM is set in the *spawn-flags* attribute). The default value of this  
 48726 attribute is unspecified.

48727 **RETURN VALUE**

48728 Upon successful completion, *posix\_spawnattr\_getschedparam()* shall return zero and store the  
 48729 value of the *spawn-schedparam* attribute of *attr* into the object referenced by the *schedparam*  
 48730 parameter; otherwise, an error number shall be returned to indicate the error.

48731 Upon successful completion, *posix\_spawnattr\_setschedparam()* shall return zero; otherwise, an  
 48732 error number shall be returned to indicate the error.

48733 **ERRORS**

48734 These functions may fail if:

48735 [EINVAL] The value specified by *attr* is invalid.

48736 The *posix\_spawnattr\_setschedparam()* function may fail if:

48737 [EINVAL] The value of the attribute being set is not valid.

48738 **EXAMPLES**

48739 None.

48740 **APPLICATION USAGE**

48741 These functions are part of the Spawn and Process Scheduling options and need not be provided  
 48742 on all implementations.

48743 **RATIONALE**

48744 None.

48745 **FUTURE DIRECTIONS**

48746 None.

48747 **SEE ALSO**

48748 *posix\_spawn()*, *posix\_spawnattr\_destroy()*, *posix\_spawnattr\_getsigdefault()*,  
 48749 *posix\_spawnattr\_getflags()*, *posix\_spawnattr\_getpgroup()*, *posix\_spawnattr\_getschedpolicy()*,  
 48750 *posix\_spawnattr\_getsigmask()*

48751

XBD <sched.h>, <spawn.h>

48752

CHANGE HISTORY

48753

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

48754 **NAME**

48755 posix\_spawnattr\_getschedpolicy, posix\_spawnattr\_setschedpolicy — get and set the spawn-  
 48756 schedpolicy attribute of a spawn attributes object (**ADVANCED REALTIME**)

48757 **SYNOPSIS**

```
48758 SPN PS #include <spawn.h>
48759         #include <sched.h>

48760 int posix_spawnattr_getschedpolicy(const posix_spawnattr_t
48761     *restrict attr, int *restrict schedpolicy);
48762 int posix_spawnattr_setschedpolicy(posix_spawnattr_t *attr,
48763     int schedpolicy);
```

48764 **DESCRIPTION**

48765 The *posix\_spawnattr\_getschedpolicy()* function shall obtain the value of the *spawn-schedpolicy*  
 48766 attribute from the attributes object referenced by *attr*.

48767 The *posix\_spawnattr\_setschedpolicy()* function shall set the *spawn-schedpolicy* attribute in an  
 48768 initialized attributes object referenced by *attr*.

48769 The *spawn-schedpolicy* attribute represents the scheduling policy to be assigned to the new  
 48770 process image in a spawn operation (if POSIX\_SPAWN\_SETSCHEDULER is set in the *spawn-*  
 48771 *flags* attribute). The default value of this attribute is unspecified.

48772 **RETURN VALUE**

48773 Upon successful completion, *posix\_spawnattr\_getschedpolicy()* shall return zero and store the  
 48774 value of the *spawn-schedpolicy* attribute of *attr* into the object referenced by the *schedpolicy*  
 48775 parameter; otherwise, an error number shall be returned to indicate the error.

48776 Upon successful completion, *posix\_spawnattr\_setschedpolicy()* shall return zero; otherwise, an  
 48777 error number shall be returned to indicate the error.

48778 **ERRORS**

48779 These functions may fail if:

48780 [EINVAL] The value specified by *attr* is invalid.

48781 The *posix\_spawnattr\_setschedpolicy()* function may fail if:

48782 [EINVAL] The value of the attribute being set is not valid.

48783 **EXAMPLES**

48784 None.

48785 **APPLICATION USAGE**

48786 These functions are part of the Spawn and Process Scheduling options and need not be provided  
 48787 on all implementations.

48788 **RATIONALE**

48789 None.

48790 **FUTURE DIRECTIONS**

48791 None.

48792 **SEE ALSO**

48793 *posix\_spawn()*, *posix\_spawnattr\_destroy()*, *posix\_spawnattr\_getsigdefault()*,  
 48794 *posix\_spawnattr\_getflags()*, *posix\_spawnattr\_getpgroup()*, *posix\_spawnattr\_getschedparam()*,  
 48795 *posix\_spawnattr\_getsigmask()*

48796 XBD *<sched.h>*, *<spawn.h>*



48797

48798

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

48799 **NAME**

48800 `posix_spawnattr_getsigdefault`, `posix_spawnattr_setsigdefault` — get and set the `spawn-sigdefault`  
 48801 attribute of a spawn attributes object (**ADVANCED REALTIME**)

48802 **SYNOPSIS**

```
48803 SPN    #include <signal.h>
48804        #include <spawn.h>

48805        int posix_spawnattr_getsigdefault(const posix_spawnattr_t
48806            *restrict attr, sigset_t *restrict sigdefault);
48807        int posix_spawnattr_setsigdefault(posix_spawnattr_t *restrict attr,
48808            const sigset_t *restrict sigdefault);
```

48809 **DESCRIPTION**

48810 The `posix_spawnattr_getsigdefault()` function shall obtain the value of the `spawn-sigdefault`  
 48811 attribute from the attributes object referenced by `attr`.

48812 The `posix_spawnattr_setsigdefault()` function shall set the `spawn-sigdefault` attribute in an  
 48813 initialized attributes object referenced by `attr`.

48814 The `spawn-sigdefault` attribute represents the set of signals to be forced to default signal handling  
 48815 in the new process image (if `POSIX_SPAWN_SETSIGDEF` is set in the `spawn-flags` attribute) by a  
 48816 spawn operation. The default value of this attribute shall be an empty signal set.

48817 **RETURN VALUE**

48818 Upon successful completion, `posix_spawnattr_getsigdefault()` shall return zero and store the value  
 48819 of the `spawn-sigdefault` attribute of `attr` into the object referenced by the `sigdefault` parameter;  
 48820 otherwise, an error number shall be returned to indicate the error.

48821 Upon successful completion, `posix_spawnattr_setsigdefault()` shall return zero; otherwise, an error  
 48822 number shall be returned to indicate the error.

48823 **ERRORS**

48824 These functions may fail if:

48825 [EINVAL] The value specified by `attr` is invalid.

48826 The `posix_spawnattr_setsigdefault()` function may fail if:

48827 [EINVAL] The value of the attribute being set is not valid.

48828 **EXAMPLES**

48829 None.

48830 **APPLICATION USAGE**

48831 These functions are part of the Spawn option and need not be provided on all implementations.

48832 **RATIONALE**

48833 None.

48834 **FUTURE DIRECTIONS**

48835 None.

48836 **SEE ALSO**

48837 `posix_spawn()`, `posix_spawnattr_destroy()`, `posix_spawnattr_getflags()`, `posix_spawnattr_getpgroup()`,  
 48838 `posix_spawnattr_getschedparam()`, `posix_spawnattr_getschedpolicy()`, `posix_spawnattr_getsigmask()`

48839 XBD `<signal.h>`, `<spawn.h>`

48840

48841

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

48842 **NAME**

48843 posix\_spawnattr\_getsigmask, posix\_spawnattr\_setsigmask — get and set the spawn-sigmask  
 48844 attribute of a spawn attributes object (**ADVANCED REALTIME**)

48845 **SYNOPSIS**

```
48846 SPN      #include <signal.h>
48847           #include <spawn.h>

48848           int posix_spawnattr_getsigmask(const posix_spawnattr_t *restrict attr,
48849           sigset_t *restrict sigmask);
48850           int posix_spawnattr_setsigmask(posix_spawnattr_t *restrict attr,
48851           const sigset_t *restrict sigmask);
```

48852 **DESCRIPTION**

48853 The *posix\_spawnattr\_getsigmask()* function shall obtain the value of the *spawn-sigmask* attribute  
 48854 from the attributes object referenced by *attr*.

48855 The *posix\_spawnattr\_setsigmask()* function shall set the *spawn-sigmask* attribute in an initialized  
 48856 attributes object referenced by *attr*.

48857 The *spawn-sigmask* attribute represents the signal mask in effect in the new process image of a  
 48858 spawn operation (if `POSIX_SPAWN_SETSIGMASK` is set in the *spawn-flags* attribute). The  
 48859 default value of this attribute is unspecified.

48860 **RETURN VALUE**

48861 Upon successful completion, *posix\_spawnattr\_getsigmask()* shall return zero and store the value  
 48862 of the *spawn-sigmask* attribute of *attr* into the object referenced by the *sigmask* parameter;  
 48863 otherwise, an error number shall be returned to indicate the error.

48864 Upon successful completion, *posix\_spawnattr\_setsigmask()* shall return zero; otherwise, an error  
 48865 number shall be returned to indicate the error.

48866 **ERRORS**

48867 These functions may fail if:

48868 [EINVAL] The value specified by *attr* is invalid.

48869 The *posix\_spawnattr\_setsigmask()* function may fail if:

48870 [EINVAL] The value of the attribute being set is not valid.

48871 **EXAMPLES**

48872 None.

48873 **APPLICATION USAGE**

48874 These functions are part of the Spawn option and need not be provided on all implementations.

48875 **RATIONALE**

48876 None.

48877 **FUTURE DIRECTIONS**

48878 None.

48879 **SEE ALSO**

48880 *posix\_spawn()*, *posix\_spawnattr\_destroy()*, *posix\_spawnattr\_getsigdefault()*,  
 48881 *posix\_spawnattr\_getflags()*, *posix\_spawnattr\_getpgroup()*, *posix\_spawnattr\_getschedparam()*,  
 48882 *posix\_spawnattr\_getschedpolicy()*

48883 XBD `<signal.h>`, `<spawn.h>`

48884

48885

CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

48886 **NAME**48887       posix\_spawnattr\_init — initialize the spawn attributes object (**ADVANCED REALTIME**)48888 **SYNOPSIS**

48889 SPN       #include &lt;spawn.h&gt;

48890       int posix\_spawnattr\_init(posix\_spawnattr\_t \*attr);

48891 **DESCRIPTION**48892       Refer to *posix\_spawnattr\_destroy()*.

48893 **NAME**

48894       posix\_spawnattr\_setflags — set the spawn-flags attribute of a spawn attributes object  
48895       (ADVANCED REALTIME)

48896 **SYNOPSIS**

```
48897 SPN       #include <spawn.h>  
48898       int posix_spawnattr_setflags(posix_spawnattr_t *attr, short flags);
```

48899 **DESCRIPTION**

48900       Refer to *posix\_spawnattr\_getflags()*.

48901 **NAME**

48902       posix\_spawnattr\_setpgroup — set the spawn-pgroup attribute of a spawn attributes object  
48903       (ADVANCED REALTIME)

48904 **SYNOPSIS**

48905 SPN       #include <spawn.h>  
48906       int posix\_spawnattr\_setpgroup(posix\_spawnattr\_t \*attr, pid\_t pgroup);

48907 **DESCRIPTION**

48908       Refer to *posix\_spawnattr\_getpgroup()*.



48909 **NAME**

48910        posix\_spawnattr\_setschedparam — set the spawn-schedparam attribute of a spawn attributes  
48911        object (**ADVANCED REALTIME**)

48912 **SYNOPSIS**

```
48913 SPN PS  #include <sched.h>  
48914          #include <spawn.h>  
  
48915          int posix_spawnattr_setschedparam(posix_spawnattr_t *restrict attr,  
48916          const struct sched_param *restrict schedparam);
```

48917 **DESCRIPTION**

48918        Refer to *posix\_spawnattr\_getschedparam()*.

48919 **NAME**

48920 posix\_spawnattr\_setschedpolicy — set the spawn-schedpolicy attribute of a spawn attributes  
48921 object (**ADVANCED REALTIME**)

48922 **SYNOPSIS**

```
48923 SPN PS #include <sched.h>  
48924 #include <spawn.h>  
  
48925 int posix_spawnattr_setschedpolicy(posix_spawnattr_t *attr,  
48926     int schedpolicy);
```

48927 **DESCRIPTION**

48928 Refer to *posix\_spawnattr\_getschedpolicy()*.

48929 **NAME**

48930 posix\_spawnattr\_setsigdefault — set the spawn-sigdefault attribute of a spawn attributes object  
48931 (**ADVANCED REALTIME**)

48932 **SYNOPSIS**

```
48933 SPN      #include <signal.h>  
48934          #include <spawn.h>  
  
48935          int posix_spawnattr_setsigdefault(posix_spawnattr_t *restrict attr,  
48936          const sigset_t *restrict sigdefault);
```

48937 **DESCRIPTION**

48938 Refer to *posix\_spawnattr\_getsigdefault()*.

48939 **NAME**

48940        `posix_spawnattr_setsigmask` — set the spawn-sigmask attribute of a spawn attributes object  
48941        (**ADVANCED REALTIME**)

48942 **SYNOPSIS**

```
48943 SPN      #include <signal.h>  
48944           #include <spawn.h>  
  
48945           int posix_spawnattr_setsigmask(posix_spawnattr_t *restrict attr,  
48946           const sigset_t *restrict sigmask);
```

48947 **DESCRIPTION**

48948        Refer to *[posix\\_spawnattr\\_getsigmask\(\)](#)*.

48949 **NAME**48950 posix\_spawn — spawn a process (**ADVANCED REALTIME**)48951 **SYNOPSIS**48952 SPN `#include <spawn.h>`

```
48953 int posix_spawn(pid_t *restrict pid, const char *restrict file,  
48954     const posix_spawn_file_actions_t *file_actions,  
48955     const posix_spawnattr_t *restrict attrp,  
48956     char *const argv[restrict], char *const envp[restrict]);
```

48957 **DESCRIPTION**48958 Refer to *posix\_spawn()*.

48959 **NAME**

48960        posix\_trace\_attr\_destroy, posix\_trace\_attr\_init — destroy and initialize the trace stream  
 48961        attributes object (**TRACING**)

48962 **SYNOPSIS**

```
48963 OB TRC #include <trace.h>
48964         int posix_trace_attr_destroy(trace_attr_t *attr);
48965         int posix_trace_attr_init(trace_attr_t *attr);
```

48966 **DESCRIPTION**

48967        The *posix\_trace\_attr\_destroy()* function shall destroy an initialized trace attributes object. A  
 48968        destroyed *attr* attributes object can be reinitialized using *posix\_trace\_attr\_init()*; the results of  
 48969        otherwise referencing the object after it has been destroyed are undefined.

48970        The *posix\_trace\_attr\_init()* function shall initialize a trace attributes object *attr* with the default  
 48971        value for all of the individual attributes used by a given implementation. The read-only  
 48972        *generation-version* and *clock-resolution* attributes of the newly initialized trace attributes object  
 48973        shall be set to their appropriate values (see [Section 2.11.1.2](#), on page 542).

48974        Results are undefined if *posix\_trace\_attr\_init()* is called specifying an already initialized *attr*  
 48975        attributes object.

48976        Implementations may add extensions to the trace attributes object structure as permitted in XBD  
 48977        [Chapter 2](#) (on page 15).

48978        The resulting attributes object (possibly modified by setting individual attributes values), when  
 48979        used by *posix\_trace\_create()*, defines the attributes of the trace stream created. A single attributes  
 48980        object can be used in multiple calls to *posix\_trace\_create()*. After one or more trace streams have  
 48981        been created using an attributes object, any function affecting that attributes object, including  
 48982        destruction, shall not affect any trace stream previously created. An initialized attributes object  
 48983        also serves to receive the attributes of an existing trace stream or trace log when calling the  
 48984        *posix\_trace\_get\_attr()* function.

48985 **RETURN VALUE**

48986        Upon successful completion, these functions shall return a value of zero. Otherwise, they shall  
 48987        return the corresponding error number.

48988 **ERRORS**

48989        The *posix\_trace\_attr\_destroy()* function may fail if:

48990        [EINVAL]        The value of *attr* is invalid.

48991        The *posix\_trace\_attr\_init()* function shall fail if:

48992        [ENOMEM]        Insufficient memory exists to initialize the trace attributes object.

48993 **EXAMPLES**

48994        None.

48995 **APPLICATION USAGE**

48996        None.

48997 **RATIONALE**

48998        None.

**48999 FUTURE DIRECTIONS**

49000 The *posix\_trace\_attr\_destroy()* and *posix\_trace\_attr\_init()* functions may be removed in a future  
49001 version.

**49002 SEE ALSO**

49003 *posix\_trace\_create()*, *posix\_trace\_get\_attr()*, *uname()*

49004 XBD <trace.h>

**49005 CHANGE HISTORY**

49006 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

49007 IEEE PASC Interpretation 1003.1 #123 is applied.

**49008 Issue 7**

49009 The *posix\_trace\_attr\_destroy()* and *posix\_trace\_attr\_init()* functions are marked obsolescent.

**NAME**

posix\_trace\_attr\_getclockres, posix\_trace\_attr\_getcreatetime, posix\_trace\_attr\_getgenversion, posix\_trace\_attr\_getname, posix\_trace\_attr\_setname — retrieve and set information about a trace stream (**TRACING**)

**SYNOPSIS**

```
OB TRC #include <time.h>
#include <trace.h>

int posix_trace_attr_getclockres(const trace_attr_t *attr,
    struct timespec *resolution);
int posix_trace_attr_getcreatetime(const trace_attr_t *attr,
    struct timespec *createtime);

#include <trace.h>

int posix_trace_attr_getgenversion(const trace_attr_t *attr,
    char *genversion);
int posix_trace_attr_getname(const trace_attr_t *attr,
    char *tracename);
int posix_trace_attr_setname(trace_attr_t *attr,
    const char *tracename);
```

**DESCRIPTION**

The *posix\_trace\_attr\_getclockres()* function shall copy the clock resolution of the clock used to generate timestamps from the *clock-resolution* attribute of the attributes object pointed to by the *attr* argument into the structure pointed to by the *resolution* argument.

The *posix\_trace\_attr\_getcreatetime()* function shall copy the trace stream creation time from the *creation-time* attribute of the attributes object pointed to by the *attr* argument into the structure pointed to by the *createtime* argument. The *creation-time* attribute shall represent the time of creation of the trace stream.

The *posix\_trace\_attr\_getgenversion()* function shall copy the string containing version information from the *generation-version* attribute of the attributes object pointed to by the *attr* argument into the string pointed to by the *genversion* argument. The *genversion* argument shall be the address of a character array which can store at least {TRACE\_NAME\_MAX} characters.

The *posix\_trace\_attr\_getname()* function shall copy the string containing the trace name from the *trace-name* attribute of the attributes object pointed to by the *attr* argument into the string pointed to by the *tracename* argument. The *tracename* argument shall be the address of a character array which can store at least {TRACE\_NAME\_MAX} characters.

The *posix\_trace\_attr\_setname()* function shall set the name in the *trace-name* attribute of the attributes object pointed to by the *attr* argument, using the trace name string supplied by the *tracename* argument. If the supplied string contains more than {TRACE\_NAME\_MAX} characters, the name copied into the *trace-name* attribute may be truncated to one less than the length of {TRACE\_NAME\_MAX} characters. The default value is a null string.

**RETURN VALUE**

Upon successful completion, these functions shall return a value of zero. Otherwise, they shall return the corresponding error number.

If successful, the *posix\_trace\_attr\_getclockres()* function stores the *clock-resolution* attribute value in the object pointed to by *resolution*. Otherwise, the content of this object is unspecified.

If successful, the *posix\_trace\_attr\_getcreatetime()* function stores the trace stream creation time in



49055 the object pointed to by *createtime*. Otherwise, the content of this object is unspecified.

49056 If successful, the *posix\_trace\_attr\_getgenversion()* function stores the trace version information in

49057 the string pointed to by *genversion*. Otherwise, the content of this string is unspecified.

49058 If successful, the *posix\_trace\_attr\_getname()* function stores the trace name in the string pointed

49059 to by *tracename*. Otherwise, the content of this string is unspecified.

#### 49060 ERRORS

49061 The *posix\_trace\_attr\_getclockres()*, *posix\_trace\_attr\_getcreatetime()*, *posix\_trace\_attr\_getgenversion()*,

49062 and *posix\_trace\_attr\_getname()* functions may fail if:

49063 [EINVAL] The value specified by one of the arguments is invalid.

#### 49064 EXAMPLES

49065 None.

#### 49066 APPLICATION USAGE

49067 None.

#### 49068 RATIONALE

49069 None.

#### 49070 FUTURE DIRECTIONS

49071 The *posix\_trace\_attr\_getclockres()*, *posix\_trace\_attr\_getcreatetime()*, *posix\_trace\_attr\_getgenversion()*,

49072 *posix\_trace\_attr\_getname()*, and *posix\_trace\_attr\_setname()* functions may be removed in a future

49073 version.

#### 49074 SEE ALSO

49075 *posix\_trace\_attr\_destroy()*, *posix\_trace\_create()*, *posix\_trace\_get\_attr()*, *uname()*

49076 XBD [\*\*<time.h>\*\*](#), [\*\*<trace.h>\*\*](#)

#### 49077 CHANGE HISTORY

49078 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

#### 49079 Issue 7

49080 The *posix\_trace\_attr\_getclockres()*, *posix\_trace\_attr\_getcreatetime()*, *posix\_trace\_attr\_getgenversion()*,

49081 *posix\_trace\_attr\_getname()*, and *posix\_trace\_attr\_setname()* functions are marked obsolescent.

## 49082 NAME

49083 posix\_trace\_attr\_getinherited, posix\_trace\_attr\_getlogfullpolicy,  
 49084 posix\_trace\_attr\_getstreamfullpolicy, posix\_trace\_attr\_setinherited,  
 49085 posix\_trace\_attr\_setlogfullpolicy, posix\_trace\_attr\_setstreamfullpolicy — retrieve and set the  
 49086 behavior of a trace stream (TRACING)

## 49087 SYNOPSIS

```
49088 OB TRC #include <trace.h>
49089 TRI int posix_trace_attr_getinherited(const trace_attr_t *restrict attr,
49090 int *restrict inheritancepolicy);
49091 TRL int posix_trace_attr_getlogfullpolicy(const trace_attr_t *restrict attr,
49092 int *restrict logpolicy);
49093 int posix_trace_attr_getstreamfullpolicy(const trace_attr_t *restrict
49094 attr, int *restrict streampolicy);
49095 TRI int posix_trace_attr_setinherited(trace_attr_t *attr,
49096 int inheritancepolicy);
49097 TRL int posix_trace_attr_setlogfullpolicy(trace_attr_t *attr,
49098 int logpolicy);
49099 int posix_trace_attr_setstreamfullpolicy(trace_attr_t *attr,
49100 int streampolicy);
```

## 49101 DESCRIPTION

49102 TRI The *posix\_trace\_attr\_getinherited()* and *posix\_trace\_attr\_setinherited()* functions, respectively, shall  
 49103 get and set the inheritance policy stored in the *inheritance* attribute for traced processes across the  
 49104 *fork()* and *spawn()* operations. The *inheritance* attribute of the attributes object pointed to by the  
 49105 *attr* argument shall be set to one of the following values defined by manifest constants in the  
 49106 **<trace.h>** header:

## 49107 POSIX\_TRACE\_CLOSE\_FOR\_CHILD

49108 After a *fork()* or *spawn()* operation, the child shall not be traced, and tracing of the parent  
 49109 shall continue.

## 49110 POSIX\_TRACE\_INHERITED

49111 After a *fork()* or *spawn()* operation, if the parent is being traced, its child shall be  
 49112 concurrently traced using the same trace stream.

49113 The default value for the *inheritance* attribute is **POSIX\_TRACE\_CLOSE\_FOR\_CHILD**.

49114 TRL The *posix\_trace\_attr\_getlogfullpolicy()* and *posix\_trace\_attr\_setlogfullpolicy()* functions,  
 49115 respectively, shall get and set the trace log full policy stored in the *log-full-policy* attribute of the  
 49116 attributes object pointed to by the *attr* argument.

49117 The *log-full-policy* attribute shall be set to one of the following values defined by manifest  
 49118 constants in the **<trace.h>** header:

## 49119 POSIX\_TRACE\_LOOP

49120 The trace log shall loop until the associated trace stream is stopped. This policy means that  
 49121 when the trace log gets full, the file system shall reuse the resources allocated to the oldest  
 49122 trace events that were recorded. In this way, the trace log will always contain the most  
 49123 recent trace events flushed.

## 49124 POSIX\_TRACE\_UNTIL\_FULL

49125 The trace stream shall be flushed to the trace log until the trace log is full. This condition can  
 49126 be deduced from the *posix\_log\_full\_status* member status (see the **posix\_trace\_status\_info**  
 49127 structure defined in **<trace.h>**). The last recorded trace event shall be the  
 49128 **POSIX\_TRACE\_STOP** trace event.

**POSIX\_TRACE\_APPEND**

The associated trace stream shall be flushed to the trace log without log size limitation. If the application specifies **POSIX\_TRACE\_APPEND**, the implementation shall ignore the *log-max-size* attribute.

The default value for the *log-full-policy* attribute is **POSIX\_TRACE\_LOOP**.

The *posix\_trace\_attr\_getstreamfullpolicy()* and *posix\_trace\_attr\_setstreamfullpolicy()* functions, respectively, shall get and set the trace stream full policy stored in the *stream-full-policy* attribute of the attributes object pointed to by the *attr* argument.

The *stream-full-policy* attribute shall be set to one of the following values defined by manifest constants in the **<trace.h>** header:

**POSIX\_TRACE\_LOOP**

The trace stream shall loop until explicitly stopped by the *posix\_trace\_stop()* function. This policy means that when the trace stream is full, the trace system shall reuse the resources allocated to the oldest trace events recorded. In this way, the trace stream will always contain the most recent trace events recorded.

**POSIX\_TRACE\_UNTIL\_FULL**

The trace stream will run until the trace stream resources are exhausted. Then the trace stream will stop. This condition can be deduced from *posix\_stream\_status* and *posix\_stream\_full\_status* (see the **posix\_trace\_status\_info** structure defined in **<trace.h>**). When this trace stream is read, a **POSIX\_TRACE\_STOP** trace event shall be reported after reporting the last recorded trace event. The trace system shall reuse the resources allocated to any trace events already reported—see the *posix\_trace\_getnext\_event()*, *posix\_trace\_trygetnext\_event()*, and *posix\_trace\_timedgetnext\_event()* functions—or already flushed for an active trace stream with log if the Trace Log option is supported; see the *posix\_trace\_flush()* function. The trace system shall restart the trace stream when it is empty and may restart it sooner. A **POSIX\_TRACE\_START** trace event shall be reported before reporting the next recorded trace event.

**POSIX\_TRACE\_FLUSH**

If the Trace Log option is supported, this policy is identical to the **POSIX\_TRACE\_UNTIL\_FULL** trace stream full policy except that the trace stream shall be flushed regularly as if *posix\_trace\_flush()* had been explicitly called. Defining this policy for an active trace stream without log shall be invalid.

The default value for the *stream-full-policy* attribute shall be **POSIX\_TRACE\_LOOP** for an active trace stream without log.

If the Trace Log option is supported, the default value for the *stream-full-policy* attribute shall be **POSIX\_TRACE\_FLUSH** for an active trace stream with log.

**RETURN VALUE**

Upon successful completion, these functions shall return a value of zero. Otherwise, they shall return the corresponding error number.

If successful, the *posix\_trace\_attr\_getinherited()* function shall store the *inheritance* attribute value in the object pointed to by *inheritancepolicy*. Otherwise, the content of this object is undefined.

If successful, the *posix\_trace\_attr\_getlogfullpolicy()* function shall store the *log-full-policy* attribute value in the object pointed to by *logpolicy*. Otherwise, the content of this object is undefined.

If successful, the *posix\_trace\_attr\_getstreamfullpolicy()* function shall store the *stream-full-policy* attribute value in the object pointed to by *streampolicy*. Otherwise, the content of this object is undefined.

49175 **ERRORS**

49176 These functions may fail if:

49177 [EINVAL] The value specified by at least one of the arguments is invalid.

49178 **EXAMPLES**

49179 None.

49180 **APPLICATION USAGE**

49181 None.

49182 **RATIONALE**

49183 None.

49184 **FUTURE DIRECTIONS**

49185 The following functions:

49186 *posix\_trace\_attr\_getinherited()*  
 49187 *posix\_trace\_attr\_getlogfullpolicy()*  
 49188 *posix\_trace\_attr\_getstreamfullpolicy()*  
 49189 *posix\_trace\_attr\_setinherited()*  
 49190 *posix\_trace\_attr\_setlogfullpolicy()*  
 49191 *posix\_trace\_attr\_setstreamfullpolicy()*

49192 may be removed in a future version.

49193 **SEE ALSO**

49194 *fork()*, *posix\_trace\_attr\_destroy()*, *posix\_trace\_create()*, *posix\_trace\_get\_attr()*,  
 49195 *posix\_trace\_getnext\_event()*, *posix\_trace\_start()*

49196 XBD &lt;trace.h&gt;

49197 **CHANGE HISTORY**

49198 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

49199 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/39 is applied, adding the TRL and TRC  
 49200 margin codes to the *posix\_trace\_attr\_setlogfullpolicy()* function.

49201 **Issue 7**

49202 SD5-XSH-ERN-116 is applied, adding the missing **restrict** keyword to the  
 49203 *posix\_trace\_attr\_getstreamfullpolicy()* function declaration.

49204 These functions are marked obsolescent.

**NAME**

posix\_trace\_attr\_getlogsize, posix\_trace\_attr\_getmaxdatasize,  
 posix\_trace\_attr\_getmaxsystemeventsize, posix\_trace\_attr\_getmaxusereventsize,  
 posix\_trace\_attr\_getstreamsize, posix\_trace\_attr\_setlogsize, posix\_trace\_attr\_setmaxdatasize,  
 posix\_trace\_attr\_setstreamsize — retrieve and set trace stream size attributes (TRACING)

**SYNOPSIS**

```
OB TRC  #include <sys/types.h>
        #include <trace.h>

TRL     int posix_trace_attr_getlogsize(const trace_attr_t *restrict attr,
        size_t *restrict logsize);

        int posix_trace_attr_getmaxdatasize(const trace_attr_t *restrict attr,
        size_t *restrict maxdatasize);

        int posix_trace_attr_getmaxsystemeventsize(
        const trace_attr_t *restrict attr,
        size_t *restrict eventsize);

        int posix_trace_attr_getmaxusereventsize(
        const trace_attr_t *restrict attr,
        size_t data_len, size_t *restrict eventsize);

        int posix_trace_attr_getstreamsize(const trace_attr_t *restrict attr,
        size_t *restrict streamsize);

TRL     int posix_trace_attr_setlogsize(trace_attr_t *attr,
        size_t logsize);

        int posix_trace_attr_setmaxdatasize(trace_attr_t *attr,
        size_t maxdatasize);

        int posix_trace_attr_setstreamsize(trace_attr_t *attr,
        size_t streamsize);
```

**DESCRIPTION**

The *posix\_trace\_attr\_getlogsize()* function shall copy the log size, in bytes, from the *log-max-size* attribute of the attributes object pointed to by the *attr* argument into the variable pointed to by the *logsize* argument. This log size is the maximum total of bytes that shall be allocated for system and user trace events in the trace log. The default value for the *log-max-size* attribute is implementation-defined.

The *posix\_trace\_attr\_setlogsize()* function shall set the maximum allowed size, in bytes, in the *log-max-size* attribute of the attributes object pointed to by the *attr* argument, using the size value supplied by the *logsize* argument.

The trace log size shall be used if the *log-full-policy* attribute is set to *POSIX\_TRACE\_LOOP* or *POSIX\_TRACE\_UNTIL\_FULL*. If the *log-full-policy* attribute is set to *POSIX\_TRACE\_APPEND*, the implementation shall ignore the *log-max-size* attribute.

The *posix\_trace\_attr\_getmaxdatasize()* function shall copy the maximum user trace event data size, in bytes, from the *max-data-size* attribute of the attributes object pointed to by the *attr* argument into the variable pointed to by the *maxdatasize* argument. The default value for the *max-data-size* attribute is implementation-defined.

The *posix\_trace\_attr\_getmaxsystemeventsize()* function shall calculate the maximum memory size, in bytes, required to store a single system trace event. This value is calculated for the trace stream attributes object pointed to by the *attr* argument and is returned in the variable pointed to by the *eventsize* argument.

The values returned as the maximum memory sizes of the user and system trace events shall be such that if the sum of the maximum memory sizes of a set of the trace events that may be

recorded in a trace stream is less than or equal to the *stream-min-size* attribute of that trace stream, the system provides the necessary resources for recording all those trace events, without loss.

The *posix\_trace\_attr\_getmaxusereventsize()* function shall calculate the maximum memory size, in bytes, required to store a single user trace event generated by a call to *posix\_trace\_event()* with a *data\_len* parameter equal to the *data\_len* value specified in this call. This value is calculated for the trace stream attributes object pointed to by the *attr* argument and is returned in the variable pointed to by the *eventsize* argument.

The *posix\_trace\_attr\_getstreamsize()* function shall copy the stream size, in bytes, from the *stream-min-size* attribute of the attributes object pointed to by the *attr* argument into the variable pointed to by the *streamsize* argument.

This stream size is the current total memory size reserved for system and user trace events in the trace stream. The default value for the *stream-min-size* attribute is implementation-defined. The stream size refers to memory used to store trace event records. Other stream data (for example, trace attribute values) shall not be included in this size.

The *posix\_trace\_attr\_setmaxdatasize()* function shall set the maximum allowed size, in bytes, in the *max-data-size* attribute of the attributes object pointed to by the *attr* argument, using the size value supplied by the *maxdatasize* argument. This maximum size is the maximum allowed size for the user data argument which may be passed to *posix\_trace\_event()*. The implementation shall be allowed to truncate data passed to *trace\_user\_event* which is longer than *maxdatasize*.

The *posix\_trace\_attr\_setstreamsize()* function shall set the minimum allowed size, in bytes, in the *stream-min-size* attribute of the attributes object pointed to by the *attr* argument, using the size value supplied by the *streamsize* argument.

#### RETURN VALUE

Upon successful completion, these functions shall return a value of zero. Otherwise, they shall return the corresponding error number.

TRL The *posix\_trace\_attr\_getlogsize()* function stores the maximum trace log allowed size in the object pointed to by *logsize*, if successful.

The *posix\_trace\_attr\_getmaxdatasize()* function stores the maximum trace event record memory size in the object pointed to by *maxdatasize*, if successful.

The *posix\_trace\_attr\_getmaxsystemeventsize()* function stores the maximum memory size to store a single system trace event in the object pointed to by *eventsize*, if successful.

The *posix\_trace\_attr\_getmaxusereventsize()* function stores the maximum memory size to store a single user trace event in the object pointed to by *eventsize*, if successful.

The *posix\_trace\_attr\_getstreamsize()* function stores the maximum trace stream allowed size in the object pointed to by *streamsize*, if successful.

#### ERRORS

These functions may fail if:

[EINVAL] The value specified by one of the arguments is invalid.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

The following functions:

*posix\_trace\_attr\_getlogsize()**posix\_trace\_attr\_getmaxdatasize()**posix\_trace\_attr\_getmaxsystemeventsize()**posix\_trace\_attr\_getmaxusereventsize()**posix\_trace\_attr\_getstreamsize()**posix\_trace\_attr\_setlogsize()**posix\_trace\_attr\_setmaxdatasize()**posix\_trace\_attr\_setstreamsize()*

may be removed in a future version.

**SEE ALSO***posix\_trace\_attr\_destroy(), posix\_trace\_create(), posix\_trace\_event(), posix\_trace\_get\_attr()*XBD **<sys/types.h>**, **<trace.h>****CHANGE HISTORY**

First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

**Issue 7**

These functions are marked obsolescent.

49316 **NAME**49317 posix\_trace\_attr\_getname — retrieve and set information about a trace stream (**TRACING**)49318 **SYNOPSIS**49319 OB TRC `#include <trace.h>`49320 `int posix_trace_attr_getname(const trace_attr_t *attr,`  
49321 `char *tracename);`49322 **DESCRIPTION**49323 Refer to *posix\_trace\_attr\_getclockres()*.



49324 **NAME**

49325       posix\_trace\_attr\_getstreamfullpolicy — retrieve and set the behavior of a trace stream  
49326       (**TRACING**)

49327 **SYNOPSIS**

```
49328 OB TRC #include <trace.h>  
49329         int posix_trace_attr_getstreamfullpolicy(const trace_attr_t *restrict  
49330             attr, int *restrict streampolicy);
```

49331 **DESCRIPTION**

49332       Refer to *posix\_trace\_attr\_getinherited()*.

49333 **NAME**49334        posix\_trace\_attr\_getstreamsize — retrieve and set trace stream size attributes (**TRACING**)49335 **SYNOPSIS**

```
49336 OB TRC #include <sys/types.h>
49337         #include <trace.h>
49338
49338         int posix_trace_attr_getstreamsize(const trace_attr_t *restrict attr,
49339         size_t *restrict streamsize);
```

49340 **DESCRIPTION**49341        Refer to *posix\_trace\_attr\_getlogsize()*.

49342 **NAME**

49343       posix\_trace\_attr\_init — initialize the trace stream attributes object (TRACING)

49344 **SYNOPSIS**

49345 OB TRC   #include &lt;trace.h&gt;

49346       int posix\_trace\_attr\_init(trace\_attr\_t \*attr);

49347 **DESCRIPTION**49348       Refer to *posix\_trace\_attr\_destroy()*.

49349 **NAME**

49350 posix\_trace\_attr\_setinherited, posix\_trace\_attr\_setlogfullpolicy — retrieve and set the behavior  
49351 of a trace stream (TRACING)

49352 **SYNOPSIS**

```
49353 OB TRC #include <trace.h>
49354 TRI int posix_trace_attr_setinherited(trace_attr_t *attr,
49355 int inheritancepolicy);
49356 TRL int posix_trace_attr_setlogfullpolicy(trace_attr_t *attr,
49357 int logpolicy);
```

49358 **DESCRIPTION**

49359 Refer to *posix\_trace\_attr\_getinherited()*.

49360 **NAME**

49361 posix\_trace\_attr\_setlogsize, posix\_trace\_attr\_setmaxdatasize — retrieve and set trace stream size  
 49362 attributes (**TRACING**)

49363 **SYNOPSIS**

```
49364 OB TRC #include <sys/types.h>
49365          #include <trace.h>

49366 TRL     int posix_trace_attr_setlogsize(trace_attr_t *attr,
49367          size_t logsize);
49368 OB TRC   int posix_trace_attr_setmaxdatasize(trace_attr_t *attr,
49369          size_t maxdatasize);
```

49370 **DESCRIPTION**

49371 Refer to *posix\_trace\_attr\_getlogsize()*.

49372 **NAME**49373 posix\_trace\_attr\_setname — retrieve and set information about a trace stream (**TRACING**)49374 **SYNOPSIS**49375 OB TRC `#include <trace.h>`49376 `int posix_trace_attr_setname(trace_attr_t *attr,`  
49377 `const char *tracename);`49378 **DESCRIPTION**49379 Refer to *posix\_trace\_attr\_getclockres()*.

49380 **NAME**

49381 posix\_trace\_attr\_setstreamfullpolicy — retrieve and set the behavior of a trace stream  
49382 (**TRACING**)

49383 **SYNOPSIS**

```
49384 OB TRC #include <trace.h>
49385 int posix_trace_attr_setstreamfullpolicy(trace_attr_t *attr,
49386 int streampolicy);
```

49387 **DESCRIPTION**

49388 Refer to *posix\_trace\_attr\_getinherited()*.

49389 **NAME**

49390 posix\_trace\_attr\_setstreamsize — retrieve and set trace stream size attributes (TRACING)

49391 **SYNOPSIS**

```
49392 OB TRC #include <sys/types.h>
49393         #include <trace.h>
49394         int posix_trace_attr_setstreamsize(trace_attr_t *attr,
49395         size_t streamsize);
```

49396 **DESCRIPTION**49397 Refer to *posix\_trace\_attr\_getlogsize()*.



49398 **NAME**

49399       posix\_trace\_clear — clear trace stream and trace log (TRACING)

49400 **SYNOPSIS**

```
49401 OB TRC  #include <sys/types.h>
49402          #include <trace.h>
49403          int posix_trace_clear(trace_id_t trid);
```

49404 **DESCRIPTION**

49405       The *posix\_trace\_clear()* function shall reinitialize the trace stream identified by the argument *trid*  
49406       as if it were returning from the *posix\_trace\_create()* function, except that the same allocated  
49407       resources shall be reused, the mapping of trace event type identifiers to trace event names shall  
49408       be unchanged, and the trace stream status shall remain unchanged (that is, if it was running, it  
49409       remains running and if it was suspended, it remains suspended).

49410       All trace events in the trace stream recorded before the call to *posix\_trace\_clear()* shall be lost. The  
49411       *posix\_stream\_full\_status* status shall be set to POSIX\_TRACE\_NOT\_FULL. There is no guarantee  
49412       that all trace events that occurred during the *posix\_trace\_clear()* call are recorded; the behavior  
49413       with respect to trace points that may occur during this call is unspecified.

49414 OB TRL    If the Trace Log option is supported and the trace stream has been created with a log, the  
49415       *posix\_trace\_clear()* function shall reinitialize the trace stream with the same behavior as if the  
49416       trace stream was created without the log, plus it shall reinitialize the trace log associated with  
49417       the trace stream identified by the argument *trid* as if it were returning from the  
49418       *posix\_trace\_create\_withlog()* function, except that the same allocated resources, for the trace log,  
49419       may be reused and the associated trace stream status remains unchanged. The first trace event  
49420       recorded in the trace log after the call to *posix\_trace\_clear()* shall be the same as the first trace  
49421       event recorded in the active trace stream after the call to *posix\_trace\_clear()*. The  
49422       *posix\_log\_full\_status* status shall be set to POSIX\_TRACE\_NOT\_FULL. There is no guarantee that  
49423       all trace events that occurred during the *posix\_trace\_clear()* call are recorded in the trace log; the  
49424       behavior with respect to trace points that may occur during this call is unspecified. If the log full  
49425       policy is POSIX\_TRACE\_APPEND, the effect of a call to this function is unspecified for the trace  
49426       log associated with the trace stream identified by the *trid* argument.

49427 **RETURN VALUE**

49428       Upon successful completion, the *posix\_trace\_clear()* function shall return a value of zero.  
49429       Otherwise, it shall return the corresponding error number.

49430 **ERRORS**

49431       The *posix\_trace\_clear()* function shall fail if:

49432       [EINVAL]       The value of the *trid* argument does not correspond to an active trace stream.

49433 **EXAMPLES**

49434       None.

49435 **APPLICATION USAGE**

49436       None.

49437 **RATIONALE**

49438       None.

49439 **FUTURE DIRECTIONS**

49440       The *posix\_trace\_clear()* function may be removed in a future version.

49441 **SEE ALSO**

49442 *posix\_trace\_attr\_destroy()*, *posix\_trace\_create()*, *posix\_trace\_get\_attr()*

49443 XBD *<sys/types.h>*, *<trace.h>*

49444 **CHANGE HISTORY**

49445 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

49446 IEEE PASC Interpretation 1003.1 #123 is applied.

49447 **Issue 7**

49448 The *posix\_trace\_clear()* function is marked obsolescent.

**NAME**

posix\_trace\_close, posix\_trace\_open, posix\_trace\_rewind — trace log management (**TRACING**)

**SYNOPSIS**

```
#include <trace.h>
```

```
int posix_trace_close(trace_id_t trid);
int posix_trace_open(int file_desc, trace_id_t *trid);
int posix_trace_rewind(trace_id_t trid);
```

**DESCRIPTION**

The *posix\_trace\_close()* function shall deallocate the trace log identifier indicated by *trid*, and all of its associated resources. If there is no valid trace log pointed to by the *trid*, this function shall fail.

The *posix\_trace\_open()* function shall allocate the necessary resources and establish the connection between a trace log identified by the *file\_desc* argument and a trace stream identifier identified by the object pointed to by the *trid* argument. The *file\_desc* argument should be a valid open file descriptor that corresponds to a trace log. The *file\_desc* argument shall be open for reading. The current trace event timestamp, which specifies the timestamp of the trace event that will be read by the next call to *posix\_trace\_getnext\_event()*, shall be set to the timestamp of the oldest trace event recorded in the trace log identified by *trid*.

The *posix\_trace\_open()* function shall return a trace stream identifier in the variable pointed to by the *trid* argument, that may only be used by the following functions:

|                                               |                                    |
|-----------------------------------------------|------------------------------------|
| <i>posix_trace_close()</i>                    | <i>posix_trace_get_attr()</i>      |
| <i>posix_trace_eventid_equal()</i>            | <i>posix_trace_get_status()</i>    |
| <i>posix_trace_eventid_get_name()</i>         | <i>posix_trace_getnext_event()</i> |
| <i>posix_trace_eventtypelist_getnext_id()</i> | <i>posix_trace_rewind()</i>        |
| <i>posix_trace_eventtypelist_rewind()</i>     |                                    |

In particular, notice that the operations normally used by a trace controller process, such as *posix\_trace\_start()*, *posix\_trace\_stop()*, or *posix\_trace\_shutdown()*, cannot be invoked using the trace stream identifier returned by the *posix\_trace\_open()* function.

The *posix\_trace\_rewind()* function shall reset the current trace event timestamp, which specifies the timestamp of the trace event that will be read by the next call to *posix\_trace\_getnext\_event()*, to the timestamp of the oldest trace event recorded in the trace log identified by *trid*.

**RETURN VALUE**

Upon successful completion, these functions shall return a value of zero. Otherwise, they shall return the corresponding error number.

If successful, the *posix\_trace\_open()* function stores the trace stream identifier value in the object pointed to by *trid*.

**ERRORS**

The *posix\_trace\_open()* function shall fail if:

|          |                                                                                     |
|----------|-------------------------------------------------------------------------------------|
| [EINTR]  | The operation was interrupted by a signal and thus no trace log was opened.         |
| [EINVAL] | The object pointed to by <i>file_desc</i> does not correspond to a valid trace log. |

49489           The *posix\_trace\_close()* and *posix\_trace\_rewind()* functions may fail if:  
49490           [EINVAL]           The object pointed to by *trid* does not correspond to a valid trace log.

49491 **EXAMPLES**  
49492           None.

49493 **APPLICATION USAGE**  
49494           None.

49495 **RATIONALE**  
49496           None.

49497 **FUTURE DIRECTIONS**  
49498           The *posix\_trace\_close()*, *posix\_trace\_open()*, and *posix\_trace\_rewind()* functions may be removed in  
49499           a future version.

49500 **SEE ALSO**  
49501           *posix\_trace\_get\_attr()*, *posix\_trace\_get\_filter()*, *posix\_trace\_getnext\_event()*  
49502           XBD <trace.h>

49503 **CHANGE HISTORY**  
49504           First released in Issue 6. Derived from IEEE Std 1003.1q-2000.  
49505           IEEE PASC Interpretation 1003.1 #123 is applied.

49506 **Issue 7**  
49507           The *posix\_trace\_close()*, *posix\_trace\_open()*, and *posix\_trace\_rewind()* functions are marked  
49508           obsolescent.

49509 **NAME**

49510 `posix_trace_create`, `posix_trace_create_withlog`, `posix_trace_flush`, `posix_trace_shutdown` —  
 49511 trace stream initialization, flush, and shutdown from a process (**TRACING**)

49512 **SYNOPSIS**

```

49513 OB TRC #include <sys/types.h>
49514         #include <trace.h>

49515         int posix_trace_create(pid_t pid,
49516                               const trace_attr_t *restrict attr,
49517                               trace_id_t *restrict trid);
49518 TRL     int posix_trace_create_withlog(pid_t pid,
49519   const trace_attr_t *restrict attr, int file_desc,
49520   trace_id_t *restrict trid);
49521         int posix_trace_flush(trace_id_t trid);
49522 OB TRC  int posix_trace_shutdown(trace_id_t trid);

```

49523 **DESCRIPTION**

49524 The `posix_trace_create()` function shall create an active trace stream. It allocates all the resources  
 49525 needed by the trace stream being created for tracing the process specified by *pid* in accordance  
 49526 with the *attr* argument. The *attr* argument represents the initial attributes of the trace stream and  
 49527 shall have been initialized by the function `posix_trace_attr_init()` prior to the `posix_trace_create()`  
 49528 call. If the argument *attr* is NULL, the default attributes shall be used. The *attr* attributes object  
 49529 shall be manipulated through a set of functions described in the `posix_trace_attr` family of  
 49530 functions. If the attributes of the object pointed to by *attr* are modified later, the attributes of the  
 49531 trace stream shall not be affected. The *creation-time* attribute of the newly created trace stream  
 49532 shall be set to the value of the system clock, if the Timers option is not supported, or to the value  
 49533 of the CLOCK\_REALTIME clock, if the Timers option is supported.

49534 The *pid* argument represents the target process to be traced. If the process executing this function  
 49535 does not have appropriate privileges to trace the process identified by *pid*, an error shall be  
 49536 returned. If the *pid* argument is zero, the calling process shall be traced.

49537 The `posix_trace_create()` function shall store the trace stream identifier of the new trace stream in  
 49538 the object pointed to by the *trid* argument. This trace stream identifier shall be used in  
 49539 subsequent calls to control tracing. The *trid* argument may only be used by the following  
 49540 functions:

|       |                                                     |                                               |
|-------|-----------------------------------------------------|-----------------------------------------------|
| 49541 | <code>posix_trace_clear()</code>                    | <code>posix_trace_getnext_event()</code>      |
| 49542 | <code>posix_trace_eventid_equal()</code>            | <code>posix_trace_shutdown()</code>           |
| 49543 | <code>posix_trace_eventid_get_name()</code>         | <code>posix_trace_start()</code>              |
| 49544 | <code>posix_trace_eventtypelist_getnext_id()</code> | <code>posix_trace_stop()</code>               |
| 49545 | <code>posix_trace_eventtypelist_rewind()</code>     | <code>posix_trace_timedgetnext_event()</code> |
| 49546 | <code>posix_trace_get_attr()</code>                 | <code>posix_trace_trid_eventid_open()</code>  |
| 49547 | <code>posix_trace_get_status()</code>               | <code>posix_trace_trygetnext_event()</code>   |

49548 TEF If the Trace Event Filter option is supported, the following additional functions may use the *trid*  
 49549 argument:

```

49550 posix_trace_get_filter()   posix_trace_set_filter()

```

49551 In particular, notice that the operations normally used by a trace analyzer process, such as  
 49552 `posix_trace_rewind()` or `posix_trace_close()`, cannot be invoked using the trace stream identifier  
 49553 returned by the `posix_trace_create()` function.

49554 TEF A trace stream shall be created in a suspended state. If the Trace Event Filter option is  
49555 supported, its trace event type filter shall be empty.

49556 The *posix\_trace\_create()* function may be called multiple times from the same or different  
49557 processes, with the system-wide limit indicated by the runtime invariant value  
49558 {TRACE\_SYS\_MAX}, which has the minimum value {\_POSIX\_TRACE\_SYS\_MAX}.

49559 The trace stream identifier returned by the *posix\_trace\_create()* function in the argument pointed  
49560 to by *trid* is valid only in the process that made the function call. If it is used from another  
49561 process, that is a child process, in functions defined in POSIX.1-2008, these functions shall return  
49562 with the error [EINVAL].

49563 TRL The *posix\_trace\_create\_withlog()* function shall be equivalent to *posix\_trace\_create()*, except that it  
49564 associates a trace log with this stream. The *file\_desc* argument shall be the file descriptor  
49565 designating the trace log destination. The function shall fail if this file descriptor refers to a file  
49566 with a file type that is not compatible with the log policy associated with the trace log. The list of  
49567 the appropriate file types that are compatible with each log policy is implementation-defined.

49568 The *posix\_trace\_create\_withlog()* function shall return in the parameter pointed to by *trid* the trace  
49569 stream identifier, which uniquely identifies the newly created trace stream, and shall be used in  
49570 subsequent calls to control tracing. The *trid* argument may only be used by the following  
49571 functions:

|                                                     |                                         |
|-----------------------------------------------------|-----------------------------------------|
| 49572 <i>posix_trace_clear()</i>                    | <i>posix_trace_get_status()</i>         |
| 49573 <i>posix_trace_eventid_equal()</i>            | <i>posix_trace_getnext_event()</i>      |
| 49574 <i>posix_trace_eventid_get_name()</i>         | <i>posix_trace_shutdown()</i>           |
| 49575 <i>posix_trace_eventtypelist_getnext_id()</i> | <i>posix_trace_start()</i>              |
| 49576 <i>posix_trace_eventtypelist_rewind()</i>     | <i>posix_trace_stop()</i>               |
| 49577 <i>posix_trace_flush()</i>                    | <i>posix_trace_timedgetnext_event()</i> |
| 49578 <i>posix_trace_get_attr()</i>                 | <i>posix_trace_trid_eventid_open()</i>  |

49579 TEF TRL If the Trace Event Filter option is supported, the following additional functions may use the *trid*  
49580 argument:

49581 *posix\_trace\_get\_filter()*    *posix\_trace\_set\_filter()*

49582 TRL In particular, notice that the operations normally used by a trace analyzer process, such as  
49583 *posix\_trace\_rewind()* or *posix\_trace\_close()*, cannot be invoked using the trace stream identifier  
49584 returned by the *posix\_trace\_create\_withlog()* function.

49585 The *posix\_trace\_flush()* function shall initiate a flush operation which copies the contents of the  
49586 trace stream identified by the argument *trid* into the trace log associated with the trace stream at  
49587 the creation time. If no trace log has been associated with the trace stream pointed to by *trid*, this  
49588 function shall return an error. The termination of the flush operation can be polled by the  
49589 *posix\_trace\_get\_status()* function. During the flush operation, it shall be possible to trace new  
49590 trace events up to the point when the trace stream becomes full. After flushing is completed, the  
49591 space used by the flushed trace events shall be available for tracing new trace events.

49592 If flushing the trace stream causes the resulting trace log to become full, the trace log full policy  
49593 shall be applied. If the trace *log-full-policy* attribute is set, the following occurs:

49594 POSIX\_TRACE\_UNTIL\_FULL

49595 The trace events that have not yet been flushed shall be discarded.

49596 POSIX\_TRACE\_LOOP  
 49597 The trace events that have not yet been flushed shall be written to the beginning of the trace  
 49598 log, overwriting previous trace events stored there.

49599 POSIX\_TRACE\_APPEND  
 49600 The trace events that have not yet been flushed shall be appended to the trace log.

49601 The *posix\_trace\_shutdown()* function shall stop the tracing of trace events in the trace stream  
 49602 identified by *trid*, as if *posix\_trace\_stop()* had been invoked. The *posix\_trace\_shutdown()* function  
 49603 shall free all the resources associated with the trace stream.

49604 The *posix\_trace\_shutdown()* function shall not return until all the resources associated with the  
 49605 trace stream have been freed. When the *posix\_trace\_shutdown()* function returns, the *trid*  
 49606 argument becomes an invalid trace stream identifier. A call to this function shall unconditionally  
 49607 deallocate the resources regardless of whether all trace events have been retrieved by the  
 49608 analyzer process. Any thread blocked on one of the *trace\_getnext\_event()* functions (which  
 49609 specified this *trid*) before this call is unblocked with the error [EINVAL].

49610 If the process exits, invokes a member of the *exec* family of functions, or is terminated, the trace  
 49611 streams that the process had created and that have not yet been shut down, shall be  
 49612 automatically shut down as if an explicit call were made to the *posix\_trace\_shutdown()* function.

49613 TRL For an active trace stream with log, when the *posix\_trace\_shutdown()* function is called, all trace  
 49614 events that have not yet been flushed to the trace log shall be flushed, as in the  
 49615 *posix\_trace\_flush()* function, and the trace log shall be closed.

49616 When a trace log is closed, all the information that may be retrieved later from the trace log  
 49617 through the trace interface shall have been written to the trace log. This information includes the  
 49618 trace attributes, the list of trace event types (with the mapping between trace event names and  
 49619 trace event type identifiers), and the trace status.

49620 In addition, unspecified information shall be written to the trace log to allow detection of a valid  
 49621 trace log during the *posix\_trace\_open()* operation.

49622 The *posix\_trace\_shutdown()* function shall not return until all trace events have been flushed.

## 49623 RETURN VALUE

49624 Upon successful completion, these functions shall return a value of zero. Otherwise, they shall  
 49625 return the corresponding error number.

49626 TRL The *posix\_trace\_create()* and *posix\_trace\_create\_withlog()* functions store the trace stream  
 49627 identifier value in the object pointed to by *trid*, if successful.

## 49628 ERRORS

49629 TRL The *posix\_trace\_create()* and *posix\_trace\_create\_withlog()* functions shall fail if:

49630 [EAGAIN] No more trace streams can be started now. {TRACE\_SYS\_MAX} has been  
 49631 exceeded.

49632 [EINTR] The operation was interrupted by a signal. No trace stream was created.

49633 [EINVAL] One or more of the trace parameters specified by the *attr* parameter is invalid.

49634 [ENOMEM] The implementation does not currently have sufficient memory to create the  
 49635 trace stream with the specified parameters.

49636 [EPERM] The caller does not have appropriate privileges to trace the process specified  
 49637 by *pid*.



49638 [ESRCH] The *pid* argument does not refer to an existing process.

49639 TRL The *posix\_trace\_create\_withlog()* function shall fail if:

49640 [EBADF] The *file\_desc* argument is not a valid file descriptor open for writing.

49641 [EINVAL] The *file\_desc* argument refers to a file with a file type that does not support the

49642 log policy associated with the trace log.

49643 [ENOSPC] No space left on device. The device corresponding to the argument *file\_desc*

49644 does not contain the space required to create this trace log.

49645 TRL The *posix\_trace\_flush()* and *posix\_trace\_shutdown()* functions shall fail if:

49646 [EINVAL] The value of the *trid* argument does not correspond to an active trace stream

49647 with log.

49648 [EFBIG] The trace log file has attempted to exceed an implementation-defined

49649 maximum file size.

49650 [ENOSPC] No space left on device.

#### 49651 EXAMPLES

49652 None.

#### 49653 APPLICATION USAGE

49654 None.

#### 49655 RATIONALE

49656 None.

#### 49657 FUTURE DIRECTIONS

49658 The *posix\_trace\_create()*, *posix\_trace\_create\_withlog()*, *posix\_trace\_flush()*, and

49659 *posix\_trace\_shutdown()* functions may be removed in a future version.

#### 49660 SEE ALSO

49661 *clock\_getres()*, *exec*, *posix\_trace\_attr\_destroy()*, *posix\_trace\_clear()*, *posix\_trace\_close()*,

49662 *posix\_trace\_eventid\_equal()*, *posix\_trace\_eventtypelist\_getnext\_id()*, *posix\_trace\_get\_attr()*,

49663 *posix\_trace\_get\_filter()*, *posix\_trace\_getnext\_event()*, *posix\_trace\_start()*, *posix\_trace\_start()*, *time()*

49664 XBD <sys/types.h>, <trace.h>

#### 49665 CHANGE HISTORY

49666 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

#### 49667 Issue 7

49668 These functions are marked obsolescent.

49669 SD5-XSH-ERN-154 is applied, updating the DESCRIPTION to remove the

49670 *posix\_trace\_trygetnext\_event()* function from the list of functions that use the *trid* argument.

49671 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0441 [358] is applied.



**NAME**

posix\_trace\_event, posix\_trace\_eventid\_open — trace functions for instrumenting application code (**TRACING**)

**SYNOPSIS**

```
OB TRC #include <sys/types.h>
#include <trace.h>

void posix_trace_event(trace_event_id_t event_id,
    const void *restrict data_ptr, size_t data_len);
int posix_trace_eventid_open(const char *restrict event_name,
    trace_event_id_t *restrict event_id);
```

**DESCRIPTION**

The *posix\_trace\_event()* function shall record the *event\_id* and the user data pointed to by *data\_ptr* in the trace stream into which the calling process is being traced and in which *event\_id* is not filtered out. If the total size of the user trace event data represented by *data\_len* is not greater than the declared maximum size for user trace event data, then the *truncation-status* attribute of the trace event recorded is POSIX\_TRACE\_NOT\_TRUNCATED. Otherwise, the user trace event data is truncated to this declared maximum size and the *truncation-status* attribute of the trace event recorded is POSIX\_TRACE\_TRUNCATED\_RECORD.

If there is no trace stream created for the process or if the created trace stream is not running, or if the trace event specified by *event\_id* is filtered out in the trace stream, the *posix\_trace\_event()* function shall have no effect.

The *posix\_trace\_eventid\_open()* function shall associate a user trace event name with a trace event type identifier for the calling process. The trace event name is the string pointed to by the argument *event\_name*. It shall have a maximum of {TRACE\_EVENT\_NAME\_MAX} characters (which has the minimum value {POSIX\_TRACE\_EVENT\_NAME\_MAX}). The number of user trace event type identifiers that can be defined for any given process is limited by the maximum value {TRACE\_USER\_EVENT\_MAX}, which has the minimum value {POSIX\_TRACE\_USER\_EVENT\_MAX}.

If the Trace Inherit option is not supported, the *posix\_trace\_eventid\_open()* function shall associate the user trace event name pointed to by the *event\_name* argument with a trace event type identifier that is unique for the traced process, and is returned in the variable pointed to by the *event\_id* argument. If the user trace event name has already been mapped for the traced process, then the previously assigned trace event type identifier shall be returned. If the per-process user trace event name limit represented by {TRACE\_USER\_EVENT\_MAX} has been reached, the pre-defined POSIX\_TRACE\_UNNAMED\_USEREVENT (see Table 2-7, on page 546) user trace event shall be returned.

**TRI**

If the Trace Inherit option is supported, the *posix\_trace\_eventid\_open()* function shall associate the user trace event name pointed to by the *event\_name* argument with a trace event type identifier that is unique for all the processes being traced in this same trace stream, and is returned in the variable pointed to by the *event\_id* argument. If the user trace event name has already been mapped for the traced processes, then the previously assigned trace event type identifier shall be returned. If the per-process user trace event name limit represented by {TRACE\_USER\_EVENT\_MAX} has been reached, the pre-defined POSIX\_TRACE\_UNNAMED\_USEREVENT (Table 2-7, on page 546) user trace event shall be returned.

**Note:** The above procedure, together with the fact that multiple processes can only be traced into the same trace stream by inheritance, ensure that all the processes that are traced into a trace stream have the same mapping of trace event names to trace event type identifiers.

49720 If there is no trace stream created, the *posix\_trace\_eventid\_open()* function shall store this  
 49721 information for future trace streams created for this process.

#### 49722 RETURN VALUE

49723 No return value is defined for the *posix\_trace\_event()* function.

49724 Upon successful completion, the *posix\_trace\_eventid\_open()* function shall return a value of zero.  
 49725 Otherwise, it shall return the corresponding error number. The *posix\_trace\_eventid\_open()*  
 49726 function stores the trace event type identifier value in the object pointed to by *event\_id*, if  
 49727 successful.

#### 49728 ERRORS

49729 The *posix\_trace\_eventid\_open()* function shall fail if:

49730 [ENAMETOOLONG]

49731 The size of the name pointed to by the *event\_name* argument was longer than  
 49732 the implementation-defined value {TRACE\_EVENT\_NAME\_MAX}.

#### 49733 EXAMPLES

49734 None.

#### 49735 APPLICATION USAGE

49736 None.

#### 49737 RATIONALE

49738 None.

#### 49739 FUTURE DIRECTIONS

49740 The *posix\_trace\_event()* and *posix\_trace\_eventid\_open()* functions may be removed in a future  
 49741 version.

#### 49742 SEE ALSO

49743 [Table 2-7](#) (on page 546), *exec*, *posix\_trace\_eventid\_equal()*, *posix\_trace\_start()*

49744 XBD [<sys/types.h>](#), [<trace.h>](#)

#### 49745 CHANGE HISTORY

49746 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

49747 IEEE PASC Interpretation 1003.1 #123 is applied.

49748 IEEE PASC Interpretation 1003.1 #127 is applied, correcting some editorial errors in the names of  
 49749 the *posix\_trace\_eventid\_open()* function and the *event\_id* argument.

#### 49750 Issue 7

49751 The *posix\_trace\_event()* and *posix\_trace\_eventid\_open()* functions are marked obsolescent.

**NAME**

posix\_trace\_eventid\_equal, posix\_trace\_eventid\_get\_name, posix\_trace\_trid\_eventid\_open — manipulate the trace event type identifier (TRACING)

**SYNOPSIS**

```
#include <trace.h>

int posix_trace_eventid_equal(trace_id_t trid, trace_event_id_t event1,
    trace_event_id_t event2);
int posix_trace_eventid_get_name(trace_id_t trid,
    trace_event_id_t event, char *event_name);
int posix_trace_trid_eventid_open(trace_id_t trid,
    const char *restrict event_name,
    trace_event_id_t *restrict event);
```

**DESCRIPTION**

The *posix\_trace\_eventid\_equal()* function shall compare the trace event type identifiers *event1* and *event2* from the same trace stream or the same trace log identified by the *trid* argument. If the trace event type identifiers *event1* and *event2* are from different trace streams, the return value shall be unspecified.

The *posix\_trace\_eventid\_get\_name()* function shall return, in the argument pointed to by *event\_name*, the trace event name associated with the trace event type identifier identified by the argument *event*, for the trace stream or for the trace log identified by the *trid* argument. The name of the trace event shall have a maximum of {TRACE\_EVENT\_NAME\_MAX} characters (which has the minimum value {\_POSIX\_TRACE\_EVENT\_NAME\_MAX}). Successive calls to this function with the same trace event type identifier and the same trace stream identifier shall return the same event name.

The *posix\_trace\_trid\_eventid\_open()* function shall associate a user trace event name with a trace event type identifier for a given trace stream. The trace stream is identified by the *trid* argument, and it shall be an active trace stream. The trace event name is the string pointed to by the argument *event\_name*. It shall have a maximum of {TRACE\_EVENT\_NAME\_MAX} characters (which has the minimum value {\_POSIX\_TRACE\_EVENT\_NAME\_MAX}). The number of user trace event type identifiers that can be defined for any given process is limited by the maximum value {TRACE\_USER\_EVENT\_MAX}, which has the minimum value {\_POSIX\_TRACE\_USER\_EVENT\_MAX}.

If the Trace Inherit option is not supported, the *posix\_trace\_trid\_eventid\_open()* function shall associate the user trace event name pointed to by the *event\_name* argument with a trace event type identifier that is unique for the process being traced in the trace stream identified by the *trid* argument, and is returned in the variable pointed to by the *event* argument. If the user trace event name has already been mapped for the traced process, then the previously assigned trace event type identifier shall be returned. If the per-process user trace event name limit represented by {TRACE\_USER\_EVENT\_MAX} has been reached, the pre-defined POSIX\_TRACE\_UNNAMED\_USEREVENT (see Table 2-7, on page 546) user trace event shall be returned.

If the Trace Inherit option is supported, the *posix\_trace\_trid\_eventid\_open()* function shall associate the user trace event name pointed to by the *event\_name* argument with a trace event type identifier that is unique for all the processes being traced in the trace stream identified by the *trid* argument, and is returned in the variable pointed to by the *event* argument. If the user trace event name has already been mapped for the traced processes, then the previously assigned trace event type identifier shall be returned. If the per-process user trace event name limit represented by {TRACE\_USER\_EVENT\_MAX} has been reached, the pre-defined

49800 POSIX\_TRACE\_UNNAMED\_USEREVENT (see Table 2-7, on page 546) user trace event shall be  
49801 returned.

## 49802 RETURN VALUE

49803 TEF Upon successful completion, the `posix_trace_eventid_get_name()` and  
49804 `posix_trace_trid_eventid_open()` functions shall return a value of zero. Otherwise, they shall return  
49805 the corresponding error number.

49806 The `posix_trace_eventid_equal()` function shall return a non-zero value if *event1* and *event2* are  
49807 equal; otherwise, a value of zero shall be returned. No errors are defined. If either *event1* or  
49808 *event2* are not valid trace event type identifiers for the trace stream specified by *trid* or if the *trid*  
49809 is invalid, the behavior shall be unspecified.

49810 The `posix_trace_eventid_get_name()` function stores the trace event name value in the object  
49811 pointed to by *event\_name*, if successful.

49812 TEF The `posix_trace_trid_eventid_open()` function stores the trace event type identifier value in the  
49813 object pointed to by *event*, if successful.

## 49814 ERRORS

49815 TEF The `posix_trace_eventid_get_name()` and `posix_trace_trid_eventid_open()` functions shall fail if:

49816 [EINVAL] The *trid* argument was not a valid trace stream identifier.

49817 TEF The `posix_trace_trid_eventid_open()` function shall fail if:

49818 TEF [ENAMETOOLONG]

49819 The size of the name pointed to by the *event\_name* argument was longer than  
49820 the implementation-defined value {TRACE\_EVENT\_NAME\_MAX}.

49821 The `posix_trace_eventid_get_name()` function shall fail if:

49822 [EINVAL] The trace event type identifier *event* was not associated with any name.

## 49823 EXAMPLES

49824 None.

## 49825 APPLICATION USAGE

49826 None.

## 49827 RATIONALE

49828 None.

## 49829 FUTURE DIRECTIONS

49830 The `posix_trace_eventid_equal()`, `posix_trace_eventid_get_name()`, and  
49831 `posix_trace_trid_eventid_open()` functions may be removed in a future version.

## 49832 SEE ALSO

49833 Table 2-7 (on page 546), `exec`, `posix_trace_event()`, `posix_trace_getnext_event()`

49834 XBD <trace.h>

## 49835 CHANGE HISTORY

49836 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

49837 IEEE PASC Interpretations 1003.1 #123 and #129 are applied.

## 49838 Issue 7

49839 These functions are marked obsolescent.

49840 **NAME**49841       posix\_trace\_eventid\_open — trace functions for instrumenting application code (**TRACING**)49842 **SYNOPSIS**

```
49843 OB TRC #include <sys/types.h>
49844         #include <trace.h>
49845         int posix_trace_eventid_open(const char *restrict event_name,
49846                                     trace_event_id_t *restrict event_id);
```

49847 **DESCRIPTION**49848       Refer to *posix\_trace\_event()*.

## NAME

posix\_trace\_eventset\_add, posix\_trace\_eventset\_del, posix\_trace\_eventset\_empty,  
 posix\_trace\_eventset\_fill, posix\_trace\_eventset\_ismember — manipulate trace event type sets  
 (TRACING)

## SYNOPSIS

```
#include <trace.h>

int posix_trace_eventset_add(trace_event_id_t event_id,
    trace_event_set_t *set);
int posix_trace_eventset_del(trace_event_id_t event_id,
    trace_event_set_t *set);
int posix_trace_eventset_empty(trace_event_set_t *set);
int posix_trace_eventset_fill(trace_event_set_t *set, int what);
int posix_trace_eventset_ismember(trace_event_id_t event_id,
    const trace_event_set_t *restrict set, int *restrict ismember);
```

## DESCRIPTION

These primitives manipulate sets of trace event types. They operate on data objects addressable by the application, not on the current trace event filter of any trace stream.

The *posix\_trace\_eventset\_add()* and *posix\_trace\_eventset\_del()* functions, respectively, shall add or delete the individual trace event type specified by the value of the argument *event\_id* to or from the trace event type set pointed to by the argument *set*. Adding a trace event type already in the set or deleting a trace event type not in the set shall not be considered an error.

The *posix\_trace\_eventset\_empty()* function shall initialize the trace event type set pointed to by the *set* argument such that all trace event types defined, both system and user, shall be excluded from the set.

The *posix\_trace\_eventset\_fill()* function shall initialize the trace event type set pointed to by the argument *set*, such that the set of trace event types defined by the argument *what* shall be included in the set. The value of the argument *what* shall consist of one of the following values, as defined in the **<trace.h>** header:

## POSIX\_TRACE\_WOPID\_EVENTS

All the process-independent implementation-defined system trace event types are included in the set.

## POSIX\_TRACE\_SYSTEM\_EVENTS

All the implementation-defined system trace event types are included in the set, as are those defined in POSIX.1-2008.

## POSIX\_TRACE\_ALL\_EVENTS

All trace event types defined, both system and user, are included in the set.

Applications shall call either *posix\_trace\_eventset\_empty()* or *posix\_trace\_eventset\_fill()* at least once for each object of type **trace\_event\_set\_t** prior to any other use of that object. If such an object is not initialized in this way, but is nonetheless supplied as an argument to any of the *posix\_trace\_eventset\_add()*, *posix\_trace\_eventset\_del()*, or *posix\_trace\_eventset\_ismember()* functions, the results are undefined.

The *posix\_trace\_eventset\_ismember()* function shall test whether the trace event type specified by the value of the argument *event\_id* is a member of the set pointed to by the argument *set*. The value returned in the object pointed to by *ismember* argument is zero if the trace event type identifier is not a member of the set and a value different from zero if it is a member of the set.

**49894 RETURN VALUE**

49895       Upon successful completion, these functions shall return a value of zero. Otherwise, they shall  
49896       return the corresponding error number.

**49897 ERRORS**

49898       These functions may fail if:

49899       [EINVAL]       The value of one of the arguments is invalid.

**49900 EXAMPLES**

49901       None.

**49902 APPLICATION USAGE**

49903       None.

**49904 RATIONALE**

49905       None.

**49906 FUTURE DIRECTIONS**

49907       The `posix_trace_eventset_add()`, `posix_trace_eventset_del()`, `posix_trace_eventset_empty()`,  
49908       `posix_trace_eventset_fill()`, and `posix_trace_eventset_ismember()` functions may be removed in a  
49909       future version.

**49910 SEE ALSO**

49911       [\*posix\\_trace\\_eventid\\_equal\(\)\*](#), [\*posix\\_trace\\_get\\_filter\(\)\*](#)

49912       XBD [\*\*<trace.h>\*\*](#)

**49913 CHANGE HISTORY**

49914       First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

**49915 Issue 7**

49916       The `posix_trace_eventset_add()`, `posix_trace_eventset_del()`, `posix_trace_eventset_empty()`,  
49917       `posix_trace_eventset_fill()`, and `posix_trace_eventset_ismember()` functions are marked obsolescent.



49918 **NAME**

49919        posix\_trace\_eventtypelist\_getnext\_id, posix\_trace\_eventtypelist\_rewind — iterate over a  
 49920        mapping of trace event types (**TRACING**)

49921 **SYNOPSIS**

```
49922 OB TRC #include <trace.h>
49923
49924 int posix_trace_eventtypelist_getnext_id(trace_id_t trid,
49925     trace_event_id_t *restrict event, int *restrict unavailable);
49926 int posix_trace_eventtypelist_rewind(trace_id_t trid);
```

49926 **DESCRIPTION**

49927        The first time *posix\_trace\_eventtypelist\_getnext\_id()* is called, the function shall return in the  
 49928        variable pointed to by *event* the first trace event type identifier of the list of trace events of the  
 49929        trace stream identified by the *trid* argument. Successive calls to  
 49930        *posix\_trace\_eventtypelist\_getnext\_id()* return in the variable pointed to by *event* the next trace  
 49931        event type identifier in that same list. Each time a trace event type identifier is successfully  
 49932        written into the variable pointed to by the *event* argument, the variable pointed to by the  
 49933        *unavailable* argument shall be set to zero. When no more trace event type identifiers are available,  
 49934        and so none is returned, the variable pointed to by the *unavailable* argument shall be set to a  
 49935        value different from zero.

49936        The *posix\_trace\_eventtypelist\_rewind()* function shall reset the next trace event type identifier to  
 49937        be read to the first trace event type identifier from the list of trace events used in the trace stream  
 49938        identified by *trid*.

49939 **RETURN VALUE**

49940        Upon successful completion, these functions shall return a value of zero. Otherwise, they shall  
 49941        return the corresponding error number.

49942        The *posix\_trace\_eventtypelist\_getnext\_id()* function stores the trace event type identifier value in  
 49943        the object pointed to by *event*, if successful.

49944 **ERRORS**

49945        These functions shall fail if:

49946        [EINVAL]        The *trid* argument was not a valid trace stream identifier.

49947 **EXAMPLES**

49948        None.

49949 **APPLICATION USAGE**

49950        None.

49951 **RATIONALE**

49952        None.

49953 **FUTURE DIRECTIONS**

49954        The *posix\_trace\_eventtypelist\_getnext\_id()* and *posix\_trace\_eventtypelist\_rewind()* functions may be  
 49955        removed in a future version.

49956 **SEE ALSO**

49957        *posix\_trace\_event()*, *posix\_trace\_eventid\_equal()*, *posix\_trace\_getnext\_event()*

49958        XBD *<trace.h>*



**CHANGE HISTORY**

49959 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

49960 IEEE PASC Interpretations 1003.1 #123 and #129 are applied.

**Issue 7**

49962 The *posix\_trace\_eventtypelist\_getnext\_id()* and *posix\_trace\_eventtypelist\_rewind()* functions are  
49963 marked obsolescent.  
49964

49965 **NAME**49966       posix\_trace\_flush — trace stream flush from a process (**TRACING**)49967 **SYNOPSIS**

```
49968 OB TRC  #include <sys/types.h>
49969          #include <trace.h>
49970 TRL      int posix_trace_flush(trace_id_t trid);
```

49971 **DESCRIPTION**49972       Refer to *posix\_trace\_create()*.

49973 **NAME**

49974        `posix_trace_get_attr`, `posix_trace_get_status` — retrieve the trace attributes or trace status  
 49975        (**TRACING**)

49976 **SYNOPSIS**

```
49977 OB TRC #include <trace.h>
49978
49978     int posix_trace_get_attr(trace_id_t trid, trace_attr_t *attr);
49979     int posix_trace_get_status(trace_id_t trid,
49980                               struct posix_trace_status_info *statusinfo);
```

49981 **DESCRIPTION**

49982        The `posix_trace_get_attr()` function shall copy the attributes of the active trace stream identified  
 49983        TRL by `trid` into the object pointed to by the `attr` argument. If the Trace Log option is supported, `trid`  
 49984        may represent a pre-recorded trace log.

49985        The `posix_trace_get_status()` function shall return, in the structure pointed to by the `statusinfo`  
 49986        argument, the current trace status for the trace stream identified by the `trid` argument. These  
 49987        status values returned in the structure pointed to by `statusinfo` shall have been appropriately  
 49988        TRL read to ensure that the returned values are consistent. If the Trace Log option is supported and  
 49989        the `trid` argument refers to a pre-recorded trace stream, the status shall be the status of the  
 49990        completed trace stream.

49991        Each time the `posix_trace_get_status()` function is used, the overrun status of the trace stream  
 49992        TRL shall be reset to `POSIX_TRACE_NO_OVERRUN` immediately after the call completes. If the  
 49993        Trace Log option is supported, the `posix_trace_get_status()` function shall behave the same as  
 49994        when the option is not supported except for the following differences:

- 49995        • If the `trid` argument refers to a trace stream with log, each time the `posix_trace_get_status()`  
 49996        function is used, the log overrun status of the trace stream shall be reset to  
 49997        `POSIX_TRACE_NO_OVERRUN` and the `flush_error` status shall be reset to zero  
 49998        immediately after the call completes.
- 49999        • If the `trid` argument refers to a pre-recorded trace stream, the status returned shall be the  
 50000        status of the completed trace stream and the status values of the trace stream shall not be  
 50001        reset.

50002 **RETURN VALUE**

50003        Upon successful completion, these functions shall return a value of zero. Otherwise, they shall  
 50004        return the corresponding error number.

50005        The `posix_trace_get_attr()` function stores the trace attributes in the object pointed to by `attr`, if  
 50006        successful.

50007        The `posix_trace_get_status()` function stores the trace status in the object pointed to by `statusinfo`,  
 50008        if successful.

50009 **ERRORS**

50010        These functions shall fail if:

- 50011        [EINVAL]        The trace stream argument `trid` does not correspond to a valid active trace  
 50012        stream or a valid trace log.

50013 **EXAMPLES**

50014       None.

50015 **APPLICATION USAGE**

50016       None.

50017 **RATIONALE**

50018       None.

50019 **FUTURE DIRECTIONS**50020       The *posix\_trace\_get\_attr()* and *posix\_trace\_get\_status()* functions may be removed in a future  
50021       version.50022 **SEE ALSO**50023       *posix\_trace\_attr\_destroy()*, *posix\_trace\_close()*, *posix\_trace\_create()*

50024       XBD &lt;trace.h&gt;

50025 **CHANGE HISTORY**

50026       First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

50027       IEEE PASC Interpretation 1003.1 #123 is applied.

50028 **Issue 7**50029       The *posix\_trace\_get\_attr()* and *posix\_trace\_get\_status()* functions are marked obsolescent.

50030 **NAME**

50031        `posix_trace_get_filter`, `posix_trace_set_filter` — retrieve and set the filter of an initialized trace  
 50032        stream (**TRACING**)

50033 **SYNOPSIS**

```
50034 OB TRC  #include <trace.h>

50035 TEF      int posix_trace_get_filter(trace_id_t trid, trace_event_set_t *set);
50036          int posix_trace_set_filter(trace_id_t trid,
50037          const trace_event_set_t *set, int how);
```

50038 **DESCRIPTION**

50039        The `posix_trace_get_filter()` function shall retrieve, into the argument pointed to by `set`, the actual  
 50040        trace event filter from the trace stream specified by `trid`.

50041        The `posix_trace_set_filter()` function shall change the set of filtered trace event types after a trace  
 50042        stream identified by the `trid` argument is created. This function may be called prior to starting  
 50043        the trace stream, or while the trace stream is active. By default, if no call is made to  
 50044        `posix_trace_set_filter()`, all trace events shall be recorded (that is, none of the trace event types are  
 50045        filtered out).

50046        If this function is called while the trace is in progress, a special system trace event,  
 50047        `POSIX_TRACE_FILTER`, shall be recorded in the trace indicating both the old and the new sets  
 50048        of filtered trace event types (see [Table 2-4](#) (on page 544) and [Table 2-6](#), on page 545).

50049        If the `posix_trace_set_filter()` function is interrupted by a signal, an error shall be returned and the  
 50050        filter shall not be changed. In this case, the state of the trace stream shall not be changed.

50051        The value of the argument `how` indicates the manner in which the set is to be changed and shall  
 50052        have one of the following values, as defined in the `<trace.h>` header:

50053 **POSIX\_TRACE\_SET\_EVENTSET**

50054        The resulting set of trace event types to be filtered shall be the trace event type set pointed  
 50055        to by the argument `set`.

50056 **POSIX\_TRACE\_ADD\_EVENTSET**

50057        The resulting set of trace event types to be filtered shall be the union of the current set and  
 50058        the trace event type set pointed to by the argument `set`.

50059 **POSIX\_TRACE\_SUB\_EVENTSET**

50060        The resulting set of trace event types to be filtered shall be all trace event types in the  
 50061        current set that are not in the set pointed to by the argument `set`; that is, remove each  
 50062        element of the specified set from the current filter.

50063 **RETURN VALUE**

50064        Upon successful completion, these functions shall return a value of zero. Otherwise, they shall  
 50065        return the corresponding error number.

50066        The `posix_trace_get_filter()` function stores the set of filtered trace event types in `set`, if successful.

50067 **ERRORS**

50068        These functions shall fail if:

50069        [EINVAL]        The value of the `trid` argument does not correspond to an active trace stream  
 50070        or the value of the argument pointed to by `set` is invalid.

50071        [EINTR]        The operation was interrupted by a signal.

50072 **EXAMPLES**

50073 None.

50074 **APPLICATION USAGE**

50075 None.

50076 **RATIONALE**

50077 None.

50078 **FUTURE DIRECTIONS**

50079 The *posix\_trace\_get\_filter()* and *posix\_trace\_set\_filter()* functions may be removed in a future  
50080 version.

50081 **SEE ALSO**50082 [Table 2-4](#) (on page 544), [Table 2-6](#) (on page 545), [posix\\_trace\\_eventset\\_add\(\)](#)50083 XBD [<trace.h>](#)50084 **CHANGE HISTORY**

50085 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

50086 IEEE PASC Interpretation 1003.1 #123 is applied.

50087 **Issue 7**50088 The *posix\_trace\_get\_filter()* and *posix\_trace\_set\_filter()* functions are marked obsolescent.

50089 **NAME**

50090       posix\_trace\_get\_status — retrieve the trace status (TRACING)

50091 **SYNOPSIS**

50092 OB TRC   #include &lt;trace.h&gt;

```
50093       int posix_trace_get_status(trace_id_t trid,  
50094           struct posix_trace_status_info *statusinfo);
```

50095 **DESCRIPTION**50096       Refer to *posix\_trace\_get\_attr()*.

## 50097 NAME

50098 `posix_trace_getnext_event`, `posix_trace_timedgetnext_event`, `posix_trace_trygetnext_event` —  
 50099 retrieve a trace event (**TRACING**)

## 50100 SYNOPSIS

```
50101 OB TRC #include <sys/types.h>
50102         #include <trace.h>

50103         int posix_trace_getnext_event(trace_id_t trid,
50104             struct posix_trace_event_info *restrict event,
50105             void *restrict data, size_t num_bytes,
50106             size_t *restrict data_len, int *restrict unavailable);
50107         int posix_trace_timedgetnext_event(trace_id_t trid,
50108             struct posix_trace_event_info *restrict event,
50109             void *restrict data, size_t num_bytes,
50110             size_t *restrict data_len, int *restrict unavailable,
50111             const struct timespec *restrict abstime);
50112         int posix_trace_trygetnext_event(trace_id_t trid,
50113             struct posix_trace_event_info *restrict event,
50114             void *restrict data, size_t num_bytes,
50115             size_t *restrict data_len, int *restrict unavailable);
```

## 50116 DESCRIPTION

50117 The `posix_trace_getnext_event()` function shall report a recorded trace event either from an active  
 50118 TRL trace stream without log or a pre-recorded trace stream identified by the *trid* argument. The  
 50119 `posix_trace_trygetnext_event()` function shall report a recorded trace event from an active trace  
 50120 stream without log identified by the *trid* argument.

50121 The trace event information associated with the recorded trace event shall be copied by the  
 50122 function into the structure pointed to by the argument *event* and the data associated with the  
 50123 trace event shall be copied into the buffer pointed to by the *data* argument.

50124 The `posix_trace_getnext_event()` function shall block if the *trid* argument identifies an active trace  
 50125 stream and there is currently no trace event ready to be retrieved. When returning, if a recorded  
 50126 trace event was reported, the variable pointed to by the *unavailable* argument shall be set to zero.  
 50127 Otherwise, the variable pointed to by the *unavailable* argument shall be set to a value different  
 50128 from zero.

50129 The `posix_trace_timedgetnext_event()` function shall attempt to get another trace event from an  
 50130 active trace stream without log, as in the `posix_trace_getnext_event()` function. However, if no  
 50131 trace event is available from the trace stream, the implied wait shall be terminated when the  
 50132 timeout specified by the argument *abstime* expires, and the function shall return the error  
 50133 [ETIMEDOUT].

50134 The timeout shall expire when the absolute time specified by *abstime* passes, as measured by the  
 50135 clock upon which timeouts are based (that is, when the value of that clock equals or exceeds  
 50136 *abstime*), or if the absolute time specified by *abstime* has already passed at the time of the call.

50137 The timeout shall be based on the CLOCK\_REALTIME clock. The resolution of the timeout shall  
 50138 be the resolution of the clock on which it is based. The **timespec** data type is defined in the  
 50139 **<time.h>** header.

50140 Under no circumstance shall the function fail with a timeout if a trace event is immediately  
 50141 available from the trace stream. The validity of the *abstime* argument need not be checked if a  
 50142 trace event is immediately available from the trace stream.



50143 The behavior of this function for a pre-recorded trace stream is unspecified.

50144 TRL The *posix\_trace\_trygetnext\_event()* function shall not block. This function shall return an error if  
50145 the *trid* argument identifies a pre-recorded trace stream. If a recorded trace event was reported,  
50146 the variable pointed to by the *unavailable* argument shall be set to zero. Otherwise, if no trace  
50147 event was reported, the variable pointed to by the *unavailable* argument shall be set to a value  
50148 different from zero.

50149 The argument *num\_bytes* shall be the size of the buffer pointed to by the *data* argument. The  
50150 argument *data\_len* reports to the application the length in bytes of the data record just  
50151 transferred. If *num\_bytes* is greater than or equal to the size of the data associated with the trace  
50152 event pointed to by the *event* argument, all the recorded data shall be transferred. In this case,  
50153 the *truncation-status* member of the trace event structure shall be either  
50154 POSIX\_TRACE\_NOT\_TRUNCATED, if the trace event data was recorded without truncation  
50155 while tracing, or POSIX\_TRACE\_TRUNCATED\_RECORD, if the trace event data was truncated  
50156 when it was recorded. If the *num\_bytes* argument is less than the length of recorded trace event  
50157 data, the data transferred shall be truncated to a length of *num\_bytes*, the value stored in the  
50158 variable pointed to by *data\_len* shall be equal to *num\_bytes*, and the *truncation-status* member of  
50159 the *event* structure argument shall be set to POSIX\_TRACE\_TRUNCATED\_READ (see the  
50160 *posix\_trace\_event\_info* structure defined in <trace.h>).

50161 The report of a trace event shall be sequential starting from the oldest recorded trace event. Trace  
50162 events shall be reported in the order in which they were generated, up to an implementation-  
50163 defined time resolution that causes the ordering of trace events occurring very close to each  
50164 other to be unknown. Once reported, a trace event cannot be reported again from an active trace  
50165 stream. Once a trace event is reported from an active trace stream without log, the trace stream  
50166 shall make the resources associated with that trace event available to record future generated  
50167 trace events.

#### 50168 RETURN VALUE

50169 Upon successful completion, these functions shall return a value of zero. Otherwise, they shall  
50170 return the corresponding error number.

50171 If successful, these functions store:

- 50172 • The recorded trace event in the object pointed to by *event*
- 50173 • The trace event information associated with the recorded trace event in the object pointed  
50174 to by *data*
- 50175 • The length of this trace event information in the object pointed to by *data\_len*
- 50176 • The value of zero in the object pointed to by *unavailable*

#### 50177 ERRORS

50178 These functions shall fail if:

50179 [EINVAL] The trace stream identifier argument *trid* is invalid.

50180 The *posix\_trace\_getnext\_event()* and *posix\_trace\_timedgetnext\_event()* functions shall fail if:

50181 [EINTR] The operation was interrupted by a signal, and so the call had no effect.

50182 The *posix\_trace\_trygetnext\_event()* function shall fail if:

50183 [EINVAL] The trace stream identifier argument *trid* does not correspond to an active  
50184 trace stream.

50185 The *posix\_trace\_timedgetnext\_event()* function shall fail if:  
50186 [EINVAL] There is no trace event immediately available from the trace stream, and the  
50187 *timeout* argument is invalid.  
50188 [ETIMEDOUT] No trace event was available from the trace stream before the specified  
50189 timeout *timeout* expired.

50190 **EXAMPLES**

50191 None.

50192 **APPLICATION USAGE**

50193 None.

50194 **RATIONALE**

50195 None.

50196 **FUTURE DIRECTIONS**

50197 These functions may be removed in a future version.

50198 **SEE ALSO**

50199 *posix\_trace\_close()*, *posix\_trace\_create()*

50200 XBD <sys/types.h>, <trace.h>

50201 **CHANGE HISTORY**

50202 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

50203 IEEE PASC Interpretation 1003.1 #123 is applied.

50204 **Issue 7**

50205 The *posix\_trace\_getnext\_event()*, *posix\_trace\_timedgetnext\_event()*, and  
50206 *posix\_trace\_trygetnext\_event()* functions are marked obsolescent.

50207 Functionality relating to the Timers option is moved to the Base.

50208 **NAME**

50209       posix\_trace\_open, posix\_trace\_rewind — trace log management (TRACING)

50210 **SYNOPSIS**

50211 OB TRC   #include &lt;trace.h&gt;

50212 TRL       int posix\_trace\_open(int *file\_desc*, trace\_id\_t \*trid);

50213       int posix\_trace\_rewind(trace\_id\_t trid);

50214 **DESCRIPTION**50215       Refer to *posix\_trace\_close()*.

50216 **NAME**50217       posix\_trace\_set\_filter — set filter of an initialized trace stream (**TRACING**)50218 **SYNOPSIS**

50219 OB TRC   #include &lt;trace.h&gt;

50220 TEF       int posix\_trace\_set\_filter(trace\_id\_t trid,  
50221       const trace\_event\_set\_t \*set, int how);50222 **DESCRIPTION**50223       Refer to *posix\_trace\_get\_filter()*.

50224 **NAME**

50225       posix\_trace\_shutdown — trace stream shutdown from a process (TRACING)

50226 **SYNOPSIS**

```
50227 OB TRC #include <sys/types.h>  
50228         #include <trace.h>  
50229         int posix_trace_shutdown(trace_id_t trid);
```

50230 **DESCRIPTION**50231       Refer to *posix\_trace\_create()*.

50232 **NAME**

50233       posix\_trace\_start, posix\_trace\_stop — trace start and stop (TRACING)

50234 **SYNOPSIS**

```
50235 OB TRC  #include <trace.h>
50236         int posix_trace_start(trace_id_t trid);
50237         int posix_trace_stop (trace_id_t trid);
```

50238 **DESCRIPTION**

50239       The *posix\_trace\_start()* and *posix\_trace\_stop()* functions, respectively, shall start and stop the trace  
50240       stream identified by the argument *trid*.

50241       The effect of calling the *posix\_trace\_start()* function shall be recorded in the trace stream as the  
50242       POSIX\_TRACE\_START system trace event and the status of the trace stream shall become  
50243       POSIX\_TRACE\_RUNNING. If the trace stream is in progress when this function is called, the  
50244       POSIX\_TRACE\_START system trace event shall not be recorded and the trace stream shall  
50245       continue to run. If the trace stream is full, the POSIX\_TRACE\_START system trace event shall  
50246       not be recorded and the status of the trace stream shall not be changed.

50247       The effect of calling the *posix\_trace\_stop()* function shall be recorded in the trace stream as the  
50248       POSIX\_TRACE\_STOP system trace event and the status of the trace stream shall become  
50249       POSIX\_TRACE\_SUSPENDED. If the trace stream is suspended when this function is called, the  
50250       POSIX\_TRACE\_STOP system trace event shall not be recorded and the trace stream shall remain  
50251       suspended. If the trace stream is full, the POSIX\_TRACE\_STOP system trace event shall not be  
50252       recorded and the status of the trace stream shall not be changed.

50253 **RETURN VALUE**

50254       Upon successful completion, these functions shall return a value of zero. Otherwise, they shall  
50255       return the corresponding error number.

50256 **ERRORS**

50257       These functions shall fail if:

50258       [EINVAL]       The value of the argument *trid* does not correspond to an active trace stream  
50259       and thus no trace stream was started or stopped.

50260       [EINTR]       The operation was interrupted by a signal and thus the trace stream was not  
50261       necessarily started or stopped.

50262 **EXAMPLES**

50263       None.

50264 **APPLICATION USAGE**

50265       None.

50266 **RATIONALE**

50267       None.

50268 **FUTURE DIRECTIONS**

50269       The *posix\_trace\_start()* and *posix\_trace\_stop()* functions may be removed in a future version.

50270 **SEE ALSO**

50271       *posix\_trace\_create()*

50272       XBD *<trace.h>*

50273 **CHANGE HISTORY**

50274 First released in Issue 6. Derived from IEEE Std 1003.1q-2000.

50275 IEEE PASC Interpretation 1003.1 #123 is applied.

50276 **Issue 7**

50277 The *posix\_trace\_start()* and *posix\_trace\_stop()* functions are marked obsolescent.

50278 **NAME**50279       posix\_trace\_timedgetnext\_event — retrieve a trace event (**TRACING**)50280 **SYNOPSIS**

```
50281 OB TRC #include <sys/types.h>
50282         #include <trace.h>

50283         int posix_trace_timedgetnext_event(trace_id_t trid,
50284         struct posix_trace_event_info *restrict event,
50285         void *restrict data, size_t num_bytes,
50286         size_t *restrict data_len, int *restrict unavailable,
50287         const struct timespec *restrict abstime);
```

50288 **DESCRIPTION**50289       Refer to *posix\_trace\_getnext\_event()*.



50290 **NAME**50291       posix\_trace\_trid\_eventid\_open — open a trace event type identifier (**TRACING**)50292 **SYNOPSIS**

```
50293 OB TRC  #include <trace.h>
50294 TEF      int posix_trace_trid_eventid_open(trace_id_t trid,
50295      const char *restrict event_name,
50296      trace_event_id_t *restrict event);
```

50297 **DESCRIPTION**50298       Refer to *posix\_trace\_eventid\_equal()*.

50299 **NAME**50300        posix\_trace\_trygetnext\_event — retrieve a trace event (**TRACING**)50301 **SYNOPSIS**

```
50302 OB TRC #include <sys/types.h>
50303         #include <trace.h>
50304
50304         int posix_trace_trygetnext_event(trace_id_t trid,
50305         struct posix_trace_event_info *restrict event,
50306         void *restrict data, size_t num_bytes,
50307         size_t *restrict data_len, int *restrict unavailable);
```

50308 **DESCRIPTION**50309        Refer to *posix\_trace\_getnext\_event()*.

**50310 NAME**

50311 `posix_typed_mem_get_info` — query typed memory information (**ADVANCED REALTIME**)

**50312 SYNOPSIS**

```
50313 TYM      #include <sys/mman.h>
50314
50314      int posix_typed_mem_get_info(int fildes,
50315                                  struct posix_typed_mem_info *info);
```

**50316 DESCRIPTION**

50317 The `posix_typed_mem_get_info()` function shall return, in the `posix_tmi_length` field of the  
 50318 **posix\_typed\_mem\_info** structure pointed to by *info*, the maximum length which may be  
 50319 successfully allocated by the typed memory object designated by *fildes*. This maximum length  
 50320 shall take into account the flag `POSIX_TYPED_MEM_ALLOCATE` or  
 50321 `POSIX_TYPED_MEM_ALLOCATE_CONTIG` specified when the typed memory object  
 50322 represented by *fildes* was opened. The maximum length is dynamic; therefore, the value  
 50323 returned is valid only while the current mapping of the corresponding typed memory pool  
 50324 remains unchanged.

50325 If *fildes* represents a typed memory object opened with neither the  
 50326 `POSIX_TYPED_MEM_ALLOCATE` flag nor the `POSIX_TYPED_MEM_ALLOCATE_CONTIG`  
 50327 flag specified, the returned value of *info*->`posix_tmi_length` is unspecified.

50328 The `posix_typed_mem_get_info()` function may return additional implementation-defined  
 50329 information in other fields of the **posix\_typed\_mem\_info** structure pointed to by *info*.

50330 If the memory object specified by *fildes* is not a typed memory object, then the behavior of this  
 50331 function is undefined.

**50332 RETURN VALUE**

50333 Upon successful completion, the `posix_typed_mem_get_info()` function shall return zero;  
 50334 otherwise, the corresponding error status value shall be returned.

**50335 ERRORS**

50336 The `posix_typed_mem_get_info()` function shall fail if:

|       |          |                                                                                  |
|-------|----------|----------------------------------------------------------------------------------|
| 50337 | [EBADF]  | The <i>fildes</i> argument is not a valid open file descriptor.                  |
| 50338 | [ENODEV] | The <i>fildes</i> argument is not connected to a memory object supported by this |
| 50339 |          | function.                                                                        |

50340 This function shall not return an error code of [EINTR].

**50341 EXAMPLES**

50342 None.

**50343 APPLICATION USAGE**

50344 None.

**50345 RATIONALE**

50346 An application that needs to allocate a block of typed memory with length dependent upon the  
 50347 amount of memory currently available must either query the typed memory object to obtain the  
 50348 amount available, or repeatedly invoke `mmap()` attempting to guess an appropriate length.  
 50349 While the latter method is existing practice with `malloc()`, it is awkward and imprecise. The  
 50350 `posix_typed_mem_get_info()` function allows an application to immediately determine available  
 50351 memory. This is particularly important for typed memory objects that may in some cases be  
 50352 scarce resources. Note that when a typed memory pool is a shared resource, some form of  
 50353 mutual-exclusion or synchronization may be required while typed memory is being queried and

50354 allocated to prevent race conditions.

50355 The existing *fstat()* function is not suitable for this purpose. We realize that implementations  
50356 may wish to provide other attributes of typed memory objects (for example, alignment  
50357 requirements, page size, and so on). The *fstat()* function returns a structure which is not  
50358 extensible and, furthermore, contains substantial information that is inappropriate for typed  
50359 memory objects.

50360 **FUTURE DIRECTIONS**

50361 None.

50362 **SEE ALSO**

50363 *fstat()*, *mmap()*, *posix\_typed\_mem\_open()*

50364 XBD <sys/mman.h>

50365 **CHANGE HISTORY**

50366 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

**NAME**

posix\_typed\_mem\_open — open a typed memory object (**ADVANCED REALTIME**)

**SYNOPSIS**

```
#include <sys/mman.h>

int posix_typed_mem_open(const char *name, int oflag, int tflag);
```

**DESCRIPTION**

The `posix_typed_mem_open()` function shall establish a connection between the typed memory object specified by the string pointed to by *name* and a file descriptor. It shall create an open file description that refers to the typed memory object and a file descriptor that refers to that open file description. The file descriptor shall be allocated as described in [Section 2.14](#) (on page 549) and can be used by other functions to refer to that typed memory object. It is unspecified whether the name appears in the file system and is visible to other functions that take pathnames as arguments. The *name* argument conforms to the construction rules for a pathname, except that the interpretation of <slash> characters other than the leading <slash> character in *name* is implementation-defined, and that the length limits for the *name* argument are implementation-defined and need not be the same as the pathname limits {PATH\_MAX} and {NAME\_MAX}. If *name* begins with the <slash> character, then processes calling `posix_typed_mem_open()` with the same value of *name* shall refer to the same typed memory object. If *name* does not begin with the <slash> character, the effect is implementation-defined.

Each typed memory object supported in a system shall be identified by a name which specifies not only its associated typed memory pool, but also the path or port by which it is accessed. That is, the same typed memory pool accessed via several different ports shall have several different corresponding names. The binding between names and typed memory objects is established in an implementation-defined manner. Unlike shared memory objects, there is no way within POSIX.1-2008 for a program to create a typed memory object.

The value of *tflag* shall determine how the typed memory object behaves when subsequently mapped by calls to `mmap()`. At most, one of the following flags defined in `<sys/mman.h>` may be specified:

POSIX\_TYPED\_MEM\_ALLOCATE

Allocate on `mmap()`.

POSIX\_TYPED\_MEM\_ALLOCATE\_CONTIG

Allocate contiguously on `mmap()`.

POSIX\_TYPED\_MEM\_MAP\_ALLOCATABLE

Map on `mmap()`, without affecting allocatability.

If *tflag* has the flag POSIX\_TYPED\_MEM\_ALLOCATE specified, any subsequent call to `mmap()` using the returned file descriptor shall result in allocation and mapping of typed memory from the specified typed memory pool. The allocated memory may be a contiguous previously unallocated area of the typed memory pool or several non-contiguous previously unallocated areas (mapped to a contiguous portion of the process address space). If *tflag* has the flag POSIX\_TYPED\_MEM\_ALLOCATE\_CONTIG specified, any subsequent call to `mmap()` using the returned file descriptor shall result in allocation and mapping of a single contiguous previously unallocated area of the typed memory pool (also mapped to a contiguous portion of the process address space). If *tflag* has none of the flags POSIX\_TYPED\_MEM\_ALLOCATE or POSIX\_TYPED\_MEM\_ALLOCATE\_CONTIG specified, any subsequent call to `mmap()` using the returned file descriptor shall map an application-chosen area from the specified typed memory pool such that this mapped area becomes unavailable for allocation until unmapped by all processes. If *tflag* has the flag POSIX\_TYPED\_MEM\_MAP\_ALLOCATABLE specified, any

subsequent call to *mmap()* using the returned file descriptor shall map an application-chosen area from the specified typed memory pool without an effect on the availability of that area for allocation; that is, mapping such an object leaves each byte of the mapped area unallocated if it was unallocated prior to the mapping or allocated if it was allocated prior to the mapping. Appropriate privileges to specify the `POSIX_TYPED_MEM_MAP_ALLOCATABLE` flag are implementation-defined.

If successful, *posix\_typed\_mem\_open()* shall return a file descriptor for the typed memory object. The open file description is new, and therefore the file descriptor shall not share it with any other processes. It is unspecified whether the file offset is set. The `FD_CLOEXEC` file descriptor flag associated with the new file descriptor shall be cleared.

The behavior of *msync()*, *ftruncate()*, and all file operations other than *mmap()*, *posix\_mem\_offset()*, *posix\_typed\_mem\_get\_info()*, *fstat()*, *dup()*, *dup2()*, and *close()*, is unspecified when passed a file descriptor connected to a typed memory object by this function.

The file status flags of the open file description shall be set according to the value of *oflag*. Applications shall specify exactly one of the three access mode values described below and defined in the `<fcntl.h>` header, as the value of *oflag*.

- `O_RDONLY`      Open for read access only.
- `O_WRONLY`      Open for write access only.
- `O_RDWR`        Open for read or write access.

#### RETURN VALUE

Upon successful completion, the *posix\_typed\_mem\_open()* function shall return a non-negative integer representing the file descriptor. Otherwise, it shall return `-1` and set *errno* to indicate the error.

#### ERRORS

The *posix\_typed\_mem\_open()* function shall fail if:

- `[EACCES]`      The typed memory object exists and the permissions specified by *oflag* are denied.
- `[EINTR]`        The *posix\_typed\_mem\_open()* operation was interrupted by a signal.
- `[EINVAL]`        The flags specified in *tflag* are invalid (more than one of `POSIX_TYPED_MEM_ALLOCATE`, `POSIX_TYPED_MEM_ALLOCATE_CONTIG`, or `POSIX_TYPED_MEM_MAP_ALLOCATABLE` is specified).
- `[EMFILE]`        All file descriptors available to the process are currently open.
- `[ENFILE]`        Too many file descriptors are currently open in the system.
- `[ENOENT]`        The named typed memory object does not exist.
- `[EPERM]`        The caller lacks appropriate privileges to specify the `POSIX_TYPED_MEM_MAP_ALLOCATABLE` flag in the *tflag* argument.

The *posix\_typed\_mem\_open()* function may fail if:

- `[ENAMETOOLONG]`      The length of the *name* argument exceeds `{_POSIX_PATH_MAX}` on systems that do not support the XSI option or exceeds `{_XOPEN_PATH_MAX}` on XSI systems, or has a pathname component that is longer than

50456 XSI { \_POSIX\_NAME\_MAX } on systems that do not support the XSI option or  
50457 longer than { \_XOPEN\_NAME\_MAX } on XSI systems.

#### 50458 EXAMPLES

50459 None.

#### 50460 APPLICATION USAGE

50461 None.

#### 50462 RATIONALE

50463 None.

#### 50464 FUTURE DIRECTIONS

50465 None.

#### 50466 SEE ALSO

50467 Section 2.14 (on page 549), *close()*, *dup()*, *exec*, *fcntl()*, *fstat()*, *ftruncate()*, *mmap()*, *msync()*,  
50468 *posix\_mem\_offset()*, *posix\_typed\_mem\_get\_info()*, *umask()*

50469 XBD <fcntl.h>, <sys/mman.h>

#### 50470 CHANGE HISTORY

50471 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

#### 50472 Issue 7

50473 Austin Group Interpretation 1003.1-2001 #143 is applied.

50474 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0442 [119,428] is applied.

50475 **NAME**

50476 pow, powf, powl — power function

50477 **SYNOPSIS**

```
50478 #include <math.h>
50479 double pow(double x, double y);
50480 float powf(float x, float y);
50481 long double powl(long double x, long double y);
```

50482 **DESCRIPTION**

50483 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 50484 conflict between the requirements described here and the ISO C standard is unintentional. This  
 50485 volume of POSIX.1-2008 defers to the ISO C standard.

50486 These functions shall compute the value of  $x$  raised to the power  $y$ ,  $x^y$ . If  $x$  is negative, the  
 50487 application shall ensure that  $y$  is an integer value.

50488 An application wishing to check for error situations should set *errno* to zero and call  
 50489 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 50490 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 50491 zero, an error has occurred.

50492 **RETURN VALUE**

50493 Upon successful completion, these functions shall return the value of  $x$  raised to the power  $y$ .

50494 MX For finite values of  $x < 0$ , and finite non-integer values of  $y$ , a domain error shall occur and  
 50495 either a NaN (if representable), or an implementation-defined value shall be returned.

50496 If the correct value would cause overflow, a range error shall occur and *pow()*, *powf()*, and  
 50497 *powl()* shall return  $\pm\text{HUGE\_VAL}$ ,  $\pm\text{HUGE\_VALF}$ , and  $\pm\text{HUGE\_VALL}$ , respectively, with the  
 50498 same sign as the correct value of the function.

50499 MXX If the correct value would cause underflow, and is not representable, a range error may occur,  
 50500 MXX and *pow()*, *powf()*, and *powl()* shall return 0.0, or (if IEC 60559 Floating-Point is not supported)  
 50501 an implementation-defined value no greater in magnitude than DBL\_MIN, FLT\_MIN, and  
 50502 LDBL\_MIN, respectively.

50503 CX For  $y < 0$ , if  $x$  is zero, a pole error may occur and *pow()*, *powf()*, and *powl()* shall return  
 50504 MX  $\pm\text{HUGE\_VAL}$ ,  $\pm\text{HUGE\_VALF}$ , and  $\pm\text{HUGE\_VALL}$ , respectively. On systems that support the  
 50505 IEC 60559 Floating-Point option, if  $x$  is  $\pm 0$ , a pole error shall occur and *pow()*, *powf()*, and *powl()*  
 50506 shall return  $\pm\text{HUGE\_VAL}$ ,  $\pm\text{HUGE\_VALF}$ , and  $\pm\text{HUGE\_VALL}$ , respectively if  $y$  is an odd integer,  
 50507 or  $\text{HUGE\_VAL}$ ,  $\text{HUGE\_VALF}$ , and  $\text{HUGE\_VALL}$ , respectively if  $y$  is not an odd integer.

50508 MX If  $x$  or  $y$  is a NaN, a NaN shall be returned (unless specified elsewhere in this description).

50509 For any value of  $y$  (including NaN), if  $x$  is +1, 1.0 shall be returned.

50510 For any value of  $x$  (including NaN), if  $y$  is  $\pm 0$ , 1.0 shall be returned.

50511 For any odd integer value of  $y > 0$ , if  $x$  is  $\pm 0$ ,  $\pm 0$  shall be returned.

50512 For  $y > 0$  and not an odd integer, if  $x$  is  $\pm 0$ , +0 shall be returned.

50513 If  $x$  is -1, and  $y$  is  $\pm\text{Inf}$ , 1.0 shall be returned.

50514 For  $|x| < 1$ , if  $y$  is  $-\text{Inf}$ ,  $+\text{Inf}$  shall be returned.

50515 For  $|x| > 1$ , if  $y$  is  $-\text{Inf}$ , +0 shall be returned.

50516 For  $|x| < 1$ , if  $y$  is  $+\text{Inf}$ , +0 shall be returned.



50517 For  $|x| > 1$ , if  $y$  is  $+\text{Inf}$ ,  $+\text{Inf}$  shall be returned.

50518 For  $y$  an odd integer  $< 0$ , if  $x$  is  $-\text{Inf}$ ,  $-\text{Inf}$  shall be returned.

50519 For  $y < 0$  and not an odd integer, if  $x$  is  $-\text{Inf}$ ,  $+\text{Inf}$  shall be returned.

50520 For  $y$  an odd integer  $> 0$ , if  $x$  is  $-\text{Inf}$ ,  $-\text{Inf}$  shall be returned.

50521 For  $y > 0$  and not an odd integer, if  $x$  is  $-\text{Inf}$ ,  $+\text{Inf}$  shall be returned.

50522 For  $y < 0$ , if  $x$  is  $+\text{Inf}$ ,  $+\text{Inf}$  shall be returned.

50523 For  $y > 0$ , if  $x$  is  $+\text{Inf}$ ,  $+\text{Inf}$  shall be returned.

50524 MXX If the correct value would cause underflow, and is representable, a range error may occur and  
50525 the correct value shall be returned.

**ERRORS**

50526 These functions shall fail if:

50527 Domain Error The value of  $x$  is negative and  $y$  is a finite non-integer.

50528 If the integer expression  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$  is non-zero,  
50529 then *errno* shall be set to [EDOM]. If the integer expression  $(\text{math\_errhandling}$   
50530  $\ \& \ \text{MATH\_ERREXCEPT})$  is non-zero, then the invalid floating-point exception  
50531 shall be raised.  
50532

50533 MX Pole Error The value of  $x$  is zero and  $y$  is negative.

50534 If the integer expression  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$  is non-zero,  
50535 then *errno* shall be set to [ERANGE]. If the integer expression  
50536  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$  is non-zero, then the divide-by-zero  
50537 floating-point exception shall be raised.

50538 Range Error The result overflows.

50539 If the integer expression  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$  is non-zero,  
50540 then *errno* shall be set to [ERANGE]. If the integer expression  
50541  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$  is non-zero, then the overflow  
50542 floating-point exception shall be raised.

50543 These functions may fail if:

50544 Pole Error The value of  $x$  is zero and  $y$  is negative.

50545 If the integer expression  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$  is non-zero,  
50546 then *errno* shall be set to [ERANGE]. If the integer expression  
50547  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$  is non-zero, then the divide-by-zero  
50548 floating-point exception shall be raised.

50549 Range Error The result underflows.

50550 If the integer expression  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERRNO})$  is non-zero,  
50551 then *errno* shall be set to [ERANGE]. If the integer expression  
50552  $(\text{math\_errhandling} \ \& \ \text{MATH\_ERREXCEPT})$  is non-zero, then the underflow  
50553 floating-point exception shall be raised.

**EXAMPLES**

None.

**APPLICATION USAGE**

On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* & MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*exp()*, *feclearexcept()*, *fetestexcept()*, *isnan()*

XBD Section 4.20 (on page 117), <math.h>

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

The DESCRIPTION is updated to indicate how an application should check for an error. This text was previously published in the APPLICATION USAGE section.

**Issue 6**

The normative text is updated to avoid use of the term “must” for application requirements.

The *powf()* and *powl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/42 is applied, correcting the third paragraph in the RETURN VALUE section.

**Issue 7**

ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #51 (SD5-XSH-ERN-81) is applied.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0443 [68], XSH/TC1-2008/0444 [148], and XSH/TC1-2008/0445 [68] are applied.

50584 **NAME**50585       

```
pread
```

 — read from a file50586 **SYNOPSIS**50587       

```
#include <unistd.h>
```

50588       

```
ssize_t pread(int fildes, void *buf, size_t nbyte, off_t offset);
```

50589 **DESCRIPTION**50590       Refer to *read()*.

50591 **NAME**

50592       printf — print formatted output

50593 **SYNOPSIS**

50594       #include &lt;stdio.h&gt;

50595       int printf(const char \*restrict *format*, ...);50596 **DESCRIPTION**50597       Refer to *fprintf()*.

**NAME**

pselect, select — synchronous I/O multiplexing

**SYNOPSIS**

```
#include <sys/select.h>

int pselect(int nfds, fd_set *restrict readfds,
            fd_set *restrict writefds, fd_set *restrict errorfds,
            const struct timespec *restrict timeout,
            const sigset_t *restrict sigmask);
int select(int nfds, fd_set *restrict readfds,
           fd_set *restrict writefds, fd_set *restrict errorfds,
           struct timeval *restrict timeout);
void FD_CLR(int fd, fd_set *fdset);
int FD_ISSET(int fd, fd_set *fdset);
void FD_SET(int fd, fd_set *fdset);
void FD_ZERO(fd_set *fdset);
```

**DESCRIPTION**

The *pselect()* function shall examine the file descriptor sets whose addresses are passed in the *readfds*, *writefds*, and *errorfds* parameters to see whether some of their descriptors are ready for reading, are ready for writing, or have an exceptional condition pending, respectively.

The *select()* function shall be equivalent to the *pselect()* function, except as follows:

- For the *select()* function, the timeout period is given in seconds and microseconds in an argument of type **struct timeval**, whereas for the *pselect()* function the timeout period is given in seconds and nanoseconds in an argument of type **struct timespec**.
- The *select()* function has no *sigmask* argument; it shall behave as *pselect()* does when *sigmask* is a null pointer.
- Upon successful completion, the *select()* function may modify the object pointed to by the *timeout* argument.

The *pselect()* and *select()* functions shall support regular files, terminal and pseudo-terminal devices, **STREAMS-based files**, FIFOs, pipes, and sockets. The behavior of *pselect()* and *select()* on file descriptors that refer to other types of file is unspecified.

The *nfds* argument specifies the range of descriptors to be tested. The first *nfds* descriptors shall be checked in each set; that is, the descriptors from zero through *nfds*–1 in the descriptor sets shall be examined.

If the *readfds* argument is not a null pointer, it points to an object of type **fd\_set** that on input specifies the file descriptors to be checked for being ready to read, and on output indicates which file descriptors are ready to read.

If the *writefds* argument is not a null pointer, it points to an object of type **fd\_set** that on input specifies the file descriptors to be checked for being ready to write, and on output indicates which file descriptors are ready to write.

If the *errorfds* argument is not a null pointer, it points to an object of type **fd\_set** that on input specifies the file descriptors to be checked for error conditions pending, and on output indicates which file descriptors have error conditions pending.

Upon successful completion, the *pselect()* or *select()* function shall modify the objects pointed to by the *readfds*, *writefds*, and *errorfds* arguments to indicate which file descriptors are ready for reading, ready for writing, or have an error condition pending, respectively, and shall return the total number of ready descriptors in all the output sets. For each file descriptor less than *nfds*, the

corresponding bit shall be set upon successful completion if it was set on input and the associated condition is true for that file descriptor.

If none of the selected descriptors are ready for the requested operation, the *pselect()* or *select()* function shall block until at least one of the requested operations becomes ready, until the *timeout* occurs, or until interrupted by a signal. The *timeout* parameter controls how long the *pselect()* or *select()* function shall take before timing out. If the *timeout* parameter is not a null pointer, it specifies a maximum interval to wait for the selection to complete. If the specified time interval expires without any requested operation becoming ready, the function shall return. If the *timeout* parameter is a null pointer, then the call to *pselect()* or *select()* shall block indefinitely until at least one descriptor meets the specified criteria. To effect a poll, the *timeout* parameter should not be a null pointer, and should point to a zero-valued **timespec** structure.

The use of a timeout does not affect any pending timers set up by *alarm()* or *setitimer()*.

Implementations may place limitations on the maximum timeout interval supported. All implementations shall support a maximum timeout interval of at least 31 days. If the *timeout* argument specifies a timeout interval greater than the implementation-defined maximum value, the maximum value shall be used as the actual timeout value. Implementations may also place limitations on the granularity of timeout intervals. If the requested timeout interval requires a finer granularity than the implementation supports, the actual timeout interval shall be rounded up to the next supported value.

If *sigmask* is not a null pointer, then the *pselect()* function shall replace the signal mask of the caller by the set of signals pointed to by *sigmask* before examining the descriptors, and shall restore the signal mask of the calling thread before returning.

A descriptor shall be considered ready for reading when a call to an input function with **O\_NONBLOCK** clear would not block, whether or not the function would transfer data successfully. (The function might return data, an end-of-file indication, or an error other than one indicating that it is blocked, and in each of these cases the descriptor shall be considered ready for reading.)

A descriptor shall be considered ready for writing when a call to an output function with **O\_NONBLOCK** clear would not block, whether or not the function would transfer data successfully.

If a socket has a pending error, it shall be considered to have an exceptional condition pending. Otherwise, what constitutes an exceptional condition is file type-specific. For a file descriptor for use with a socket, it is protocol-specific except as noted below. For other file types it is implementation-defined. If the operation is meaningless for a particular file type, *pselect()* or *select()* shall indicate that the descriptor is ready for read or write operations, and shall indicate that the descriptor has no exceptional condition pending.

If a descriptor refers to a socket, the implied input function is the *recvmsg()* function with parameters requesting normal and ancillary data, such that the presence of either type shall cause the socket to be marked as readable. The presence of out-of-band data shall be checked if the socket option **SO\_OOBINLINE** has been enabled, as out-of-band data is enqueued with normal data. If the socket is currently listening, then it shall be marked as readable if an incoming connection request has been received, and a call to the *accept()* function shall complete without blocking.

If a descriptor refers to a socket, the implied output function is the *sendmsg()* function supplying an amount of normal data equal to the current value of the **SO\_SNDLOWAT** option for the socket. If a non-blocking call to the *connect()* function has been made for a socket, and the connection attempt has either succeeded or failed leaving a pending error, the socket shall be

marked as writable.

A socket shall be considered to have an exceptional condition pending if a receive operation with `O_NONBLOCK` clear for the open file description and with the `MSG_OOB` flag set would return out-of-band data without blocking. (It is protocol-specific whether the `MSG_OOB` flag would be used to read out-of-band data.) A socket shall also be considered to have an exceptional condition pending if an out-of-band data mark is present in the receive queue. Other circumstances under which a socket may be considered to have an exceptional condition pending are protocol-specific and implementation-defined.

If the `readfds`, `writefds`, and `errorfds` arguments are all null pointers and the `timeout` argument is not a null pointer, the `pselect()` or `select()` function shall block for the time specified, or until interrupted by a signal. If the `readfds`, `writefds`, and `errorfds` arguments are all null pointers and the `timeout` argument is a null pointer, the `pselect()` or `select()` function shall block until interrupted by a signal.

File descriptors associated with regular files shall always select true for ready to read, ready to write, and error conditions.

On failure, the objects pointed to by the `readfds`, `writefds`, and `errorfds` arguments shall not be modified. If the timeout interval expires without the specified condition being true for any of the specified file descriptors, the objects pointed to by the `readfds`, `writefds`, and `errorfds` arguments shall have all bits set to 0.

File descriptor masks of type `fd_set` can be initialized and tested with `FD_CLR()`, `FD_ISSET()`, `FD_SET()`, and `FD_ZERO()`. It is unspecified whether each of these is a macro or a function. If a macro definition is suppressed in order to access an actual function, or a program defines an external identifier with any of these names, the behavior is undefined.

`FD_CLR(fd, fdsetp)` shall remove the file descriptor *fd* from the set pointed to by *fdsetp*. If *fd* is not a member of this set, there shall be no effect on the set, nor will an error be returned.

`FD_ISSET(fd, fdsetp)` shall evaluate to non-zero if the file descriptor *fd* is a member of the set pointed to by *fdsetp*, and shall evaluate to zero otherwise.

`FD_SET(fd, fdsetp)` shall add the file descriptor *fd* to the set pointed to by *fdsetp*. If the file descriptor *fd* is already in this set, there shall be no effect on the set, nor will an error be returned.

`FD_ZERO(fdsetp)` shall initialize the descriptor set pointed to by *fdsetp* to the null set. No error is returned if the set is not empty at the time `FD_ZERO()` is invoked.

The behavior of these macros is undefined if the *fd* argument is less than 0 or greater than or equal to `FD_SETSIZE`, or if *fd* is not a valid file descriptor, or if any of the arguments are expressions with side-effects.

If a thread gets canceled during a `pselect()` call, the signal mask in effect when executing the registered cleanup functions is either the original signal mask or the signal mask installed as part of the `pselect()` call.

## RETURN VALUE

Upon successful completion, the `pselect()` and `select()` functions shall return the total number of bits set in the bit masks. Otherwise, `-1` shall be returned, and `errno` shall be set to indicate the error.

`FD_CLR()`, `FD_SET()`, and `FD_ZERO()` do not return a value. `FD_ISSET()` shall return a non-zero value if the bit for the file descriptor *fd* is set in the file descriptor set pointed to by *fdset*, and 0 otherwise.

**ERRORS**

Under the following conditions, *pselect()* and *select()* shall fail and set *errno* to:

[EBADF] One or more of the file descriptor sets specified a file descriptor that is not a valid open file descriptor.

[EINTR] The function was interrupted while blocked waiting for any of the selected descriptors to become ready and before the timeout interval expired.

If SA\_RESTART has been set for the interrupting signal, it is implementation-defined whether the function restarts or returns with [EINTR].

[EINVAL] An invalid timeout interval was specified.

[EINVAL] The *nfds* argument is less than 0 or greater than FD\_SETSIZE.

[EINVAL] One of the specified file descriptors refers to a STREAM or multiplexer that is linked (directly or indirectly) downstream from a multiplexer.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

In earlier versions of the Single UNIX Specification, the *select()* function was defined in the **<sys/time.h>** header. This is now changed to **<sys/select.h>**. The rationale for this change was as follows: the introduction of the *pselect()* function included the **<sys/select.h>** header and the **<sys/select.h>** header defines all the related definitions for the *pselect()* and *select()* functions. Backwards-compatibility to existing XSI implementations is handled by allowing **<sys/time.h>** to include **<sys/select.h>**.

Code which wants to avoid the ambiguity of the signal mask for thread cancellation handlers can install an additional cancellation handler which resets the signal mask to the expected value.

```
void cleanup(void *arg)
{
    sigset_t *ss = (sigset_t *) arg;
    pthread_sigmask(SIG_SETMASK, ss, NULL);
}

int call_pselect(int nfds, fd_set *readfds, fd_set *writefds,
    fd_set errorfds, const struct timespec *timeout,
    const sigset_t *sigmask)
{
    sigset_t oldmask;
    int result;
    pthread_sigmask(SIG_SETMASK, NULL, &oldmask);
    pthread_cleanup_push(cleanup, &oldmask);
    result = pselect(nfds, readfds, writefds, errorfds, timeout, sigmask);
    pthread_cleanup_pop(0);
    return result;
}
```



**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*accept()*, *alarm()*, *connect()*, *fcntl()*, *getitimer()*, *poll()*, *read()*, *recvmsg()*, *sendmsg()*, *write()*

XBD **<sys/select.h>**, **<sys/time.h>**

**CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

In the ERRORS section, the text has been changed to indicate that [EINVAL] is returned when *nfds* is less than 0 or greater than FD\_SETSIZE. It previously stated less than 0, or greater than or equal to FD\_SETSIZE.

Text about *timeout* is moved from the APPLICATION USAGE section to the DESCRIPTION.

**Issue 6**

The Open Group Corrigendum U026/6 is applied, changing the occurrences of *readfs* and *writefs* in the *select()* DESCRIPTION to be *readfds* and *writefds*.

Text referring to sockets is added to the DESCRIPTION.

The DESCRIPTION and ERRORS sections are updated so that references to STREAMS are marked as part of the XSI STREAMS Option Group.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- These functions are now mandatory.

The *pselect()* function is added for alignment with IEEE Std 1003.1g-2000 and additional detail related to sockets semantics is added to the DESCRIPTION.

The *select()* function now requires inclusion of **<sys/select.h>**.

The **restrict** keyword is added to the *select()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/70 is applied, updating the DESCRIPTION to reference the signal mask in terms of the calling thread rather than the process.

**Issue 7**

SD5-XSH-ERN-122 is applied, adding text to the DESCRIPTION for when a thread is canceled during a call to *pselect()*, and adding example code to the RATIONALE.

Functionality relating to the XSI STREAMS option is marked obsolescent.

Functionality relating to the Threads option is moved to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0446 [372] is applied.

**NAME**

psiginfo, psignal — write signal information to standard error

**SYNOPSIS**

```
CX      #include <signal.h>

void psiginfo(const siginfo_t *pinfo, const char *message);
void psignal(int signum, const char *message);
```

**DESCRIPTION**

The *psiginfo()* and *psignal()* functions shall write a language-dependent message associated with a signal number to the standard error stream as follows:

- First, if *message* is not a null pointer and is not the empty string, the string pointed to by the *message* argument shall be written, followed by a <colon> and a <space>.
- Then the signal description string associated with *signum* or with the signal indicated by *pinfo* shall be written, followed by a <newline>.

For *psiginfo()*, the application shall ensure that the argument *pinfo* references a valid **siginfo\_t** structure. For *psignal()*, if *signum* is not a valid signal number, the behavior is implementation-defined.

The *psiginfo()* and *psignal()* functions shall not change the orientation of the standard error stream.

The *psiginfo()* and *psignal()* functions shall mark for update the last data modification and last file status change timestamps of the file associated with the standard error stream at some time between their successful completion and *exit()*, *abort()*, or the completion of *fflush()* or *fclose()* on *stderr*.

The *psiginfo()* and *psignal()* functions shall not change the setting of *errno* if successful.

On error, the *psiginfo()* and *psignal()* functions shall set the error indicator for the stream to which *stderr* points, and shall set *errno* to indicate the error.

Since no value is returned, an application wishing to check for error situations should set *errno* to 0, then call *psiginfo()* or *psignal()*, then check *errno*.

**RETURN VALUE**

These functions shall not return a value.

**ERRORS**

Refer to *fputc()*.

**EXAMPLES**

None.

**APPLICATION USAGE**

As an alternative to setting *errno* to zero before the call and checking if it is non-zero afterwards, applications can use *ferror()* to detect whether *psiginfo()* or *psignal()* encountered an error.

An application wishing to use this method to check for error situations should call *clearerr(stderr)* before calling *psiginfo()* or *psignal()*, then if *ferror(stderr)* returns non-zero, the value of *errno* indicates which error occurred.

50853 **RATIONALE**

50854 System V historically has *psignal()* and *psiginfo()* in **<siginfo.h>**. However, the **<siginfo.h>**  
50855 header is not specified in the Base Definitions volume of POSIX.1-2008, and the type **siginfo\_t** is  
50856 defined in **<signal.h>**.

50857 **FUTURE DIRECTIONS**

50858 None.

50859 **SEE ALSO**

50860 *fputc()*, *perror()*, *strsignal()*

50861 XBD **<signal.h>**

50862 **CHANGE HISTORY**

50863 First released in Issue 7.

50864 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0447 [399,428], XSH/TC1-2008/0448  
50865 [399], and XSH/TC1-2008/0449 [399,401] are applied.

## NAME

pthread\_atfork — register fork handlers

## SYNOPSIS

```
#include <pthread.h>

int pthread_atfork(void (*prepare)(void), void (*parent)(void),
void (*child)(void));
```

## DESCRIPTION

The *pthread\_atfork()* function shall declare fork handlers to be called before and after *fork()*, in the context of the thread that called *fork()*. The *prepare* fork handler shall be called before *fork()* processing commences. The *parent* fork handler shall be called after *fork()* processing completes in the parent process. The *child* fork handler shall be called after *fork()* processing completes in the child process. If no handling is desired at one or more of these three points, the corresponding fork handler address(es) may be set to NULL.

If a *fork()* call in a multi-threaded process leads to a *child* fork handler calling any function that is not async-signal-safe, the behavior is undefined. +

The order of calls to *pthread\_atfork()* is significant. The *parent* and *child* fork handlers shall be called in the order in which they were established by calls to *pthread\_atfork()*. The *prepare* fork handlers shall be called in the opposite order.

## RETURN VALUE

Upon successful completion, *pthread\_atfork()* shall return a value of zero; otherwise, an error number shall be returned to indicate the error.

## ERRORS

The *pthread\_atfork()* function shall fail if:

[ENOMEM]      Insufficient table space exists to record the fork handler addresses.

The *pthread\_atfork()* function shall not return an error code of [EINTR].

## EXAMPLES

None.

## APPLICATION USAGE

The original usage pattern envisaged for *pthread\_atfork()* was for the *prepare* fork handler to lock mutexes and other locks, and for the *parent* and *child* handlers to unlock them. However, since all of the relevant unlocking functions, except *sem\_post()*, are not async-signal-safe, this usage results in undefined behavior in the child process unless the only such unlocking function it calls is *sem\_post()*.

## RATIONALE

There are at least two serious problems with the semantics of *fork()* in a multi-threaded program. One problem has to do with state (for example, memory) covered by mutexes. Consider the case where one thread has a mutex locked and the state covered by that mutex is inconsistent while another thread calls *fork()*. In the child, the mutex is in the locked state (locked by a nonexistent thread and thus can never be unlocked). Having the child simply reinitialize the mutex is unsatisfactory since this approach does not resolve the question about how to correct or otherwise deal with the inconsistent state in the child.

It is suggested that programs that use *fork()* call an *exec* function very soon afterwards in the child process, thus resetting all states. In the meantime, only a short list of async-signal-safe library routines are promised to be available.

Unfortunately, this solution does not address the needs of multi-threaded libraries. Application

programs may not be aware that a multi-threaded library is in use, and they feel free to call any number of library routines between the *fork()* and *exec* calls, just as they always have. Indeed, they may be extant single-threaded programs and cannot, therefore, be expected to obey new restrictions imposed by the threads library.

On the other hand, the multi-threaded library needs a way to protect its internal state during *fork()* in case it is re-entered later in the child process. The problem arises especially in multi-threaded I/O libraries, which are almost sure to be invoked between the *fork()* and *exec* calls to effect I/O redirection. The solution may require locking mutex variables during *fork()*, or it may entail simply resetting the state in the child after the *fork()* processing completes.

The *pthread\_atfork()* function was intended to provide multi-threaded libraries with a means to protect themselves from innocent application programs that call *fork()*, and to provide multi-threaded application programs with a standard mechanism for protecting themselves from *fork()* calls in a library routine or the application itself.

The expected usage was that the prepare handler would acquire all mutex locks and the other two fork handlers would release them.

For example, an application could have supplied a prepare routine that acquires the necessary mutexes the library maintains and supplied child and parent routines that release those mutexes, thus ensuring that the child would have got a consistent snapshot of the state of the library (and that no mutexes would have been left stranded). This is good in theory, but in reality not practical. Each and every mutex and lock in the process must be located and locked. Every component of a program including third-party components must participate and they must agree who is responsible for which mutex or lock. This is especially problematic for mutexes and locks in dynamically allocated memory. All mutexes and locks internal to the implementation must be locked, too. This possibly delays the thread calling *fork()* for a long time or even indefinitely since uses of these synchronization objects may not be under control of the application. A final problem to mention here is the problem of locking streams. At least the streams under control of the system (like *stdin*, *stdout*, *stderr*) must be protected by locking the stream with *flockfile()*. But the application itself could have done that, possibly in the same thread calling *fork()*. In this case, the process will deadlock.

Alternatively, some libraries might have been able to supply just a *child* routine that reinitializes the mutexes in the library and all associated states to some known value (for example, what it was when the image was originally executed). This approach is not possible, though, because implementations are allowed to fail *\*\_init()* and *\*\_destroy()* calls for mutexes and locks if the mutex or lock is still locked. In this case, the *child* routine is not able to reinitialize the mutexes and locks.

When *fork()* is called, only the calling thread is duplicated in the child process. Synchronization variables remain in the same state in the child as they were in the parent at the time *fork()* was called. Thus, for example, mutex locks may be held by threads that no longer exist in the child process, and any associated states may be inconsistent. The intention was that the parent process could have avoided this by explicit code that acquires and releases locks critical to the child via *pthread\_atfork()*. In addition, any critical threads would have needed to be recreated and reinitialized to the proper state in the child (also via *pthread\_atfork()*).

A higher-level package may acquire locks on its own data structures before invoking lower-level packages. Under this scenario, the order specified for fork handler calls allows a simple rule of initialization for avoiding package deadlock: a package initializes all packages on which it depends before it calls the *pthread\_atfork()* function for itself.

As explained, there is no suitable solution for functionality which requires non-atomic operations to be protected through mutexes and locks. This is why the POSIX.1 standard since

50959 the 1996 release requires that the child process after *fork()* in a multi-threaded process only calls  
50960 async-signal-safe interfaces.

50961 **FUTURE DIRECTIONS**

50962 The *pthread\_atfork()* function may be formally deprecated (for example, by shading it OB) in a  
50963 future version of this standard.

50964 **SEE ALSO**

50965 *atexit()*, *exec*, *fork()*

50966 XBD **<pthread.h>**, **<sys/types.h>**

50967 **CHANGE HISTORY**

50968 First released in Issue 5. Derived from the POSIX Threads Extension.

50969 IEEE PASC Interpretation 1003.1c #4 is applied.

50970 **Issue 6**

50971 The *pthread\_atfork()* function is marked as part of the Threads option.

50972 The **<pthread.h>** header is added to the SYNOPSIS.

50973 **Issue 7**

50974 The *pthread\_atfork()* function is moved from the Threads option to the Base.

50975 SD5-XSH-ERN-145 is applied, updating the RATIONALE.

**NAME**

pthread\_attr\_destroy, pthread\_attr\_init — destroy and initialize the thread attributes object

**SYNOPSIS**

```
#include <pthread.h>

int pthread_attr_destroy(pthread_attr_t *attr);
int pthread_attr_init(pthread_attr_t *attr);
```

**DESCRIPTION**

The *pthread\_attr\_destroy()* function shall destroy a thread attributes object. An implementation may cause *pthread\_attr\_destroy()* to set *attr* to an implementation-defined invalid value. A destroyed *attr* attributes object can be reinitialized using *pthread\_attr\_init()*; the results of otherwise referencing the object after it has been destroyed are undefined.

The *pthread\_attr\_init()* function shall initialize a thread attributes object *attr* with the default value for all of the individual attributes used by a given implementation.

The resulting attributes object (possibly modified by setting individual attribute values) when used by *pthread\_create()* defines the attributes of the thread created. A single attributes object can be used in multiple simultaneous calls to *pthread\_create()*. Results are undefined if *pthread\_attr\_init()* is called specifying an already initialized *attr* attributes object.

The behavior is undefined if the value specified by the *attr* argument to *pthread\_attr\_destroy()* does not refer to an initialized thread attributes object.

**RETURN VALUE**

Upon successful completion, *pthread\_attr\_destroy()* and *pthread\_attr\_init()* shall return a value of 0; otherwise, an error number shall be returned to indicate the error.

**ERRORS**

The *pthread\_attr\_init()* function shall fail if:

[ENOMEM]        Insufficient memory exists to initialize the thread attributes object.

These functions shall not return an error code of [EINTR].

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

Attributes objects are provided for threads, mutexes, and condition variables as a mechanism to support probable future standardization in these areas without requiring that the function itself be changed.

Attributes objects provide clean isolation of the configurable aspects of threads. For example, “stack size” is an important attribute of a thread, but it cannot be expressed portably. When porting a threaded program, stack sizes often need to be adjusted. The use of attributes objects can help by allowing the changes to be isolated in a single place, rather than being spread across every instance of thread creation.

Attributes objects can be used to set up “classes” of threads with similar attributes; for example, “threads with large stacks and high priority” or “threads with minimal stacks”. These classes can be defined in a single place and then referenced wherever threads need to be created. Changes to “class” decisions become straightforward, and detailed analysis of each *pthread\_create()* call is not required.



The attributes objects are defined as opaque types as an aid to extensibility. If these objects had been specified as structures, adding new attributes would force recompilation of all multi-threaded programs when the attributes objects are extended; this might not be possible if different program components were supplied by different vendors.

Additionally, opaque attributes objects present opportunities for improving performance. Argument validity can be checked once when attributes are set, rather than each time a thread is created. Implementations often need to cache kernel objects that are expensive to create. Opaque attributes objects provide an efficient mechanism to detect when cached objects become invalid due to attribute changes.

Since assignment is not necessarily defined on a given opaque type, implementation-defined default values cannot be defined in a portable way. The solution to this problem is to allow attributes objects to be initialized dynamically by attributes object initialization functions, so that default values can be supplied automatically by the implementation.

The following proposal was provided as a suggested alternative to the supplied attributes:

1. Maintain the style of passing a parameter formed by the bitwise-inclusive OR of flags to the initialization routines (*pthread\_create()*, *pthread\_mutex\_init()*, *pthread\_cond\_init()*). The parameter containing the flags should be an opaque type for extensibility. If no flags are set in the parameter, then the objects are created with default characteristics. An implementation may specify implementation-defined flag values and associated behavior.
2. If further specialization of mutexes and condition variables is necessary, implementations may specify additional procedures that operate on the **pthread\_mutex\_t** and **pthread\_cond\_t** objects (instead of on attributes objects).

The difficulties with this solution are:

1. A bitmask is not opaque if bits have to be set into bitvector attributes objects using explicitly-coded bitwise-inclusive OR operations. If the set of options exceeds an **int**, application programmers need to know the location of each bit. If bits are set or read by encapsulation (that is, *get* and *set* functions), then the bitmask is merely an implementation of attributes objects as currently defined and should not be exposed to the programmer.
2. Many attributes are not Boolean or very small integral values. For example, scheduling policy may be placed in 3-bit or 4-bit, but priority requires 5-bit or more, thereby taking up at least 8 bits out of a possible 16 bits on machines with 16-bit integers. Because of this, the bitmask can only reasonably control whether particular attributes are set or not, and it cannot serve as the repository of the value itself. The value needs to be specified as a function parameter (which is non-extensible), or by setting a structure field (which is non-opaque), or by *get* and *set* functions (making the bitmask a redundant addition to the attributes objects).

Stack size is defined as an optional attribute because the very notion of a stack is inherently machine-dependent. Some implementations may not be able to change the size of the stack, for example, and others may not need to because stack pages may be discontinuous and can be allocated and released on demand.

The attribute mechanism has been designed in large measure for extensibility. Future extensions to the attribute mechanism or to any attributes object defined in this volume of POSIX.1-2008 has to be done with care so as not to affect binary-compatibility.

Attributes objects, even if allocated by means of dynamic allocation functions such as *malloc()*,



may have their size fixed at compile time. This means, for example, a *pthread\_create()* in an implementation with extensions to **pthread\_attr\_t** cannot look beyond the area that the binary application assumes is valid. This suggests that implementations should maintain a size field in the attributes object, as well as possibly version information, if extensions in different directions (possibly by different vendors) are to be accommodated.

If an implementation detects that the value specified by the *attr* argument to *pthread\_attr\_destroy()* does not refer to an initialized thread attributes object, it is recommended that the function should fail and report an [EINVAL] error.

If an implementation detects that the value specified by the *attr* argument to *pthread\_attr\_init()* refers to an already initialized thread attributes object, it is recommended that the function should fail and report an [EBUSY] error.

#### **FUTURE DIRECTIONS**

None.

#### **SEE ALSO**

*pthread\_attr\_getstacksize()*, *pthread\_attr\_getdetachstate()*, *pthread\_create()*

XBD <pthread.h>

#### **CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

#### **Issue 6**

The *pthread\_attr\_destroy()* and *pthread\_attr\_init()* functions are marked as part of the Threads option.

IEEE PASC Interpretation 1003.1 #107 is applied, noting that the effect of initializing an already initialized thread attributes object is undefined.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/71 is applied, updating the ERRORS section to add the optional [EINVAL] error for the *pthread\_attr\_destroy()* function, and the optional [EBUSY] error for the *pthread\_attr\_init()* function.

#### **Issue 7**

The *pthread\_attr\_destroy()* and *pthread\_attr\_init()* functions are moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

The [EBUSY] error for an already initialized thread attributes object is removed; this condition results in undefined behavior.

51099 **NAME**

51100 pthread\_attr\_getdetachstate, pthread\_attr\_setdetachstate — get and set the detachstate attribute

51101 **SYNOPSIS**

```
51102 #include <pthread.h>
51103 int pthread_attr_getdetachstate(const pthread_attr_t *attr,
51104 int *detachstate);
51105 int pthread_attr_setdetachstate(pthread_attr_t *attr, int detachstate);
```

51106 **DESCRIPTION**

51107 The *detachstate* attribute controls whether the thread is created in a detached state. If the thread  
51108 is created detached, then use of the ID of the newly created thread by the *pthread\_detach()* or  
51109 *pthread\_join()* function is an error.

51110 The *pthread\_attr\_getdetachstate()* and *pthread\_attr\_setdetachstate()* functions, respectively, shall get  
51111 and set the *detachstate* attribute in the *attr* object.

51112 For *pthread\_attr\_getdetachstate()*, *detachstate* shall be set to either  
51113 PTHREAD\_CREATE\_DETACHED or PTHREAD\_CREATE\_JOINABLE.

51114 For *pthread\_attr\_setdetachstate()*, the application shall set *detachstate* to either  
51115 PTHREAD\_CREATE\_DETACHED or PTHREAD\_CREATE\_JOINABLE.

51116 A value of PTHREAD\_CREATE\_DETACHED shall cause all threads created with *attr* to be in  
51117 the detached state, whereas using a value of PTHREAD\_CREATE\_JOINABLE shall cause all  
51118 threads created with *attr* to be in the joinable state. The default value of the *detachstate* attribute  
51119 shall be PTHREAD\_CREATE\_JOINABLE.

51120 The behavior is undefined if the value specified by the *attr* argument to  
51121 *pthread\_attr\_getdetachstate()* or *pthread\_attr\_setdetachstate()* does not refer to an initialized thread  
51122 attributes object.

51123 **RETURN VALUE**

51124 Upon successful completion, *pthread\_attr\_getdetachstate()* and *pthread\_attr\_setdetachstate()* shall  
51125 return a value of 0; otherwise, an error number shall be returned to indicate the error.

51126 The *pthread\_attr\_getdetachstate()* function stores the value of the *detachstate* attribute in *detachstate*  
51127 if successful.

51128 **ERRORS**

51129 The *pthread\_attr\_setdetachstate()* function shall fail if:

51130 [EINVAL] The value of *detachstate* was not valid

51131 These functions shall not return an error code of [EINTR].

51132 **EXAMPLES**

51133 **Retrieving the detachstate Attribute**

51134 This example shows how to obtain the *detachstate* attribute of a thread attribute object.

```
51135 #include <pthread.h>
51136 pthread_attr_t thread_attr;
51137 int detachstate;
51138 int rc;
51139 /* code initializing thread_attr */
51140 ...
```

```

51141     rc = pthread_attr_getdetachstate (&thread_attr, &detachstate);
51142     if (rc!=0) {
51143         /* handle error */
51144         ...
51145     }
51146     else {
51147         /* legal values for detachstate are:
51148          * PTHREAD_CREATE_DETACHED or PTHREAD_CREATE_JOINABLE
51149          */
51150         ...
51151     }

```

**APPLICATION USAGE**

None.

**RATIONALE**

If an implementation detects that the value specified by the *attr* argument to *pthread\_attr\_getdetachstate()* or *pthread\_attr\_setdetachstate()* does not refer to an initialized thread attributes object, it is recommended that the function should fail and report an [EINVAL] error.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*pthread\_attr\_destroy()*, *pthread\_attr\_getstacksize()*, *pthread\_create()*

XBD <pthread.h>

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

**Issue 6**

The *pthread\_attr\_setdetachstate()* and *pthread\_attr\_getdetachstate()* functions are marked as part of the Threads option.

The normative text is updated to avoid use of the term “must” for application requirements.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/72 is applied, adding the example to the EXAMPLES section.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/73 is applied, updating the ERRORS section to include the optional [EINVAL] error.

**Issue 7**

The *pthread\_attr\_setdetachstate()* and *pthread\_attr\_getdetachstate()* functions are moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

51178 **NAME**

51179 pthread\_attr\_getguardsize, pthread\_attr\_setguardsize — get and set the thread guardsize  
 51180 attribute

51181 **SYNOPSIS**

```
51182 #include <pthread.h>

51183 int pthread_attr_getguardsize(const pthread_attr_t *restrict attr,
51184                               size_t *restrict guardsize);
51185 int pthread_attr_setguardsize(pthread_attr_t *attr,
51186                               size_t guardsize);
```

51187 **DESCRIPTION**

51188 The *pthread\_attr\_getguardsize()* function shall get the *guardsize* attribute in the *attr* object. This  
 51189 attribute shall be returned in the *guardsize* parameter.

51190 The *pthread\_attr\_setguardsize()* function shall set the *guardsize* attribute in the *attr* object. The new  
 51191 value of this attribute shall be obtained from the *guardsize* parameter. If *guardsize* is zero, a guard  
 51192 area shall not be provided for threads created with *attr*. If *guardsize* is greater than zero, a guard  
 51193 area of at least size *guardsize* bytes shall be provided for each thread created with *attr*.

51194 The *guardsize* attribute controls the size of the guard area for the created thread's stack. The  
 51195 *guardsize* attribute provides protection against overflow of the stack pointer. If a thread's stack is  
 51196 created with guard protection, the implementation allocates extra memory at the overflow end  
 51197 of the stack as a buffer against stack overflow of the stack pointer. If an application overflows  
 51198 into this buffer an error shall result (possibly in a SIGSEGV signal being delivered to the thread).

51199 A conforming implementation may round up the value contained in *guardsize* to a multiple of  
 51200 the configurable system variable {PAGESIZE} (see <sys/mman.h>). If an implementation  
 51201 rounds up the value of *guardsize* to a multiple of {PAGESIZE}, a call to *pthread\_attr\_getguardsize()*  
 51202 specifying *attr* shall store in the *guardsize* parameter the guard size specified by the previous  
 51203 *pthread\_attr\_setguardsize()* function call.

51204 The default value of the *guardsize* attribute is implementation-defined.

51205 If the *stackaddr* attribute has been set (that is, the caller is allocating and managing its own thread  
 51206 stacks), the *guardsize* attribute shall be ignored and no protection shall be provided by the  
 51207 implementation. It is the responsibility of the application to manage stack overflow along with  
 51208 stack allocation and management in this case.

51209 The behavior is undefined if the value specified by the *attr* argument to  
 51210 *pthread\_attr\_getguardsize()* or *pthread\_attr\_setguardsize()* does not refer to an initialized thread  
 51211 attributes object.

51212 **RETURN VALUE**

51213 If successful, the *pthread\_attr\_getguardsize()* and *pthread\_attr\_setguardsize()* functions shall return  
 51214 zero; otherwise, an error number shall be returned to indicate the error.

51215 **ERRORS**

51216 These functions shall fail if:

51217 [EINVAL] The parameter *guardsize* is invalid.

51218 These functions shall not return an error code of [EINTR].

51219 **EXAMPLES**51220 **Retrieving the guardsize Attribute**

51221 This example shows how to obtain the *guardsize* attribute of a thread attribute object.

```
51222 #include <pthread.h>
51223 pthread_attr_t thread_attr;
51224 size_t guardsize;
51225 int rc;
51226
51227 /* code initializing thread_attr */
51228 ...
51229 rc = pthread_attr_getguardsize (&thread_attr, &guardsize);
51230 if (rc != 0) {
51231     /* handle error */
51232     ...
51233 }
51234 else {
51235     if (guardsize > 0) {
51236         /* a guard area of at least guardsize bytes is provided */
51237         ...
51238     }
51239     else {
51240         /* no guard area provided */
51241         ...
51242     }
51243 }
```

51243 **APPLICATION USAGE**

51244 None.

51245 **RATIONALE**

51246 The *guardsize* attribute is provided to the application for two reasons:

- 51247 1. Overflow protection can potentially result in wasted system resources. An application  
51248 that creates a large number of threads, and which knows its threads never overflow their  
51249 stack, can save system resources by turning off guard areas.
- 51250 2. When threads allocate large data structures on the stack, large guard areas may be needed  
51251 to detect stack overflow.

51252 The default size of the guard area is left implementation-defined since on systems supporting  
51253 very large page sizes, the overhead might be substantial if at least one guard page is required by  
51254 default.

51255 If an implementation detects that the value specified by the *attr* argument to  
51256 *pthread\_attr\_getguardsize()* or *pthread\_attr\_setguardsize()* does not refer to an initialized thread  
51257 attributes object, it is recommended that the function should fail and report an [EINVAL] error.

51258 **FUTURE DIRECTIONS**

51259 None.

51260 **SEE ALSO**51261 XBD [<pthread.h>](#), [<sys/mman.h>](#)51262 **CHANGE HISTORY**

51263 First released in Issue 5.

51264 **Issue 6**51265 In the ERRORS section, a third [EINVAL] error condition is removed as it is covered by the  
51266 second error condition.51267 The **restrict** keyword is added to the *pthread\_attr\_getguardsize()* prototype for alignment with the  
51268 ISO/IEC 9899:1999 standard.51269 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/74 is applied, updating the ERRORS  
51270 section to remove the [EINVAL] error ("The attribute *attr* is invalid."), and replacing it with the  
51271 optional [EINVAL] error.51272 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/76 is applied, adding the example to the  
51273 EXAMPLES section.51274 **Issue 7**51275 SD5-XSH-ERN-111 is applied, removing the reference to the *stack* attribute in the DESCRIPTION.51276 SD5-XSH-ERN-175 is applied, updating the DESCRIPTION to note that the default size of the  
51277 guard area is implementation-defined.51278 The *pthread\_attr\_getguardsize()* and *pthread\_attr\_setguardsize()* functions are moved from the XSI  
51279 option to the Base.51280 The [EINVAL] error for an uninitialized thread attributes object is removed; this condition  
51281 results in undefined behavior.

**51282 NAME**

51283 pthread\_attr\_getinheritsched, pthread\_attr\_setinheritsched — get and set the inheritsched  
 51284 attribute (**REALTIME THREADS**)

**51285 SYNOPSIS**

```
51286 TPS #include <pthread.h>
51287
51287 int pthread_attr_getinheritsched(const pthread_attr_t *restrict attr,
51288 int *restrict inheritsched);
51289 int pthread_attr_setinheritsched(pthread_attr_t *attr,
51290 int inheritsched);
```

**51291 DESCRIPTION**

51292 The *pthread\_attr\_getinheritsched()* and *pthread\_attr\_setinheritsched()* functions, respectively, shall  
 51293 get and set the *inheritsched* attribute in the *attr* argument.

51294 When the attributes objects are used by *pthread\_create()*, the *inheritsched* attribute determines  
 51295 how the other scheduling attributes of the created thread shall be set.

51296 The supported values of *inheritsched* shall be:

**51297 PTHREAD\_INHERIT\_SCHED**

51298 Specifies that the thread scheduling attributes shall be inherited from the creating thread,  
 51299 and the scheduling attributes in this *attr* argument shall be ignored.

**51300 PTHREAD\_EXPLICIT\_SCHED**

51301 Specifies that the thread scheduling attributes shall be set to the corresponding values from  
 51302 this attributes object.

51303 The symbols *PTHREAD\_INHERIT\_SCHED* and *PTHREAD\_EXPLICIT\_SCHED* are defined in  
 51304 the **<pthread.h>** header.

51305 The following thread scheduling attributes defined by POSIX.1-2008 are affected by the  
 51306 *inheritsched* attribute: scheduling policy (*schedpolicy*), scheduling parameters (*schedparam*), and  
 51307 scheduling contention scope (*contentionscope*).

51308 The behavior is undefined if the value specified by the *attr* argument to  
 51309 *pthread\_attr\_getinheritsched()* or *pthread\_attr\_setinheritsched()* does not refer to an initialized  
 51310 thread attributes object.

**51311 RETURN VALUE**

51312 If successful, the *pthread\_attr\_getinheritsched()* and *pthread\_attr\_setinheritsched()* functions shall  
 51313 return zero; otherwise, an error number shall be returned to indicate the error.

**51314 ERRORS**

51315 The *pthread\_attr\_setinheritsched()* function shall fail if:

51316 [ENOTSUP] An attempt was made to set the attribute to an unsupported value.

51317 The *pthread\_attr\_setinheritsched()* function may fail if:

51318 [EINVAL] The value of *inheritsched* is not valid.

51319 These functions shall not return an error code of [EINTR].

## EXAMPLES

None.

## APPLICATION USAGE

After these attributes have been set, a thread can be created with the specified attributes using *pthread\_create()*. Using these routines does not affect the current running thread.

See [Section 2.9.4](#) (on page 515) for further details on thread scheduling attributes and their default settings.

## RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread\_attr\_getinheritsched()* or *pthread\_attr\_setinheritsched()* does not refer to an initialized thread attributes object, it is recommended that the function should fail and report an [EINVAL] error.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*pthread\_attr\_destroy()*, *pthread\_attr\_getscope()*, *pthread\_attr\_getschedpolicy()*,  
*pthread\_attr\_getschedparam()*, *pthread\_create()*

XBD [<pthread.h>](#), [<sched.h>](#)

## CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Marked as part of the Realtime Threads Feature Group.

### Issue 6

The *pthread\_attr\_getinheritsched()* and *pthread\_attr\_setinheritsched()* functions are marked as part of the Threads and Thread Execution Scheduling options.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Execution Scheduling option.

The **restrict** keyword is added to the *pthread\_attr\_getinheritsched()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/75 is applied, clarifying the values of *inheritsched* in the DESCRIPTION and adding two optional [EINVAL] errors to the ERRORS section for checking when *attr* refers to an uninitialized thread attribute object.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/77 is applied, adding a reference to [Section 2.9.4](#) (on page 515) in the APPLICATION USAGE section.

### Issue 7

The *pthread\_attr\_getinheritsched()* and *pthread\_attr\_setinheritsched()* functions are marked only as part of the Thread Execution Scheduling option as the Threads option is now part of the Base.

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0450 [314] is applied.



**51359 NAME**

51360 pthread\_attr\_getschedparam, pthread\_attr\_setschedparam — get and set the schedparam  
51361 attribute

**51362 SYNOPSIS**

```
51363 #include <pthread.h>

51364 int pthread_attr_getschedparam(const pthread_attr_t *restrict attr,
51365                               struct sched_param *restrict param);
51366 int pthread_attr_setschedparam(pthread_attr_t *restrict attr,
51367                               const struct sched_param *restrict param);
```

**51368 DESCRIPTION**

51369 The *pthread\_attr\_getschedparam()* and *pthread\_attr\_setschedparam()* functions, respectively, shall  
51370 get and set the scheduling parameter attributes in the *attr* argument. The contents of the *param*  
51371 structure are defined in the **<sched.h>** header. For the SCHED\_FIFO and SCHED\_RR policies,  
51372 the only required member of *param* is *sched\_priority*.

51373 TSP For the SCHED\_SPORADIC policy, the required members of the *param* structure are  
51374 *sched\_priority*, *sched\_ss\_low\_priority*, *sched\_ss\_repl\_period*, *sched\_ss\_init\_budget*, and  
51375 *sched\_ss\_max\_repl*. The specified *sched\_ss\_repl\_period* must be greater than or equal to the  
51376 specified *sched\_ss\_init\_budget* for the function to succeed; if it is not, then the function shall fail.  
51377 The value of *sched\_ss\_max\_repl* shall be within the inclusive range [1, {SS\_REPL\_MAX}] for the  
51378 function to succeed; if not, the function shall fail. It is unspecified whether the  
51379 *sched\_ss\_repl\_period* and *sched\_ss\_init\_budget* values are stored as provided by this function or are  
51380 rounded to align with the resolution of the clock being used.

51381 The behavior is undefined if the value specified by the *attr* argument to  
51382 *pthread\_attr\_getschedparam()* or *pthread\_attr\_setschedparam()* does not refer to an initialized thread  
51383 attributes object.

**51384 RETURN VALUE**

51385 If successful, the *pthread\_attr\_getschedparam()* and *pthread\_attr\_setschedparam()* functions shall  
51386 return zero; otherwise, an error number shall be returned to indicate the error.

**51387 ERRORS**

51388 The *pthread\_attr\_setschedparam()* function shall fail if:

51389 [ENOTSUP] An attempt was made to set the attribute to an unsupported value.

51390 The *pthread\_attr\_setschedparam()* function may fail if:

51391 [EINVAL] The value of *param* is not valid.

51392 These functions shall not return an error code of [EINTR].

**51393 EXAMPLES**

51394 None.

**51395 APPLICATION USAGE**

51396 After these attributes have been set, a thread can be created with the specified attributes using  
51397 *pthread\_create()*. Using these routines does not affect the current running thread.

**51398 RATIONALE**

51399 If an implementation detects that the value specified by the *attr* argument to  
51400 *pthread\_attr\_getschedparam()* or *pthread\_attr\_setschedparam()* does not refer to an initialized thread  
51401 attributes object, it is recommended that the function should fail and report an [EINVAL] error.

51402 **FUTURE DIRECTIONS**

51403 None.

51404 **SEE ALSO**51405 *pthread\_attr\_destroy()*, *pthread\_attr\_getscope()*, *pthread\_attr\_getinheritsched()*,  
51406 *pthread\_attr\_getschedpolicy()*, *pthread\_create()*

51407 XBD &lt;pthread.h&gt;, &lt;sched.h&gt;

51408 **CHANGE HISTORY**

51409 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

51410 **Issue 6**51411 The *pthread\_attr\_getschedparam()* and *pthread\_attr\_setschedparam()* functions are marked as part  
51412 of the Threads option.

51413 The SCHED\_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

51414 The **restrict** keyword is added to the *pthread\_attr\_getschedparam()* and  
51415 *pthread\_attr\_setschedparam()* prototypes for alignment with the ISO/IEC 9899:1999 standard.51416 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/78 is applied, updating the ERRORS  
51417 section to include optional errors for the case when *attr* refers to an uninitialized thread attribute  
51418 object.51419 **Issue 7**51420 The *pthread\_attr\_getschedparam()* and *pthread\_attr\_setschedparam()* functions are moved from the  
51421 Threads option to the Base.51422 Austin Group Interpretation 1003.1-2001 #119 is applied, clarifying the accuracy requirements  
51423 for the *sched\_ss\_repl\_period* and *sched\_ss\_init\_budget* values.51424 The [EINVAL] error for an uninitialized thread attributes object is removed; this condition  
51425 results in undefined behavior.

51426 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0451 [314] is applied.

51427 **NAME**

51428 pthread\_attr\_getschedpolicy, pthread\_attr\_setschedpolicy — get and set the schedpolicy  
 51429 attribute (**REALTIME THREADS**)

51430 **SYNOPSIS**

```
51431 TPS    #include <pthread.h>
51432
51432    int pthread_attr_getschedpolicy(const pthread_attr_t *restrict attr,
51433                                   int *restrict policy);
51434    int pthread_attr_setschedpolicy(pthread_attr_t *attr, int policy);
```

51435 **DESCRIPTION**

51436 The *pthread\_attr\_getschedpolicy()* and *pthread\_attr\_setschedpolicy()* functions, respectively, shall  
 51437 get and set the *schedpolicy* attribute in the *attr* argument.

51438 The supported values of *policy* shall include SCHED\_FIFO, SCHED\_RR, and SCHED\_OTHER,  
 51439 which are defined in the **<sched.h>** header. When threads executing with the scheduling policy  
 51440 TSP SCHED\_FIFO, SCHED\_RR, or SCHED\_SPORADIC are waiting on a mutex, they shall acquire  
 51441 the mutex in priority order when the mutex is unlocked.

51442 The behavior is undefined if the value specified by the *attr* argument to  
 51443 *pthread\_attr\_getschedpolicy()* or *pthread\_attr\_setschedpolicy()* does not refer to an initialized thread  
 51444 attributes object.

51445 **RETURN VALUE**

51446 If successful, the *pthread\_attr\_getschedpolicy()* and *pthread\_attr\_setschedpolicy()* functions shall  
 51447 return zero; otherwise, an error number shall be returned to indicate the error.

51448 **ERRORS**

51449 The *pthread\_attr\_setschedpolicy()* function shall fail if:

51450 [ENOTSUP] An attempt was made to set the attribute to an unsupported value.

51451 The *pthread\_attr\_setschedpolicy()* function may fail if:

51452 [EINVAL] The value of *policy* is not valid.

51453 These functions shall not return an error code of [EINTR].

51454 **EXAMPLES**

51455 None.

51456 **APPLICATION USAGE**

51457 After these attributes have been set, a thread can be created with the specified attributes using  
 51458 *pthread\_create()*. Using these routines does not affect the current running thread.

51459 See [Section 2.9.4](#) (on page 515) for further details on thread scheduling attributes and their  
 51460 default settings.

51461 **RATIONALE**

51462 If an implementation detects that the value specified by the *attr* argument to  
 51463 *pthread\_attr\_getschedpolicy()* or *pthread\_attr\_setschedpolicy()* does not refer to an initialized thread  
 51464 attributes object, it is recommended that the function should fail and report an [EINVAL] error.

51465 **FUTURE DIRECTIONS**

51466 None.

**SEE ALSO**

*pthread\_attr\_destroy()*, *pthread\_attr\_getscope()*, *pthread\_attr\_getinheritsched()*,  
*pthread\_attr\_getschedparam()*, *pthread\_create()*

XBD <pthread.h>, <sched.h>

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Marked as part of the Realtime Threads Feature Group.

**Issue 6**

The *pthread\_attr\_getschedpolicy()* and *pthread\_attr\_setschedpolicy()* functions are marked as part of the Threads and Thread Execution Scheduling options.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Execution Scheduling option.

The SCHED\_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the *pthread\_attr\_getschedpolicy()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/79 is applied, adding a reference to [Section 2.9.4](#) (on page 515) in the APPLICATION USAGE section.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/80 is applied, updating the ERRORS section to include optional errors for the case when *attr* refers to an uninitialized thread attribute object.

**Issue 7**

The *pthread\_attr\_getschedpolicy()* and *pthread\_attr\_setschedpolicy()* functions are marked only as part of the Thread Execution Scheduling option as the Threads option is now part of the Base.

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0452 [314] is applied.

**51493 NAME**

51494 pthread\_attr\_getscope, pthread\_attr\_setscope — get and set the contentionscope attribute  
 51495 (REALTIME THREADS)

**51496 SYNOPSIS**

```
51497 TPS #include <pthread.h>
51498
51498 int pthread_attr_getscope(const pthread_attr_t *restrict attr,
51499 int *restrict contentionscope);
51500
51500 int pthread_attr_setscope(pthread_attr_t *attr, int contentionscope);
```

**51501 DESCRIPTION**

51502 The *pthread\_attr\_getscope()* and *pthread\_attr\_setscope()* functions, respectively, shall get and set  
 51503 the *contentionscope* attribute in the *attr* object.

51504 The *contentionscope* attribute may have the values PTHREAD\_SCOPE\_SYSTEM, signifying  
 51505 system scheduling contention scope, or PTHREAD\_SCOPE\_PROCESS, signifying process  
 51506 scheduling contention scope. The symbols PTHREAD\_SCOPE\_SYSTEM and  
 51507 PTHREAD\_SCOPE\_PROCESS are defined in the **<pthread.h>** header.

51508 The behavior is undefined if the value specified by the *attr* argument to *pthread\_attr\_getscope()* or  
 51509 *pthread\_attr\_setscope()* does not refer to an initialized thread attributes object.

**51510 RETURN VALUE**

51511 If successful, the *pthread\_attr\_getscope()* and *pthread\_attr\_setscope()* functions shall return zero;  
 51512 otherwise, an error number shall be returned to indicate the error.

**51513 ERRORS**

51514 The *pthread\_attr\_setscope()* function shall fail if:

51515 [ENOTSUP] An attempt was made to set the attribute to an unsupported value.

51516 The *pthread\_attr\_setscope()* function may fail if:

51517 [EINVAL] The value of *contentionscope* is not valid.

51518 These functions shall not return an error code of [EINTR].

**51519 EXAMPLES**

51520 None.

**51521 APPLICATION USAGE**

51522 After these attributes have been set, a thread can be created with the specified attributes using  
 51523 *pthread\_create()*. Using these routines does not affect the current running thread.

51524 See [Section 2.9.4](#) (on page 515) for further details on thread scheduling attributes and their  
 51525 default settings.

**51526 RATIONALE**

51527 If an implementation detects that the value specified by the *attr* argument to  
 51528 *pthread\_attr\_getscope()* or *pthread\_attr\_setscope()* does not refer to an initialized thread attributes  
 51529 object, it is recommended that the function should fail and report an [EINVAL] error.

**51530 FUTURE DIRECTIONS**

51531 None.

**SEE ALSO**

*pthread\_attr\_destroy()*, *pthread\_attr\_getinheritsched()*, *pthread\_attr\_getschedpolicy()*,  
*pthread\_attr\_getschedparam()*, *pthread\_create()*

XBD **<pthread.h>**, **<sched.h>**

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Marked as part of the Realtime Threads Feature Group.

**Issue 6**

The *pthread\_attr\_getscope()* and *pthread\_attr\_setscope()* functions are marked as part of the Threads and Thread Execution Scheduling options.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Execution Scheduling option.

The **restrict** keyword is added to the *pthread\_attr\_getscope()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/81 is applied, adding a reference to [Section 2.9.4](#) (on page 515) in the APPLICATION USAGE section.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/82 is applied, updating the ERRORS section to include optional errors for the case when *attr* refers to an uninitialized thread attribute object.

**Issue 7**

The *pthread\_attr\_getscope()* and *pthread\_attr\_setscope()* functions are marked only as part of the Thread Execution Scheduling option as the Threads option is now part of the Base.

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0453 [314] is applied.

**NAME**

pthread\_attr\_getstack, pthread\_attr\_setstack — get and set stack attributes

**SYNOPSIS**

```
TSA TSS #include <pthread.h>
```

```
int pthread_attr_getstack(const pthread_attr_t *restrict attr,
    void **restrict stackaddr, size_t *restrict stacksize);
int pthread_attr_setstack(pthread_attr_t *attr, void *stackaddr,
    size_t stacksize);
```

**DESCRIPTION**

The *pthread\_attr\_getstack()* and *pthread\_attr\_setstack()* functions, respectively, shall get and set the thread creation stack attributes *stackaddr* and *stacksize* in the *attr* object.

The stack attributes specify the area of storage to be used for the created thread's stack. The base (lowest addressable byte) of the storage shall be *stackaddr*, and the size of the storage shall be *stacksize* bytes. The *stacksize* shall be at least {PTHREAD\_STACK\_MIN}. The *pthread\_attr\_setstack()* function may fail with [EINVAL] if *stackaddr* does not meet implementation-defined alignment requirements. All pages within the stack described by *stackaddr* and *stacksize* shall be both readable and writable by the thread.

If the *pthread\_attr\_getstack()* function is called before the *stackaddr* attribute has been set, the behavior is unspecified.

The behavior is undefined if the value specified by the *attr* argument to *pthread\_attr\_getstack()* or *pthread\_attr\_setstack()* does not refer to an initialized thread attributes object.

**RETURN VALUE**

Upon successful completion, these functions shall return a value of 0; otherwise, an error number shall be returned to indicate the error.

The *pthread\_attr\_getstack()* function shall store the stack attribute values in *stackaddr* and *stacksize* if successful.

**ERRORS**

The *pthread\_attr\_setstack()* function shall fail if:

[EINVAL] The value of *stacksize* is less than {PTHREAD\_STACK\_MIN} or exceeds an implementation-defined limit.

The *pthread\_attr\_setstack()* function may fail if:

[EINVAL] The value of *stackaddr* does not have proper alignment to be used as a stack, or ((**char** \*)*stackaddr* + *stacksize*) lacks proper alignment.

[EACCES] The stack page(s) described by *stackaddr* and *stacksize* are not both readable and writable by the thread.

These functions shall not return an error code of [EINTR].



## EXAMPLES

None.

## APPLICATION USAGE

These functions are appropriate for use by applications in an environment where the stack for a thread must be placed in some particular region of memory.

While it might seem that an application could detect stack overflow by providing a protected page outside the specified stack region, this cannot be done portably. Implementations are free to place the thread's initial stack pointer anywhere within the specified region to accommodate the machine's stack pointer behavior and allocation requirements. Furthermore, on some architectures, such as the IA-64, "overflow" might mean that two separate stack pointers allocated within the region will overlap somewhere in the middle of the region.

After a successful call to *pthread\_attr\_setstack()*, the storage area specified by the *stackaddr* parameter is under the control of the implementation, as described in [Section 2.9.8](#) (on page 522).

The specification of the *stackaddr* attribute presents several ambiguities that make portable use of these functions impossible. For example, the standard allows implementations to impose arbitrary alignment requirements on *stackaddr*. Applications cannot assume that a buffer obtained from *malloc()* is suitably aligned. Note that although the *stacksize* value passed to *pthread\_attr\_setstack()* must satisfy alignment requirements, the same is not true for *pthread\_attr\_setstacksize()* where the implementation must increase the specified size if necessary to achieve the proper alignment.

## RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread\_attr\_getstack()* or *pthread\_attr\_setstack()* does not refer to an initialized thread attributes object, it is recommended that the function should fail and report an [EINVAL] error.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[\*pthread\\_attr\\_destroy\(\)\*](#), [\*pthread\\_attr\\_getdetachstate\(\)\*](#), [\*pthread\\_attr\\_getstacksize\(\)\*](#), [\*pthread\\_create\(\)\*](#)

XBD [\*<limits.h>\*](#), [\*<pthread.h>\*](#)

## CHANGE HISTORY

First released in Issue 6. Developed as part of the XSI option and brought into the BASE by IEEE PASC Interpretation 1003.1 #101.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/83 is applied, updating the APPLICATION USAGE section to refer to [Section 2.9.8](#) (on page 522).

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC/D6/84 is applied, updating the ERRORS section to include optional errors for the case when *attr* refers to an uninitialized thread attribute object.

## Issue 7

SD5-XSH-ERN-66 is applied, correcting the use of *attr* in the [EINVAL] error condition.

Austin Group Interpretation 1003.1-2001 #057 is applied, clarifying the behavior if the function is called before the *stackaddr* attribute is set.

SD5-XSH-ERN-157 is applied, updating the APPLICATION USAGE section.

The description of the *stackaddr* attribute is updated in the DESCRIPTION and APPLICATION USAGE sections.



51637  
51638

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

51639 **NAME**

51640 pthread\_attr\_getstacksize, pthread\_attr\_setstacksize — get and set the stacksize attribute

51641 **SYNOPSIS**

```
51642 TSS      #include <pthread.h>
51643
51643      int pthread_attr_getstacksize(const pthread_attr_t *restrict attr,
51644                                   size_t *restrict stacksize);
51645      int pthread_attr_setstacksize(pthread_attr_t *attr, size_t stacksize);
```

51646 **DESCRIPTION**

51647 The *pthread\_attr\_getstacksize()* and *pthread\_attr\_setstacksize()* functions, respectively, shall get and  
51648 set the thread creation *stacksize* attribute in the *attr* object.

51649 The *stacksize* attribute shall define the minimum stack size (in bytes) allocated for the created  
51650 threads stack.

51651 The behavior is undefined if the value specified by the *attr* argument to  
51652 *pthread\_attr\_getstacksize()* or *pthread\_attr\_setstacksize()* does not refer to an initialized thread  
51653 attributes object.

51654 **RETURN VALUE**

51655 Upon successful completion, *pthread\_attr\_getstacksize()* and *pthread\_attr\_setstacksize()* shall  
51656 return a value of 0; otherwise, an error number shall be returned to indicate the error.

51657 The *pthread\_attr\_getstacksize()* function stores the *stacksize* attribute value in *stacksize* if  
51658 successful.

51659 **ERRORS**

51660 The *pthread\_attr\_setstacksize()* function shall fail if:

51661 [EINVAL] The value of *stacksize* is less than {PTHREAD\_STACK\_MIN} or exceeds a  
51662 system-imposed limit.

51663 These functions shall not return an error code of [EINTR].

51664 **EXAMPLES**

51665 None.

51666 **APPLICATION USAGE**

51667 None.

51668 **RATIONALE**

51669 If an implementation detects that the value specified by the *attr* argument to  
51670 *pthread\_attr\_getstacksize()* or *pthread\_attr\_setstacksize()* does not refer to an initialized thread  
51671 attributes object, it is recommended that the function should fail and report an [EINVAL] error.

51672 **FUTURE DIRECTIONS**

51673 None.

51674 **SEE ALSO**

51675 *pthread\_attr\_destroy()*, *pthread\_attr\_getdetachstate()*, *pthread\_create()*

51676 XBD <limits.h>, <pthread.h>

51677 **CHANGE HISTORY**

51678 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

**Issue 6**

51679 The *pthread\_attr\_getstacksize()* and *pthread\_attr\_setstacksize()* functions are marked as part of the  
51680 Threads and Thread Stack Size Attribute options.  
51681

51682 The **restrict** keyword is added to the *pthread\_attr\_getstacksize()* prototype for alignment with the  
51683 ISO/IEC 9899:1999 standard.

51684 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/43 is applied, correcting the margin code  
51685 in the SYNOPSIS from TSA to TSS and updating the CHANGE HISTORY from “Thread Stack  
51686 Address Attribute” option to “Thread Stack Size Attribute” option.

51687 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/87 is applied, updating the ERRORS  
51688 section to include optional errors for the case when *attr* refers to an uninitialized thread attribute  
51689 object.

**Issue 7**

51690 The *pthread\_attr\_getstacksize()* and *pthread\_attr\_setstacksize()* functions are marked only as part of  
51691 the Thread Stack Size Attribute option as the Threads option is now part of the Base.  
51692

51693 The [EINVAL] error for an uninitialized thread attributes object is removed; this condition  
51694 results in undefined behavior.

51695 **NAME**

51696 pthread\_attr\_init — initialize the thread attributes object

51697 **SYNOPSIS**

51698 #include <pthread.h>

51699 int pthread\_attr\_init(pthread\_attr\_t \*attr);

51700 **DESCRIPTION**

51701 Refer to *pthread\_attr\_destroy()*.

51702 **NAME**

51703 pthread\_attr\_setdetachstate — set the detachstate attribute

51704 **SYNOPSIS**

51705 #include &lt;pthread.h&gt;

51706 int pthread\_attr\_setdetachstate(pthread\_attr\_t \*attr, int detachstate);

51707 **DESCRIPTION**51708 Refer to *pthread\_attr\_getdetachstate()*.

51709 **NAME**

51710 pthread\_attr\_setguardsize — set the thread guardsize attribute

51711 **SYNOPSIS**

51712 #include &lt;pthread.h&gt;

51713 int pthread\_attr\_setguardsize(pthread\_attr\_t \*attr,  
51714 size\_t guardsize);51715 **DESCRIPTION**51716 Refer to *pthread\_attr\_getguardsize()*.

51717 **NAME**51718 pthread\_attr\_setinheritsched — set the inheritsched attribute (**REALTIME THREADS**)51719 **SYNOPSIS**

```
51720 TPS    #include <pthread.h>
51721
51721    int pthread_attr_setinheritsched(pthread_attr_t *attr,
51722    int inheritsched);
```

51723 **DESCRIPTION**51724 Refer to *pthread\_attr\_getinheritsched()*.

51725 **NAME**

51726 pthread\_attr\_setschedparam — set the schedparam attribute

51727 **SYNOPSIS**

51728 #include &lt;pthread.h&gt;

```
51729 int pthread_attr_setschedparam(pthread_attr_t *restrict attr,  
51730                               const struct sched_param *restrict param);
```

51731 **DESCRIPTION**51732 Refer to *pthread\_attr\_getschedparam()*.



51733 **NAME**51734 pthread\_attr\_setschedpolicy — set the schedpolicy attribute (**REALTIME THREADS**)51735 **SYNOPSIS**

```
51736 TPS    #include <pthread.h>
51737        int pthread_attr_setschedpolicy(pthread_attr_t *attr, int policy);
```

51738 **DESCRIPTION**51739 Refer to *pthread\_attr\_getschedpolicy()*.

51740 **NAME**51741 pthread\_attr\_setscope — set the contention scope attribute (**REALTIME THREADS**)51742 **SYNOPSIS**51743 TPS `#include <pthread.h>`51744 `int pthread_attr_setscope(pthread_attr_t *attr, int contention_scope);`51745 **DESCRIPTION**51746 Refer to *pthread\_attr\_getscope()*.

51747 **NAME**

51748 pthread\_attr\_setstack — set the stack attribute

51749 **SYNOPSIS**51750 TSA TSS `#include <pthread.h>`51751 `int pthread_attr_setstack(pthread_attr_t *attr, void *stackaddr,`  
51752 `size_t stacksize);`51753 **DESCRIPTION**51754 Refer to *pthread\_attr\_getstack()*.

51755 **NAME**

51756 pthread\_attr\_setstacksize — set the stacksize attribute

51757 **SYNOPSIS**51758 TSS `#include <pthread.h>`51759 `int pthread_attr_setstacksize(pthread_attr_t *attr, size_t stacksize);`51760 **DESCRIPTION**51761 Refer to *pthread\_attr\_getstacksize()*.

51762 **NAME**

51763 pthread\_barrier\_destroy, pthread\_barrier\_init — destroy and initialize a barrier object

51764 **SYNOPSIS**

```
51765 #include <pthread.h>
51766 int pthread_barrier_destroy(pthread_barrier_t *barrier);
51767 int pthread_barrier_init(pthread_barrier_t *restrict barrier,
51768     const pthread_barrierattr_t *restrict attr, unsigned count);
```

51769 **DESCRIPTION**

51770 The *pthread\_barrier\_destroy()* function shall destroy the barrier referenced by *barrier* and release  
 51771 any resources used by the barrier. The effect of subsequent use of the barrier is undefined until  
 51772 the barrier is reinitialized by another call to *pthread\_barrier\_init()*. An implementation may use  
 51773 this function to set *barrier* to an invalid value. The results are undefined if  
 51774 *pthread\_barrier\_destroy()* is called when any thread is blocked on the barrier, or if this function is  
 51775 called with an uninitialized barrier.

51776 The *pthread\_barrier\_init()* function shall allocate any resources required to use the barrier  
 51777 referenced by *barrier* and shall initialize the barrier with attributes referenced by *attr*. If *attr* is  
 51778 NULL, the default barrier attributes shall be used; the effect is the same as passing the address of  
 51779 a default barrier attributes object. The results are undefined if *pthread\_barrier\_init()* is called  
 51780 when any thread is blocked on the barrier (that is, has not returned from the  
 51781 *pthread\_barrier\_wait()* call). The results are undefined if a barrier is used without first being  
 51782 initialized. The results are undefined if *pthread\_barrier\_init()* is called specifying an already  
 51783 initialized barrier.

51784 The *count* argument specifies the number of threads that must call *pthread\_barrier\_wait()* before  
 51785 any of them successfully return from the call. The value specified by *count* must be greater than  
 51786 zero.

51787 If the *pthread\_barrier\_init()* function fails, the barrier shall not be initialized and the contents of  
 51788 *barrier* are undefined.

51789 See [Section 2.9.9](#) (on page 523) for further requirements. |

51790 **RETURN VALUE**

51791 Upon successful completion, these functions shall return zero; otherwise, an error number shall  
 51792 be returned to indicate the error.

51793 **ERRORS**

51794 The *pthread\_barrier\_init()* function shall fail if:

51795 [EAGAIN] The system lacks the necessary resources to initialize another barrier.

51796 [EINVAL] The value specified by *count* is equal to zero.

51797 [ENOMEM] Insufficient memory exists to initialize the barrier.

51798 These functions shall not return an error code of [EINTR].

51799 **EXAMPLES**

51800 None.

51801 **APPLICATION USAGE**

51802 None.

51803 **RATIONALE**

51804 If an implementation detects that the value specified by the *barrier* argument to  
 51805 *pthread\_barrier\_destroy()* does not refer to an initialized barrier object, it is recommended that the  
 51806 function should fail and report an [EINVAL] error.

51807 If an implementation detects that the value specified by the *attr* argument to  
 51808 *pthread\_barrier\_init()* does not refer to an initialized barrier attributes object, it is recommended  
 51809 that the function should fail and report an [EINVAL] error.

51810 If an implementation detects that the value specified by the *barrier* argument to  
 51811 *pthread\_barrier\_destroy()* or *pthread\_barrier\_init()* refers to a barrier that is in use (for example, in  
 51812 a *pthread\_barrier\_wait()* call) by another thread, or detects that the value specified by the *barrier*  
 51813 argument to *pthread\_barrier\_init()* refers to an already initialized barrier object, it is  
 51814 recommended that the function should fail and report an [EBUSY] error.

51815 **FUTURE DIRECTIONS**

51816 None.

51817 **SEE ALSO**51818 *pthread\_barrier\_wait()*

51819 XBD &lt;pthread.h&gt;

51820 **CHANGE HISTORY**

51821 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

51822 **Issue 7**

51823 The *pthread\_barrier\_destroy()* and *pthread\_barrier\_init()* functions are moved from the Barriers  
 51824 option to the Base.

51825 The [EINVAL] error for an uninitialized barrier object and an uninitialized barrier attributes  
 51826 object is removed; this condition results in undefined behavior.

51827 The [EBUSY] error for a barrier that is in use or an already initialized barrier object is removed;  
 51828 this condition results in undefined behavior.

**51829 NAME**

51830 pthread\_barrier\_wait — synchronize at a barrier

**51831 SYNOPSIS**

51832 #include <pthread.h>

51833 int pthread\_barrier\_wait(pthread\_barrier\_t \*barrier);

**51834 DESCRIPTION**

51835 The *pthread\_barrier\_wait()* function shall synchronize participating threads at the barrier  
51836 referenced by *barrier*. The calling thread shall block until the required number of threads have  
51837 called *pthread\_barrier\_wait()* specifying the barrier.

51838 When the required number of threads have called *pthread\_barrier\_wait()* specifying the barrier,  
51839 the constant PTHREAD\_BARRIER\_SERIAL\_THREAD shall be returned to one unspecified  
51840 thread and zero shall be returned to each of the remaining threads. At this point, the barrier shall  
51841 be reset to the state it had as a result of the most recent *pthread\_barrier\_init()* function that  
51842 referenced it.

51843 The constant PTHREAD\_BARRIER\_SERIAL\_THREAD is defined in <pthread.h> and its value  
51844 shall be distinct from any other value returned by *pthread\_barrier\_wait()*.

51845 The results are undefined if this function is called with an uninitialized barrier.

51846 If a signal is delivered to a thread blocked on a barrier, upon return from the signal handler the  
51847 thread shall resume waiting at the barrier if the barrier wait has not completed (that is, if the  
51848 required number of threads have not arrived at the barrier during the execution of the signal  
51849 handler); otherwise, the thread shall continue as normal from the completed barrier wait. Until  
51850 the thread in the signal handler returns from it, it is unspecified whether other threads may  
51851 proceed past the barrier once they have all reached it.

51852 A thread that has blocked on a barrier shall not prevent any unblocked thread that is eligible to  
51853 use the same processing resources from eventually making forward progress in its execution.  
51854 Eligibility for processing resources shall be determined by the scheduling policy.

**51855 RETURN VALUE**

51856 Upon successful completion, the *pthread\_barrier\_wait()* function shall return  
51857 PTHREAD\_BARRIER\_SERIAL\_THREAD for a single (arbitrary) thread synchronized at the  
51858 barrier and zero for each of the other threads. Otherwise, an error number shall be returned to  
51859 indicate the error.

**51860 ERRORS**

51861 This function shall not return an error code of [EINTR].

**51862 EXAMPLES**

51863 None.

**51864 APPLICATION USAGE**

51865 Applications using this function may be subject to priority inversion, as discussed in XBD  
51866 [Section 3.291](#) (on page 80).

**51867 RATIONALE**

51868 If an implementation detects that the value specified by the *barrier* argument to  
51869 *pthread\_barrier\_wait()* does not refer to an initialized barrier object, it is recommended that the  
51870 function should fail and report an [EINVAL] error.

51871 **FUTURE DIRECTIONS**

51872 None.

51873 **SEE ALSO**

51874 *pthread\_barrier\_destroy()*

51875 XBD [Section 3.291](#) (on page 80), [Section 4.12](#) (on page 111), [<pthread.h>](#)

51876 **CHANGE HISTORY**

51877 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

51878 In the SYNOPSIS, the inclusion of [<sys/types.h>](#) is no longer required.

51879 **Issue 7**

51880 The *pthread\_barrier\_wait()* function is moved from the Barriers option to the Base.

51881 The [EINVAL] error for an uninitialized barrier object is removed; this condition results in  
51882 undefined behavior.



**51883 NAME**

51884 pthread\_barrierattr\_destroy, pthread\_barrierattr\_init — destroy and initialize the barrier  
 51885 attributes object

**51886 SYNOPSIS**

51887 #include <pthread.h>  
 51888 int pthread\_barrierattr\_destroy(pthread\_barrierattr\_t \*attr);  
 51889 int pthread\_barrierattr\_init(pthread\_barrierattr\_t \*attr);

**51890 DESCRIPTION**

51891 The *pthread\_barrierattr\_destroy()* function shall destroy a barrier attributes object. A destroyed  
 51892 *attr* attributes object can be reinitialized using *pthread\_barrierattr\_init()*; the results of otherwise  
 51893 referencing the object after it has been destroyed are undefined. An implementation may cause  
 51894 *pthread\_barrierattr\_destroy()* to set the object referenced by *attr* to an invalid value.

51895 The *pthread\_barrierattr\_init()* function shall initialize a barrier attributes object *attr* with the  
 51896 default value for all of the attributes defined by the implementation.

51897 If *pthread\_barrierattr\_init()* is called specifying an already initialized *attr* attributes object, the  
 51898 results are undefined.

51899 After a barrier attributes object has been used to initialize one or more barriers, any function  
 51900 affecting the attributes object (including destruction) shall not affect any previously initialized  
 51901 barrier.

51902 The behavior is undefined if the value specified by the *attr* argument to  
 51903 *pthread\_barrierattr\_destroy()* does not refer to an initialized barrier attributes object.

**51904 RETURN VALUE**

51905 If successful, the *pthread\_barrierattr\_destroy()* and *pthread\_barrierattr\_init()* functions shall return  
 51906 zero; otherwise, an error number shall be returned to indicate the error.

**51907 ERRORS**

51908 The *pthread\_barrierattr\_init()* function shall fail if:

51909 [ENOMEM] Insufficient memory exists to initialize the barrier attributes object.

51910 These functions shall not return an error code of [EINTR].

**51911 EXAMPLES**

51912 None.

**51913 APPLICATION USAGE**

51914 None.

**51915 RATIONALE**

51916 If an implementation detects that the value specified by the *attr* argument to  
 51917 *pthread\_barrierattr\_destroy()* does not refer to an initialized barrier attributes object, it is  
 51918 recommended that the function should fail and report an [EINVAL] error.

**51919 FUTURE DIRECTIONS**

51920 None.

**51921 SEE ALSO**

51922 *pthread\_barrierattr\_getpshared()*

51923 XBD <pthread.h>

51924 **CHANGE HISTORY**

51925 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

51926 In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.

51927 **Issue 7**

51928 The *pthread\_barrierattr\_destroy()* and *pthread\_barrierattr\_init()* functions are moved from the  
51929 Barriers option to the Base.

51930 The [EINVAL] error for an uninitialized barrier attributes object is removed; this condition  
51931 results in undefined behavior.

**51932 NAME**

51933 pthread\_barrierattr\_getpshared, pthread\_barrierattr\_setpshared — get and set the process-  
 51934 shared attribute of the barrier attributes object

**51935 SYNOPSIS**

```
51936 TSH #include <pthread.h>
51937
51937 int pthread_barrierattr_getpshared(const pthread_barrierattr_t
51938     *restrict attr, int *restrict pshared);
51939 int pthread_barrierattr_setpshared(pthread_barrierattr_t *attr,
51940     int pshared);
```

**51941 DESCRIPTION**

51942 The *pthread\_barrierattr\_getpshared()* function shall obtain the value of the *process-shared* attribute  
 51943 from the attributes object referenced by *attr*. The *pthread\_barrierattr\_setpshared()* function shall  
 51944 set the *process-shared* attribute in an initialized attributes object referenced by *attr*.

51945 The *process-shared* attribute is set to PTHREAD\_PROCESS\_SHARED to permit a barrier to be  
 51946 operated upon by any thread that has access to the memory where the barrier is allocated. See  
 51947 [Section 2.9.9](#) (on page 523) for further requirements. The default value of the attribute shall be  
 51948 PTHREAD\_PROCESS\_PRIVATE. Both constants PTHREAD\_PROCESS\_SHARED and  
 51949 PTHREAD\_PROCESS\_PRIVATE are defined in **<pthread.h>**.

51950 Additional attributes, their default values, and the names of the associated functions to get and  
 51951 set those attribute values are implementation-defined.

51952 The behavior is undefined if the value specified by the *attr* argument to  
 51953 *pthread\_barrierattr\_getpshared()* or *pthread\_barrierattr\_setpshared()* does not refer to an initialized  
 51954 barrier attributes object.

**51955 RETURN VALUE**

51956 If successful, the *pthread\_barrierattr\_getpshared()* function shall return zero and store the value of  
 51957 the *process-shared* attribute of *attr* into the object referenced by the *pshared* parameter. Otherwise,  
 51958 an error number shall be returned to indicate the error.

51959 If successful, the *pthread\_barrierattr\_setpshared()* function shall return zero; otherwise, an error  
 51960 number shall be returned to indicate the error.

**51961 ERRORS**

51962 The *pthread\_barrierattr\_setpshared()* function may fail if:

51963 [EINVAL] The new value specified for the *process-shared* attribute is not one of the legal  
 51964 values PTHREAD\_PROCESS\_SHARED or PTHREAD\_PROCESS\_PRIVATE.

51965 These functions shall not return an error code of [EINTR].

**51966 EXAMPLES**

51967 None.

**51968 APPLICATION USAGE**

51969 The *pthread\_barrierattr\_getpshared()* and *pthread\_barrierattr\_setpshared()* functions are part of the  
 51970 Thread Process-Shared Synchronization option and need not be provided on all  
 51971 implementations.

**51972 RATIONALE**

51973 If an implementation detects that the value specified by the *attr* argument to  
 51974 *pthread\_barrierattr\_getpshared()* or *pthread\_barrierattr\_setpshared()* does not refer to an initialized  
 51975 barrier attributes object, it is recommended that the function should fail and report an [EINVAL]

51976 error.

51977 **FUTURE DIRECTIONS**

51978 None.

51979 **SEE ALSO**

51980 *pthread\_barrier\_destroy()*, *pthread\_barrierattr\_destroy()*

51981 XBD <pthread.h>

51982 **CHANGE HISTORY**

51983 First released in Issue 6. Derived from IEEE Std 1003.1j-2000

51984 **Issue 7**

51985 The *pthread\_barrierattr\_getpshared()* and *pthread\_barrierattr\_setpshared()* functions are marked |  
51986 only as part of the Thread Process-Shared Synchronization option as the Threads option is now |  
51987 part of the Base.

51988 The [EINVAL] error for an uninitialized barrier attributes object is removed; this condition  
51989 results in undefined behavior.

51990 **NAME**

51991 pthread\_barrierattr\_init — initialize the barrier attributes object

51992 **SYNOPSIS**

51993 #include &lt;pthread.h&gt;

51994 int pthread\_barrierattr\_init(pthread\_barrierattr\_t \*attr);

51995 **DESCRIPTION**51996 Refer to *pthread\_barrierattr\_destroy()*.

51997 **NAME**

51998 pthread\_barrierattr\_setpshared — set the process-shared attribute of the barrier attributes object

51999 **SYNOPSIS**

```
52000 TSH      #include <pthread.h>
52001          int pthread_barrierattr_setpshared(pthread_barrierattr_t *attr,
52002          int pshared);
```

52003 **DESCRIPTION**

52004 Refer to *pthread\_barrierattr\_getpshared()*.

52005 **NAME**

52006 pthread\_cancel — cancel execution of a thread

52007 **SYNOPSIS**

52008 #include &lt;pthread.h&gt;

52009 int pthread\_cancel(pthread\_t thread);

52010 **DESCRIPTION**

52011 The *pthread\_cancel()* function shall request that *thread* be canceled. The target thread's  
52012 cancelability state and type determines when the cancellation takes effect. When the cancellation  
52013 is acted on, the cancellation cleanup handlers for *thread* shall be called. When the last  
52014 cancellation cleanup handler returns, the thread-specific data destructor functions shall be called  
52015 for *thread*. When the last destructor function returns, *thread* shall be terminated.

52016 The cancellation processing in the target thread shall run asynchronously with respect to the  
52017 calling thread returning from *pthread\_cancel()*.

52018 **RETURN VALUE**

52019 If successful, the *pthread\_cancel()* function shall return zero; otherwise, an error number shall be  
52020 returned to indicate the error.

52021 **ERRORS**52022 The *pthread\_cancel()* function shall not return an error code of [EINTR].52023 **EXAMPLES**

52024 None.

52025 **APPLICATION USAGE**

52026 None.

52027 **RATIONALE**

52028 Two alternative functions were considered for sending the cancellation notification to a thread.  
52029 One would be to define a new SIGCANCEL signal that had the cancellation semantics when  
52030 delivered; the other was to define the new *pthread\_cancel()* function, which would trigger the  
52031 cancellation semantics.

52032 The advantage of a new signal was that so much of the delivery criteria were identical to that  
52033 used when trying to deliver a signal that making cancellation notification a signal was seen as  
52034 consistent. Indeed, many implementations implement cancellation using a special signal. On the  
52035 other hand, there would be no signal functions that could be used with this signal except  
52036 *pthread\_kill()*, and the behavior of the delivered cancellation signal would be unlike any  
52037 previously existing defined signal.

52038 The benefits of a special function include the recognition that this signal would be defined  
52039 because of the similar delivery criteria and that this is the only common behavior between a  
52040 cancellation request and a signal. In addition, the cancellation delivery mechanism does not  
52041 have to be implemented as a signal. There are also strong, if not stronger, parallels with  
52042 language exception mechanisms than with signals that are potentially obscured if the delivery  
52043 mechanism is visibly closer to signals.

52044 In the end, it was considered that as there were so many exceptions to the use of the new signal  
52045 with existing signals functions it would be misleading. A special function has resolved this  
52046 problem. This function was carefully defined so that an implementation wishing to provide the  
52047 cancellation functions on top of signals could do so. The special function also means that  
52048 implementations are not obliged to implement cancellation with signals.

52049 If an implementation detects use of a thread ID after the end of its lifetime, it is recommended  
52050 that the function should fail and report an [ESRCH] error.

52051 **FUTURE DIRECTIONS**

52052 None.

52053 **SEE ALSO**52054 *pthread\_exit()*, *pthread\_cond\_timedwait()*, *pthread\_join()*, *pthread\_setcancelstate()*

52055 XBD &lt;pthread.h&gt;

52056 **CHANGE HISTORY**

52057 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

52058 **Issue 6**52059 The *pthread\_cancel()* function is marked as part of the Threads option.52060 **Issue 7**52061 The *pthread\_cancel()* function is moved from the Threads option to the Base.

52062 Austin Group Interpretation 1003.1-2001 #142 is applied, removing the [ESRCH] error condition.



**NAME**

pthread\_cleanup\_pop, pthread\_cleanup\_push — establish cancellation handlers

**SYNOPSIS**

```
#include <pthread.h>

void pthread_cleanup_pop(int execute);
void pthread_cleanup_push(void (*routine)(void*), void *arg);
```

**DESCRIPTION**

The *pthread\_cleanup\_pop()* function shall remove the routine at the top of the calling thread's cancellation cleanup stack and optionally invoke it (if *execute* is non-zero).

The *pthread\_cleanup\_push()* function shall push the specified cancellation cleanup handler *routine* onto the calling thread's cancellation cleanup stack. The cancellation cleanup handler shall be popped from the cancellation cleanup stack and invoked with the argument *arg* when:

- The thread exits (that is, calls *pthread\_exit()*).
- The thread acts upon a cancellation request.
- The thread calls *pthread\_cleanup\_pop()* with a non-zero *execute* argument.

It is unspecified whether *pthread\_cleanup\_push()* and *pthread\_cleanup\_pop()* are macros or functions. If a macro definition is suppressed in order to access an actual function, or a program defines an external identifier with any of these names, the behavior is undefined. The application shall ensure that they appear as statements, and in pairs within the same lexical scope (that is, the *pthread\_cleanup\_push()* macro may be thought to expand to a token list whose first token is '`{`' with *pthread\_cleanup\_pop()* expanding to a token list whose last token is the corresponding '`}`').

The effect of calling *longjmp()* or *siglongjmp()* is undefined if there have been any calls to *pthread\_cleanup\_push()* or *pthread\_cleanup\_pop()* made without the matching call since the jump buffer was filled. The effect of calling *longjmp()* or *siglongjmp()* from inside a cancellation cleanup handler is also undefined unless the jump buffer was also filled in the cancellation cleanup handler.

The effect of the use of **return**, **break**, **continue**, and **goto** to prematurely leave a code block described by a pair of *pthread\_cleanup\_push()* and *pthread\_cleanup\_pop()* functions calls is undefined.

**RETURN VALUE**

The *pthread\_cleanup\_push()* and *pthread\_cleanup\_pop()* functions shall not return a value.

**ERRORS**

No errors are defined.

These functions shall not return an error code of [EINTR].

**EXAMPLES**

The following is an example using thread primitives to implement a cancelable, writers-priority read-write lock:

```
typedef struct {
    pthread_mutex_t lock;
    pthread_cond_t rcond,
        wcond;
    int lock_count; /* < 0 .. Held by writer. */
                  /* > 0 .. Held by lock_count readers. */
                  /* = 0 .. Held by nobody. */
```

```
52108     int waiting_writers; /* Count of waiting writers. */
52109 } rwlock;

52110 void
52111 waiting_reader_cleanup(void *arg)
52112 {
52113     rwlock *l;
52114
52115     l = (rwlock *) arg;
52116     pthread_mutex_unlock(&l->lock);
52117 }
52118 void
52119 lock_for_read(rwlock *l)
52120 {
52121     pthread_mutex_lock(&l->lock);
52122     pthread_cleanup_push(waiting_reader_cleanup, l);
52123     while ((l->lock_count < 0) || (l->waiting_writers != 0))
52124         pthread_cond_wait(&l->rcond, &l->lock);
52125     l->lock_count++;
52126     /*
52127      * Note the pthread_cleanup_pop executes
52128      * waiting_reader_cleanup.
52129      */
52130     pthread_cleanup_pop(1);
52131 }
52132 void
52133 release_read_lock(rwlock *l)
52134 {
52135     pthread_mutex_lock(&l->lock);
52136     if (--l->lock_count == 0)
52137         pthread_cond_signal(&l->wcond);
52138     pthread_mutex_unlock(&l->lock);
52139 }
52140 void
52141 waiting_writer_cleanup(void *arg)
52142 {
52143     rwlock *l;
52144
52145     l = (rwlock *) arg;
52146     if (--l->waiting_writers == 0) && (l->lock_count >= 0) {
52147         /*
52148          * This only happens if we have been canceled. If the
52149          * lock is not held by a writer, there may be readers who
52150          * were blocked because waiting_writers was positive; they
52151          * can now be unblocked.
52152          */
52153         pthread_cond_broadcast(&l->rcond);
52154     }
52155     pthread_mutex_unlock(&l->lock);
52156 }
52157 void
```

```
52156     lock_for_write(rwlock *l)
52157     {
52158         pthread_mutex_lock(&l->lock);
52159         l->waiting_writers++;
52160         pthread_cleanup_push(waiting_writer_cleanup, l);
52161         while (l->lock_count != 0)
52162             pthread_cond_wait(&l->wcond, &l->lock);
52163         l->lock_count = -1;
52164         /*
52165          * Note the pthread_cleanup_pop executes
52166          * waiting_writer_cleanup.
52167          */
52168         pthread_cleanup_pop(1);
52169     }
52170
52171 void
52172 release_write_lock(rwlock *l)
52173 {
52174     pthread_mutex_lock(&l->lock);
52175     l->lock_count = 0;
52176     if (l->waiting_writers == 0)
52177         pthread_cond_broadcast(&l->rcond);
52178     else
52179         pthread_cond_signal(&l->wcond);
52180     pthread_mutex_unlock(&l->lock);
52181 }
52182 /*
52183  * This function is called to initialize the read/write lock.
52184  */
52185 void
52186 initialize_rwlock(rwlock *l)
52187 {
52188     pthread_mutex_init(&l->lock, pthread_mutexattr_default);
52189     pthread_cond_init(&l->wcond, pthread_condattr_default);
52190     pthread_cond_init(&l->rcond, pthread_condattr_default);
52191     l->lock_count = 0;
52192     l->waiting_writers = 0;
52193 }
52194
52195 reader_thread()
52196 {
52197     lock_for_read(&lock);
52198     pthread_cleanup_push(release_read_lock, &lock);
52199     /*
52200      * Thread has read lock.
52201      */
52202     pthread_cleanup_pop(1);
52203 }
52204
52205 writer_thread()
52206 {
52207     lock_for_write(&lock);
52208     pthread_cleanup_push(release_write_lock, &lock);
```

```

52206      /*
52207      * Thread has write lock.
52208      */
52209      pthread_cleanup_pop(1);
52210  }

```

## APPLICATION USAGE

The two routines that push and pop cancellation cleanup handlers, *pthread\_cleanup\_push()* and *pthread\_cleanup\_pop()*, can be thought of as left and right-parentheses. They always need to be matched.

## RATIONALE

The restriction that the two routines that push and pop cancellation cleanup handlers, *pthread\_cleanup\_push()* and *pthread\_cleanup\_pop()*, have to appear in the same lexical scope allows for efficient macro or compiler implementations and efficient storage management. A sample implementation of these routines as macros might look like this:

```

52220  #define pthread_cleanup_push(rtn,arg) { \
52221      struct _pthread_handler_rec __cleanup_handler, *__head; \
52222      __cleanup_handler.rtn = rtn; \
52223      __cleanup_handler.arg = arg; \
52224      (void) pthread_getspecific(_pthread_handler_key, &__head); \
52225      __cleanup_handler.next = *__head; \
52226      *__head = &__cleanup_handler;
52227
52228  #define pthread_cleanup_pop(ex) \
52229      *__head = __cleanup_handler.next; \
52230      if (ex) (*__cleanup_handler.rtn)(__cleanup_handler.arg); \
52231  }

```

A more ambitious implementation of these routines might do even better by allowing the compiler to note that the cancellation cleanup handler is a constant and can be expanded inline.

This volume of POSIX.1-2008 currently leaves unspecified the effect of calling *longjmp()* from a signal handler executing in a POSIX System Interfaces function. If an implementation wants to allow this and give the programmer reasonable behavior, the *longjmp()* function has to call all cancellation cleanup handlers that have been pushed but not popped since the time *setjmp()* was called.

Consider a multi-threaded function called by a thread that uses signals. If a signal were delivered to a signal handler during the operation of *qsort()* and that handler were to call *longjmp()* (which, in turn, did *not* call the cancellation cleanup handlers) the helper threads created by the *qsort()* function would not be canceled. Instead, they would continue to execute and write into the argument array even though the array might have been popped off the stack.

Note that the specified cleanup handling mechanism is especially tied to the C language and, while the requirement for a uniform mechanism for expressing cleanup is language-independent, the mechanism used in other languages may be quite different. In addition, this mechanism is really only necessary due to the lack of a real exception mechanism in the C language, which would be the ideal solution.

There is no notion of a cancellation cleanup-safe function. If an application has no cancellation points in its signal handlers, blocks any signal whose handler may have cancellation points while calling async-unsafe functions, or disables cancellation while calling async-unsafe functions, all functions may be safely called from cancellation cleanup routines.

**52252 FUTURE DIRECTIONS**

52253 None.

**52254 SEE ALSO**

52255 *pthread\_cancel()*, *pthread\_setcancelstate()*

52256 XBD **<pthread.h>**

**52257 CHANGE HISTORY**

52258 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

**52259 Issue 6**

52260 The *pthread\_cleanup\_pop()* and *pthread\_cleanup\_push()* functions are marked as part of the  
52261 Threads option.

52262 The APPLICATION USAGE section is added.

52263 The normative text is updated to avoid use of the term “must” for application requirements.

52264 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/88 is applied, updating the  
52265 DESCRIPTION to describe the consequences of prematurely leaving a code block defined by the  
52266 *pthread\_cleanup\_push()* and *pthread\_cleanup\_pop()* functions.

**52267 Issue 7**

52268 The *pthread\_cleanup\_pop()* and *pthread\_cleanup\_push()* functions are moved from the Threads  
52269 option to the Base.

52270 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0454 [229] is applied.

## 52271 NAME

52272 pthread\_cond\_broadcast, pthread\_cond\_signal — broadcast or signal a condition

## 52273 SYNOPSIS

```
52274 #include <pthread.h>
52275 int pthread_cond_broadcast(pthread_cond_t *cond);
52276 int pthread_cond_signal(pthread_cond_t *cond);
```

## 52277 DESCRIPTION

52278 These functions shall unblock threads blocked on a condition variable.

52279 The *pthread\_cond\_broadcast()* function shall unblock all threads currently blocked on the  
52280 specified condition variable *cond*.

52281 The *pthread\_cond\_signal()* function shall unblock at least one of the threads that are blocked on  
52282 the specified condition variable *cond* (if any threads are blocked on *cond*).

52283 If more than one thread is blocked on a condition variable, the scheduling policy shall determine  
52284 the order in which threads are unblocked. When each thread unblocked as a result of a  
52285 *pthread\_cond\_broadcast()* or *pthread\_cond\_signal()* returns from its call to *pthread\_cond\_wait()* or  
52286 *pthread\_cond\_timedwait()*, the thread shall own the mutex with which it called  
52287 *pthread\_cond\_wait()* or *pthread\_cond\_timedwait()*. The thread(s) that are unblocked shall contend  
52288 for the mutex according to the scheduling policy (if applicable), and as if each had called  
52289 *pthread\_mutex\_lock()*.

52290 The *pthread\_cond\_broadcast()* or *pthread\_cond\_signal()* functions may be called by a thread  
52291 whether or not it currently owns the mutex that threads calling *pthread\_cond\_wait()* or  
52292 *pthread\_cond\_timedwait()* have associated with the condition variable during their waits;  
52293 however, if predictable scheduling behavior is required, then that mutex shall be locked by the  
52294 thread calling *pthread\_cond\_broadcast()* or *pthread\_cond\_signal()*.

52295 The *pthread\_cond\_broadcast()* and *pthread\_cond\_signal()* functions shall have no effect if there are  
52296 no threads currently blocked on *cond*.

52297 The behavior is undefined if the value specified by the *cond* argument to *pthread\_cond\_broadcast()*  
52298 or *pthread\_cond\_signal()* does not refer to an initialized condition variable.

## 52299 RETURN VALUE

52300 If successful, the *pthread\_cond\_broadcast()* and *pthread\_cond\_signal()* functions shall return zero;  
52301 otherwise, an error number shall be returned to indicate the error.

## 52302 ERRORS

52303 These functions shall not return an error code of [EINTR].

## 52304 EXAMPLES

52305 None.

## 52306 APPLICATION USAGE

52307 The *pthread\_cond\_broadcast()* function is used whenever the shared-variable state has been  
52308 changed in a way that more than one thread can proceed with its task. Consider a single  
52309 producer/multiple consumer problem, where the producer can insert multiple items on a list  
52310 that is accessed one item at a time by the consumers. By calling the *pthread\_cond\_broadcast()*  
52311 function, the producer would notify all consumers that might be waiting, and thereby the  
52312 application would receive more throughput on a multi-processor. In addition,  
52313 *pthread\_cond\_broadcast()* makes it easier to implement a read-write lock. The  
52314 *pthread\_cond\_broadcast()* function is needed in order to wake up all waiting readers when a  
52315 writer releases its lock. Finally, the two-phase commit algorithm can use this broadcast function  
52316 to notify all clients of an impending transaction commit.

It is not safe to use the *pthread\_cond\_signal()* function in a signal handler that is invoked asynchronously. Even if it were safe, there would still be a race between the test of the Boolean *pthread\_cond\_wait()* that could not be efficiently eliminated.

Mutexes and condition variables are thus not suitable for releasing a waiting thread by signaling from code running in a signal handler.

## RATIONALE

If an implementation detects that the value specified by the *cond* argument to *pthread\_cond\_broadcast()* or *pthread\_cond\_signal()* does not refer to an initialized condition variable, it is recommended that the function should fail and report an [EINVAL] error.

## Multiple Awakenings by Condition Signal

On a multi-processor, it may be impossible for an implementation of *pthread\_cond\_signal()* to avoid the unblocking of more than one thread blocked on a condition variable. For example, consider the following partial implementation of *pthread\_cond\_wait()* and *pthread\_cond\_signal()*, executed by two threads in the order given. One thread is trying to wait on the condition variable, another is concurrently executing *pthread\_cond\_signal()*, while a third thread is already waiting.

```
pthread_cond_wait(mutex, cond):
    value = cond->value; /* 1 */
    pthread_mutex_unlock(mutex); /* 2 */
    pthread_mutex_lock(cond->mutex); /* 10 */
    if (value == cond->value) { /* 11 */
        me->next_cond = cond->waiter;
        cond->waiter = me;
        pthread_mutex_unlock(cond->mutex);
        unable_to_run(me);
    } else
        pthread_mutex_unlock(cond->mutex); /* 12 */
    pthread_mutex_lock(mutex); /* 13 */

pthread_cond_signal(cond):
    pthread_mutex_lock(cond->mutex); /* 3 */
    cond->value++; /* 4 */
    if (cond->waiter) { /* 5 */
        sleeper = cond->waiter; /* 6 */
        cond->waiter = sleeper->next_cond; /* 7 */
        able_to_run(sleeper); /* 8 */
    }
    pthread_mutex_unlock(cond->mutex); /* 9 */
```

The effect is that more than one thread can return from its call to *pthread\_cond\_wait()* or *pthread\_cond\_timedwait()* as a result of one call to *pthread\_cond\_signal()*. This effect is called “spurious wakeup”. Note that the situation is self-correcting in that the number of threads that are so awakened is finite; for example, the next thread to call *pthread\_cond\_wait()* after the sequence of events above blocks.

While this problem could be resolved, the loss of efficiency for a fringe condition that occurs only rarely is unacceptable, especially given that one has to check the predicate associated with a condition variable anyway. Correcting this problem would unnecessarily reduce the degree of concurrency in this basic building block for all higher-level synchronization operations.

An added benefit of allowing spurious wakeups is that applications are forced to code a

52364 predicate-testing-loop around the condition wait. This also makes the application tolerate  
52365 superfluous condition broadcasts or signals on the same condition variable that may be coded in  
52366 some other part of the application. The resulting applications are thus more robust. Therefore,  
52367 POSIX.1-2008 explicitly documents that spurious wakeups may occur.

52368 **FUTURE DIRECTIONS**

52369 None.

52370 **SEE ALSO**

52371 *pthread\_cond\_destroy()*, *pthread\_cond\_timedwait()*

52372 XBD Section 4.12 (on page 111), **<pthread.h>**

52373 **CHANGE HISTORY**

52374 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

52375 **Issue 6**

52376 The *pthread\_cond\_broadcast()* and *pthread\_cond\_signal()* functions are marked as part of the  
52377 Threads option.

52378 The APPLICATION USAGE section is added.

52379 **Issue 7**

52380 The *pthread\_cond\_broadcast()* and *pthread\_cond\_signal()* functions are moved from the Threads  
52381 option to the Base.

52382 The [EINVAL] error for an uninitialized condition variable is removed; this condition results in  
52383 undefined behavior.



**NAME**

pthread\_cond\_destroy, pthread\_cond\_init — destroy and initialize condition variables

**SYNOPSIS**

```
#include <pthread.h>

int pthread_cond_destroy(pthread_cond_t *cond);
int pthread_cond_init(pthread_cond_t *restrict cond,
    const pthread_condattr_t *restrict attr);
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
```

**DESCRIPTION**

The *pthread\_cond\_destroy()* function shall destroy the given condition variable specified by *cond*; the object becomes, in effect, uninitialized. An implementation may cause *pthread\_cond\_destroy()* to set the object referenced by *cond* to an invalid value. A destroyed condition variable object can be reinitialized using *pthread\_cond\_init()*; the results of otherwise referencing the object after it has been destroyed are undefined.

It shall be safe to destroy an initialized condition variable upon which no threads are currently blocked. Attempting to destroy a condition variable upon which other threads are currently blocked results in undefined behavior.

The *pthread\_cond\_init()* function shall initialize the condition variable referenced by *cond* with attributes referenced by *attr*. If *attr* is NULL, the default condition variable attributes shall be used; the effect is the same as passing the address of a default condition variable attributes object. Upon successful initialization, the state of the condition variable shall become initialized.

See [Section 2.9.9](#) (on page 523) for further requirements.

Attempting to initialize an already initialized condition variable results in undefined behavior.

In cases where default condition variable attributes are appropriate, the macro PTHREAD\_COND\_INITIALIZER can be used to initialize condition variables. The effect shall be equivalent to dynamic initialization by a call to *pthread\_cond\_init()* with parameter *attr* specified as NULL, except that no error checks are performed.

The behavior is undefined if the value specified by the *cond* argument to *pthread\_cond\_destroy()* does not refer to an initialized condition variable.

The behavior is undefined if the value specified by the *attr* argument to *pthread\_cond\_init()* does not refer to an initialized condition variable attributes object.

**RETURN VALUE**

If successful, the *pthread\_cond\_destroy()* and *pthread\_cond\_init()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.

**ERRORS**

The *pthread\_cond\_init()* function shall fail if:

[EAGAIN]      The system lacked the necessary resources (other than memory) to initialize another condition variable.

[ENOMEM]      Insufficient memory exists to initialize the condition variable.

These functions shall not return an error code of [EINTR].

## EXAMPLES

A condition variable can be destroyed immediately after all the threads that are blocked on it are awakened. For example, consider the following code:

```

struct list {
    pthread_mutex_t lm;
    ...
}

struct elt {
    key k;
    int busy;
    pthread_cond_t notbusy;
    ...
}

/* Find a list element and reserve it. */
struct elt *
list_find(struct list *lp, key k)
{
    struct elt *ep;

    pthread_mutex_lock(&lp->lm);
    while ((ep = find_elt(l, k) != NULL) && ep->busy)
        pthread_cond_wait(&ep->notbusy, &lp->lm);
    if (ep != NULL)
        ep->busy = 1;
    pthread_mutex_unlock(&lp->lm);
    return(ep);
}

delete_elt(struct list *lp, struct elt *ep)
{
    pthread_mutex_lock(&lp->lm);
    assert(ep->busy);
    ... remove ep from list ...
    ep->busy = 0; /* Paranoid. */
    (A) pthread_cond_broadcast(&ep->notbusy);
    pthread_mutex_unlock(&lp->lm);
    (B) pthread_cond_destroy(&ep->notbusy);
    free(ep);
}

```

In this example, the condition variable and its list element may be freed (line B) immediately after all threads waiting for it are awakened (line A), since the mutex and the code ensure that no other thread can touch the element to be deleted.

## APPLICATION USAGE

None.

## RATIONALE

If an implementation detects that the value specified by the *cond* argument to *pthread\_cond\_destroy()* does not refer to an initialized condition variable, it is recommended that the function should fail and report an [EINVAL] error.

If an implementation detects that the value specified by the *cond* argument to

52471 *pthread\_cond\_destroy()* or *pthread\_cond\_init()* refers to a condition variable that is in use (for  
 52472 example, in a *pthread\_cond\_wait()* call) by another thread, or detects that the value specified by  
 52473 the *cond* argument to *pthread\_cond\_init()* refers to an already initialized condition variable, it is  
 52474 recommended that the function should fail and report an [EBUSY] error.

52475 If an implementation detects that the value specified by the *attr* argument to *pthread\_cond\_init()*  
 52476 does not refer to an initialized condition variable attributes object, it is recommended that the  
 52477 function should fail and report an [EINVAL] error.

52478 See also *pthread\_mutex\_destroy()*.

#### 52479 FUTURE DIRECTIONS

52480 None.

#### 52481 SEE ALSO

52482 *pthread\_cond\_broadcast()*, *pthread\_cond\_timedwait()*, *pthread\_mutex\_destroy()*

52483 XBD <pthread.h>

#### 52484 CHANGE HISTORY

52485 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

#### 52486 Issue 6

52487 The *pthread\_cond\_destroy()* and *pthread\_cond\_init()* functions are marked as part of the Threads  
 52488 option.

52489 IEEE PASC Interpretation 1003.1c #34 is applied, updating the DESCRIPTION.

52490 The **restrict** keyword is added to the *pthread\_cond\_init()* prototype for alignment with the  
 52491 ISO/IEC 9899:1999 standard.

#### 52492 Issue 7

52493 The *pthread\_cond\_destroy()* and *pthread\_cond\_init()* functions are moved from the Threads option  
 52494 to the Base.

52495 The [EINVAL] error for an uninitialized condition variable and an uninitialized condition  
 52496 variable attributes object is removed; this condition results in undefined behavior.

52497 The [EBUSY] error for a condition variable already in use or an already initialized condition  
 52498 variable is removed; this condition results in undefined behavior.

52499 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0455 [70] is applied.

# pthread\_cond\_signal()

System Interfaces

## 52500 NAME

52501 pthread\_cond\_signal — signal a condition

## 52502 SYNOPSIS

52503 #include <pthread.h>

52504 int pthread\_cond\_signal(pthread\_cond\_t \*cond);

## 52505 DESCRIPTION

52506 Refer to *pthread\_cond\_broadcast()*.

**NAME**

pthread\_cond\_timedwait, pthread\_cond\_wait — wait on a condition

**SYNOPSIS**

```
#include <pthread.h>

int pthread_cond_timedwait(pthread_cond_t *restrict cond,
    pthread_mutex_t *restrict mutex,
    const struct timespec *restrict abstime);
int pthread_cond_wait(pthread_cond_t *restrict cond,
    pthread_mutex_t *restrict mutex);
```

**DESCRIPTION**

The *pthread\_cond\_timedwait()* and *pthread\_cond\_wait()* functions shall block on a condition variable. The application shall ensure that these functions are called with *mutex* locked by the calling thread; otherwise, an error (for PTHREAD\_MUTEX\_ERRORCHECK and robust mutexes) or undefined behavior (for other mutexes) results.

These functions atomically release *mutex* and cause the calling thread to block on the condition variable *cond*; atomically here means “atomically with respect to access by another thread to the mutex and then the condition variable”. That is, if another thread is able to acquire the mutex after the about-to-block thread has released it, then a subsequent call to *pthread\_cond\_broadcast()* or *pthread\_cond\_signal()* in that thread shall behave as if it were issued after the about-to-block thread has blocked.

Upon successful return, the mutex shall have been locked and shall be owned by the calling thread. If *mutex* is a robust mutex where an owner terminated while holding the lock and the state is recoverable, the mutex shall be acquired even though the function returns an error code.

When using condition variables there is always a Boolean predicate involving shared variables associated with each condition wait that is true if the thread should proceed. Spurious wakeups from the *pthread\_cond\_timedwait()* or *pthread\_cond\_wait()* functions may occur. Since the return from *pthread\_cond\_timedwait()* or *pthread\_cond\_wait()* does not imply anything about the value of this predicate, the predicate should be re-evaluated upon such return.

When a thread waits on a condition variable, having specified a particular mutex to either the *pthread\_cond\_timedwait()* or the *pthread\_cond\_wait()* operation, a dynamic binding is formed between that mutex and condition variable that remains in effect as long as at least one thread is blocked on the condition variable. During this time, the effect of an attempt by any thread to wait on that condition variable using a different mutex is undefined. Once all waiting threads have been unblocked (as by the *pthread\_cond\_broadcast()* operation), the next wait operation on that condition variable shall form a new dynamic binding with the mutex specified by that wait operation. Even though the dynamic binding between condition variable and mutex may be removed or replaced between the time a thread is unblocked from a wait on the condition variable and the time that it returns to the caller or begins cancellation cleanup, the unblocked thread shall always re-acquire the mutex specified in the condition wait operation call from which it is returning.

A condition wait (whether timed or not) is a cancellation point. When the cancelability type of a thread is set to PTHREAD\_CANCEL\_DEFERRED, a side-effect of acting upon a cancellation request while in a condition wait is that the mutex is (in effect) re-acquired before calling the first cancellation cleanup handler. The effect is as if the thread were unblocked, allowed to execute up to the point of returning from the call to *pthread\_cond\_timedwait()* or *pthread\_cond\_wait()*, but at that point notices the cancellation request and instead of returning to the caller of *pthread\_cond\_timedwait()* or *pthread\_cond\_wait()*, starts the thread cancellation activities, which includes calling cancellation cleanup handlers.

52555 A thread that has been unblocked because it has been canceled while blocked in a call to  
 52556 *pthread\_cond\_timedwait()* or *pthread\_cond\_wait()* shall not consume any condition signal that may  
 52557 be directed concurrently at the condition variable if there are other threads blocked on the  
 52558 condition variable.

52559 The *pthread\_cond\_timedwait()* function shall be equivalent to *pthread\_cond\_wait()*, except that an  
 52560 error is returned if the absolute time specified by *abstime* passes (that is, system time equals or  
 52561 exceeds *abstime*) before the condition *cond* is signaled or broadcasted, or if the absolute time  
 52562 specified by *abstime* has already been passed at the time of the call. When such timeouts occur,  
 52563 *pthread\_cond\_timedwait()* shall nonetheless release and re-acquire the mutex referenced by *mutex*,  
 52564 and may consume a condition signal directed concurrently at the condition variable.

52565 The condition variable shall have a clock attribute which specifies the clock that shall be used to  
 52566 measure the time specified by the *abstime* argument. The *pthread\_cond\_timedwait()* function is  
 52567 also a cancellation point.

52568 If a signal is delivered to a thread waiting for a condition variable, upon return from the signal  
 52569 handler the thread resumes waiting for the condition variable as if it was not interrupted, or it  
 52570 shall return zero due to spurious wakeup.

52571 The behavior is undefined if the value specified by the *cond* or *mutex* argument to these  
 52572 functions does not refer to an initialized condition variable or an initialized mutex object,  
 52573 respectively.

#### 52574 RETURN VALUE

52575 Except for [ETIMEDOUT], [ENOTRECOVERABLE], and [EOWNERDEAD], all these error +  
 52576 checks shall act as if they were performed immediately at the beginning of processing for the  
 52577 function and shall cause an error return, in effect, prior to modifying the state of the mutex  
 52578 specified by *mutex* or the condition variable specified by *cond*.

52579 Upon successful completion, a value of zero shall be returned; otherwise, an error number shall  
 52580 be returned to indicate the error.

#### 52581 ERRORS

52582 These functions shall fail if:

52583 [ENOTRECOVERABLE]

52584 The state protected by the mutex is not recoverable.

52585 [EOWNERDEAD]

52586 The mutex is a robust mutex and the process containing the previous owning  
 52587 thread terminated while holding the mutex lock. The mutex lock shall be  
 52588 acquired by the calling thread and it is up to the new owner to make the state  
 52589 consistent.

52590 [EPERM]

52591 The mutex type is PTHREAD\_MUTEX\_ERRORCHECK or the mutex is a  
 robust mutex, and the current thread does not own the mutex.

52592 The *pthread\_cond\_timedwait()* function shall fail if:

52593 [ETIMEDOUT] The time specified by *abstime* to *pthread\_cond\_timedwait()* has passed.

52594 [EINVAL] The *abstime* argument specified a nanosecond value less than zero or greater  
 52595 than or equal to 1000 million.

52596 These functions may fail if:

52597 [EOWNERDEAD]

52598 The mutex is a robust mutex and the previous owning thread terminated  
52599 while holding the mutex lock. The mutex lock shall be acquired by the calling  
52600 thread and it is up to the new owner to make the state consistent.

52601 These functions shall not return an error code of [EINTR].

## 52602 EXAMPLES

52603 None.

## 52604 APPLICATION USAGE

52605 Applications that have assumed that non-zero return values are errors will need updating for  
52606 use with robust mutexes, since a valid return for a thread acquiring a mutex which is protecting  
52607 a currently inconsistent state is [EOWNERDEAD]. Applications that do not check the error  
52608 returns, due to ruling out the possibility of such errors arising, should not use robust mutexes. If  
52609 an application is supposed to work with normal and robust mutexes, it should check all return  
52610 values for error conditions and if necessary take appropriate action.

## 52611 RATIONALE

52612 If an implementation detects that the value specified by the *cond* argument to  
52613 *pthread\_cond\_timedwait()* or *pthread\_cond\_wait()* does not refer to an initialized condition  
52614 variable, or detects that the value specified by the *mutex* argument to *pthread\_cond\_timedwait()* or  
52615 *pthread\_cond\_wait()* does not refer to an initialized mutex object, it is recommended that the  
52616 function should fail and report an [EINVAL] error.

## 52617 Condition Wait Semantics

52618 It is important to note that when *pthread\_cond\_wait()* and *pthread\_cond\_timedwait()* return  
52619 without error, the associated predicate may still be false. Similarly, when  
52620 *pthread\_cond\_timedwait()* returns with the timeout error, the associated predicate may be true  
52621 due to an unavoidable race between the expiration of the timeout and the predicate state change.

52622 The application needs to recheck the predicate on any return because it cannot be sure there is  
52623 another thread waiting on the thread to handle the signal, and if there is not then the signal is  
52624 lost. The burden is on the application to check the predicate.

52625 Some implementations, particularly on a multi-processor, may sometimes cause multiple  
52626 threads to wake up when the condition variable is signaled simultaneously on different  
52627 processors.

52628 In general, whenever a condition wait returns, the thread has to re-evaluate the predicate  
52629 associated with the condition wait to determine whether it can safely proceed, should wait  
52630 again, or should declare a timeout. A return from the wait does not imply that the associated  
52631 predicate is either true or false.

52632 It is thus recommended that a condition wait be enclosed in the equivalent of a “while loop”  
52633 that checks the predicate.



### Timed Wait Semantics

An absolute time measure was chosen for specifying the timeout parameter for two reasons. First, a relative time measure can be easily implemented on top of a function that specifies absolute time, but there is a race condition associated with specifying an absolute timeout on top of a function that specifies relative timeouts. For example, assume that `clock_gettime()` returns the current time and `cond_relative_timed_wait()` uses relative timeouts:

```
clock_gettime(CLOCK_REALTIME, &now)
reltime = sleep_til_this_absolute_time - now;
cond_relative_timed_wait(c, m, &reltime);
```

If the thread is preempted between the first statement and the last statement, the thread blocks for too long. Blocking, however, is irrelevant if an absolute timeout is used. An absolute timeout also need not be recomputed if it is used multiple times in a loop, such as that enclosing a condition wait.

For cases when the system clock is advanced discontinuously by an operator, it is expected that implementations process any timed wait expiring at an intervening time as if that time had actually occurred.

### Cancellation and Condition Wait

A condition wait, whether timed or not, is a cancellation point. That is, the functions `pthread_cond_wait()` or `pthread_cond_timedwait()` are points where a pending (or concurrent) cancellation request is noticed. The reason for this is that an indefinite wait is possible at these points—whatever event is being waited for, even if the program is totally correct, might never occur; for example, some input data being awaited might never be sent. By making condition wait a cancellation point, the thread can be canceled and perform its cancellation cleanup handler even though it may be stuck in some indefinite wait.

A side-effect of acting on a cancellation request while a thread is blocked on a condition variable is to re-acquire the mutex before calling any of the cancellation cleanup handlers. This is done in order to ensure that the cancellation cleanup handler is executed in the same state as the critical code that lies both before and after the call to the condition wait function. This rule is also required when interfacing to POSIX threads from languages, such as Ada or C++, which may choose to map cancellation onto a language exception; this rule ensures that each exception handler guarding a critical section can always safely depend upon the fact that the associated mutex has already been locked regardless of exactly where within the critical section the exception was raised. Without this rule, there would not be a uniform rule that exception handlers could follow regarding the lock, and so coding would become very cumbersome.

Therefore, since *some* statement has to be made regarding the state of the lock when a cancellation is delivered during a wait, a definition has been chosen that makes application coding most convenient and error free.

When acting on a cancellation request while a thread is blocked on a condition variable, the implementation is required to ensure that the thread does not consume any condition signals directed at that condition variable if there are any other threads waiting on that condition variable. This rule is specified in order to avoid deadlock conditions that could occur if these two independent requests (one acting on a thread and the other acting on the condition variable) were not processed independently.



**Performance of Mutexes and Condition Variables**

Mutexes are expected to be locked only for a few instructions. This practice is almost automatically enforced by the desire of programmers to avoid long serial regions of execution (which would reduce total effective parallelism).

When using mutexes and condition variables, one tries to ensure that the usual case is to lock the mutex, access shared data, and unlock the mutex. Waiting on a condition variable should be a relatively rare situation. For example, when implementing a read-write lock, code that acquires a read-lock typically needs only to increment the count of readers (under mutual-exclusion) and return. The calling thread would actually wait on the condition variable only when there is already an active writer. So the efficiency of a synchronization operation is bounded by the cost of mutex lock/unlock and not by condition wait. Note that in the usual case there is no context switch.

This is not to say that the efficiency of condition waiting is unimportant. Since there needs to be at least one context switch per Ada rendezvous, the efficiency of waiting on a condition variable is important. The cost of waiting on a condition variable should be little more than the minimal cost for a context switch plus the time to unlock and lock the mutex.

**Features of Mutexes and Condition Variables**

It had been suggested that the mutex acquisition and release be decoupled from condition wait. This was rejected because it is the combined nature of the operation that, in fact, facilitates realtime implementations. Those implementations can atomically move a high-priority thread between the condition variable and the mutex in a manner that is transparent to the caller. This can prevent extra context switches and provide more deterministic acquisition of a mutex when the waiting thread is signaled. Thus, fairness and priority issues can be dealt with directly by the scheduling discipline. Furthermore, the current condition wait operation matches existing practice.

**Scheduling Behavior of Mutexes and Condition Variables**

Synchronization primitives that attempt to interfere with scheduling policy by specifying an ordering rule are considered undesirable. Threads waiting on mutexes and condition variables are selected to proceed in an order dependent upon the scheduling policy rather than in some fixed order (for example, FIFO or priority). Thus, the scheduling policy determines which thread(s) are awakened and allowed to proceed.

**Timed Condition Wait**

The `pthread_cond_timedwait()` function allows an application to give up waiting for a particular condition after a given amount of time. An example of its use follows:

```
(void) pthread_mutex_lock(&t.mn);
    t.waiters++;
    clock_gettime(CLOCK_REALTIME, &ts);
    ts.tv_sec += 5;
    rc = 0;
    while (! mypredicate(&t) && rc == 0)
        rc = pthread_cond_timedwait(&t.cond, &t.mn, &ts);
    t.waiters--;
    if (rc == 0 || mypredicate(&t))
        setmystate(&t);
(void) pthread_mutex_unlock(&t.mn);
```

52722 By making the timeout parameter absolute, it does not need to be recomputed each time the  
52723 program checks its blocking predicate. If the timeout was relative, it would have to be  
52724 recomputed before each call. This would be especially difficult since such code would need to  
52725 take into account the possibility of extra wakeups that result from extra broadcasts or signals on  
52726 the condition variable that occur before either the predicate is true or the timeout is due.

#### 52727 **FUTURE DIRECTIONS**

52728 None.

#### 52729 **SEE ALSO**

52730 [\*pthread\\_cond\\_broadcast\(\)\*](#)

52731 XBD Section 4.12 (on page 111), [\*<pthread.h>\*](#)

#### 52732 **CHANGE HISTORY**

52733 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

#### 52734 **Issue 6**

52735 The *pthread\_cond\_timedwait()* and *pthread\_cond\_wait()* functions are marked as part of the  
52736 Threads option.

52737 The Open Group Corrigendum U021/9 is applied, correcting the prototype for the  
52738 *pthread\_cond\_wait()* function.

52739 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding semantics  
52740 for the Clock Selection option.

52741 The ERRORS section has an additional case for [EPERM] in response to IEEE PASC  
52742 Interpretation 1003.1c #28.

52743 The **restrict** keyword is added to the *pthread\_cond\_timedwait()* and *pthread\_cond\_wait()*  
52744 prototypes for alignment with the ISO/IEC 9899:1999 standard.

52745 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/89 is applied, updating the  
52746 DESCRIPTION for consistency with the *pthread\_cond\_destroy()* function that states it is safe to  
52747 destroy an initialized condition variable upon which no threads are currently blocked.

52748 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/90 is applied, updating words in the  
52749 DESCRIPTION from “the cancelability enable state” to “the cancelability type”.

52750 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/91 is applied, updating the ERRORS  
52751 section to remove the error case related to *abstime* from the *pthread\_cond\_wait()* function, and to  
52752 make the error case related to *abstime* mandatory for *pthread\_cond\_timedwait()* for consistency  
52753 with other functions.

52754 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/92 is applied, adding a new paragraph to  
52755 the RATIONALE section stating that an application should check the predicate on any return  
52756 from this function.

#### 52757 **Issue 7**

52758 SD5-XSH-ERN-44 is applied, changing the definition of the “shall fail” case of the [EINVAL]  
52759 error.

52760 Changes are made from The Open Group Technical Standard, 2006, Extended API Set Part 3.

52761 The *pthread\_cond\_timedwait()* and *pthread\_cond\_wait()* functions are moved from the Threads  
52762 option to the Base.

52763 The [EINVAL] error for an uninitialized condition variable or uninitialized mutex object is  
52764 removed; this condition results in undefined behavior"

52765 The [EPERM] error is revised and moved to the “shall fail” list of error conditions for the  
52766 *pthread\_cond\_timedwait()* function.

52767 The DESCRIPTION is updated to clarify the behavior when *mutex* is a robust mutex.

52768 The ERRORS section is updated to include “shall fail” cases for  
52769 PTHREAD\_MUTEX\_ERRORCHECK mutexes.

52770 The DESCRIPTION is rewritten to clarify that undefined behavior occurs only for mutexes  
52771 where the [EPERM] error is not mandated.

52772 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0456 [91,286,437] and  
52773 XSH/TC1-2008/0457 [239] are applied.

## 52774 NAME

52775 pthread\_condattr\_destroy, pthread\_condattr\_init — destroy and initialize the condition variable  
52776 attributes object

## 52777 SYNOPSIS

```
52778 #include <pthread.h>
52779
52779 int pthread_condattr_destroy(pthread_condattr_t *attr);
52780 int pthread_condattr_init(pthread_condattr_t *attr);
```

## 52781 DESCRIPTION

52782 The *pthread\_condattr\_destroy()* function shall destroy a condition variable attributes object; the  
52783 object becomes, in effect, uninitialized. An implementation may cause *pthread\_condattr\_destroy()*  
52784 to set the object referenced by *attr* to an invalid value. A destroyed *attr* attributes object can be  
52785 reinitialized using *pthread\_condattr\_init()*; the results of otherwise referencing the object after it  
52786 has been destroyed are undefined.

52787 The *pthread\_condattr\_init()* function shall initialize a condition variable attributes object *attr* with  
52788 the default value for all of the attributes defined by the implementation.

52789 Results are undefined if *pthread\_condattr\_init()* is called specifying an already initialized *attr*  
52790 attributes object.

52791 After a condition variable attributes object has been used to initialize one or more condition  
52792 variables, any function affecting the attributes object (including destruction) shall not affect any  
52793 previously initialized condition variables.

52794 This volume of POSIX.1-2008 requires two attributes, the *clock* attribute and the *process-shared*  
52795 attribute.

52796 Additional attributes, their default values, and the names of the associated functions to get and  
52797 set those attribute values are implementation-defined.

52798 The behavior is undefined if the value specified by the *attr* argument to  
52799 *pthread\_condattr\_destroy()* does not refer to an initialized condition variable attributes object.

## 52800 RETURN VALUE

52801 If successful, the *pthread\_condattr\_destroy()* and *pthread\_condattr\_init()* functions shall return  
52802 zero; otherwise, an error number shall be returned to indicate the error.

## 52803 ERRORS

52804 The *pthread\_condattr\_init()* function shall fail if:

52805 [ENOMEM] Insufficient memory exists to initialize the condition variable attributes object.

52806 These functions shall not return an error code of [EINTR].

## 52807 EXAMPLES

52808 None.

## 52809 APPLICATION USAGE

52810 None.

## 52811 RATIONALE

52812 A *process-shared* attribute has been defined for condition variables for the same reason it has been  
52813 defined for mutexes.

52814 If an implementation detects that the value specified by the *attr* argument to  
52815 *pthread\_condattr\_destroy()* does not refer to an initialized condition variable attributes object, it is  
52816 recommended that the function should fail and report an [EINVAL] error.

52817 See also *pthread\_attr\_destroy()* and *pthread\_mutex\_destroy()*.

52818 **FUTURE DIRECTIONS**

52819 None.

52820 **SEE ALSO**

52821 *pthread\_attr\_destroy()*, *pthread\_cond\_destroy()*, *pthread\_condattr\_getpshared()*, *pthread\_create()*,  
52822 *pthread\_mutex\_destroy()*

52823 XBD <pthread.h>

52824 **CHANGE HISTORY**

52825 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

52826 **Issue 6**

52827 The *pthread\_condattr\_destroy()* and *pthread\_condattr\_init()* functions are marked as part of the  
52828 Threads option.

52829 **Issue 7**

52830 The *pthread\_condattr\_destroy()* and *pthread\_condattr\_init()* functions are moved from the Threads  
52831 option to the Base.

52832 The [EINVAL] error for an uninitialized condition variable attributes object is removed; this  
52833 condition results in undefined behavior.

## 52834 NAME

52835 pthread\_condattr\_getclock, pthread\_condattr\_setclock — get and set the clock selection  
52836 condition variable attribute

## 52837 SYNOPSIS

```
52838 #include <pthread.h>
52839 int pthread_condattr_getclock(const pthread_condattr_t *restrict attr,
52840                             clockid_t *restrict clock_id);
52841 int pthread_condattr_setclock(pthread_condattr_t *attr,
52842                             clockid_t clock_id);
```

## 52843 DESCRIPTION

52844 The *pthread\_condattr\_getclock()* function shall obtain the value of the *clock* attribute from the  
52845 attributes object referenced by *attr*.

52846 The *pthread\_condattr\_setclock()* function shall set the *clock* attribute in an initialized attributes  
52847 object referenced by *attr*. If *pthread\_condattr\_setclock()* is called with a *clock\_id* argument that  
52848 refers to a CPU-time clock, the call shall fail.

52849 The *clock* attribute is the clock ID of the clock that shall be used to measure the timeout service of  
52850 *pthread\_cond\_timedwait()*. The default value of the *clock* attribute shall refer to the system clock.

52851 The behavior is undefined if the value specified by the *attr* argument to  
52852 *pthread\_condattr\_getclock()* or *pthread\_condattr\_setclock()* does not refer to an initialized condition  
52853 variable attributes object.

## 52854 RETURN VALUE

52855 If successful, the *pthread\_condattr\_getclock()* function shall return zero and store the value of the  
52856 clock attribute of *attr* into the object referenced by the *clock\_id* argument. Otherwise, an error  
52857 number shall be returned to indicate the error.

52858 If successful, the *pthread\_condattr\_setclock()* function shall return zero; otherwise, an error  
52859 number shall be returned to indicate the error.

## 52860 ERRORS

52861 The *pthread\_condattr\_setclock()* function may fail if:

52862 [EINVAL] The value specified by *clock\_id* does not refer to a known clock, or is a CPU-  
52863 time clock.

52864 These functions shall not return an error code of [EINTR].

## 52865 EXAMPLES

52866 None.

## 52867 APPLICATION USAGE

52868 None.

## 52869 RATIONALE

52870 If an implementation detects that the value specified by the *attr* argument to  
52871 *pthread\_condattr\_getclock()* or *pthread\_condattr\_setclock()* does not refer to an initialized condition  
52872 variable attributes object, it is recommended that the function should fail and report an  
52873 [EINVAL] error.

## 52874 FUTURE DIRECTIONS

52875 None.

**SEE ALSO**

*pthread\_cond\_destroy()*, *pthread\_cond\_timedwait()*, *pthread\_condattr\_destroy()*,  
*pthread\_condattr\_getpshared()*, *pthread\_create()*, *pthread\_mutex\_destroy()*

XBD <pthread.h>

**CHANGE HISTORY**

First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

**Issue 7**

The *pthread\_condattr\_getclock()* and *pthread\_condattr\_setclock()* functions are moved from the Clock Selection option to the Base.

The [EINVAL] error for an uninitialized condition variable attributes object is removed; this condition results in undefined behavior.

## NAME

pthread\_condattr\_getpshared, pthread\_condattr\_setpshared — get and set the process-shared condition variable attributes

## SYNOPSIS

```
TSH #include <pthread.h>

int pthread_condattr_getpshared(const pthread_condattr_t *restrict attr,
    int *restrict pshared);
int pthread_condattr_setpshared(pthread_condattr_t *attr,
    int pshared);
```

## DESCRIPTION

The *pthread\_condattr\_getpshared()* function shall obtain the value of the *process-shared* attribute from the attributes object referenced by *attr*.

The *pthread\_condattr\_setpshared()* function shall set the *process-shared* attribute in an initialized attributes object referenced by *attr*.

The *process-shared* attribute is set to `PTHREAD_PROCESS_SHARED` to permit a condition variable to be operated upon by any thread that has access to the memory where the condition variable is allocated, even if the condition variable is allocated in memory that is shared by multiple processes. See [Section 2.9.9](#) (on page 523) for further requirements. The default value of the attribute is `PTHREAD_PROCESS_PRIVATE`.

The behavior is undefined if the value specified by the *attr* argument to *pthread\_condattr\_getpshared()* or *pthread\_condattr\_setpshared()* does not refer to an initialized condition variable attributes object.

## RETURN VALUE

If successful, the *pthread\_condattr\_setpshared()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

If successful, the *pthread\_condattr\_getpshared()* function shall return zero and store the value of the *process-shared* attribute of *attr* into the object referenced by the *pshared* parameter. Otherwise, an error number shall be returned to indicate the error.

## ERRORS

The *pthread\_condattr\_setpshared()* function may fail if:

[EINVAL] The new value specified for the attribute is outside the range of legal values for that attribute.

These functions shall not return an error code of [EINTR].

## EXAMPLES

None.

## APPLICATION USAGE

None.

## RATIONALE

If an implementation detects that the value specified by the *attr* argument to *pthread\_condattr\_getpshared()* or *pthread\_condattr\_setpshared()* does not refer to an initialized condition variable attributes object, it is recommended that the function should fail and report an [EINVAL] error.



52929 **FUTURE DIRECTIONS**

52930 None.

52931 **SEE ALSO**52932 *pthread\_create()*, *pthread\_cond\_destroy()*, *pthread\_condattr\_destroy()*, *pthread\_mutex\_destroy()*

52933 XBD &lt;pthread.h&gt;

52934 **CHANGE HISTORY**

52935 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

52936 **Issue 6**52937 The *pthread\_condattr\_getpshared()* and *pthread\_condattr\_setpshared()* functions are marked as part  
52938 of the Threads and Thread Process-Shared Synchronization options.52939 The **restrict** keyword is added to the *pthread\_condattr\_getpshared()* prototype for alignment with  
52940 the ISO/IEC 9899:1999 standard.52941 **Issue 7**52942 The *pthread\_condattr\_getpshared()* and *pthread\_condattr\_setpshared()* functions are marked only as |  
52943 part of the Thread Process-Shared Synchronization option as the Threads option is now part of |  
52944 the Base.52945 The [EINVAL] error for an uninitialized condition variable attributes object is removed; this  
52946 condition results in undefined behavior.

52947 **NAME**

52948 pthread\_condattr\_init — initialize the condition variable attributes object

52949 **SYNOPSIS**

52950 #include &lt;pthread.h&gt;

52951 int pthread\_condattr\_init(pthread\_condattr\_t \*attr);

52952 **DESCRIPTION**52953 Refer to *pthread\_condattr\_destroy()*.

52954 **NAME**

52955 pthread\_condattr\_setclock — set the clock selection condition variable attribute

52956 **SYNOPSIS**

52957 #include &lt;pthread.h&gt;

52958 int pthread\_condattr\_setclock(pthread\_condattr\_t \*attr,

52959 clockid\_t clock\_id);

52960 **DESCRIPTION**52961 Refer to *pthread\_condattr\_getclock()*.

52962 **NAME**

52963 pthread\_condattr\_setpshared — set the process-shared condition variable attribute

52964 **SYNOPSIS**

```
52965 TSH      #include <pthread.h>
52966          int pthread_condattr_setpshared(pthread_condattr_t *attr,
52967          int pshared);
```

52968 **DESCRIPTION**52969 Refer to *pthread\_condattr\_getpshared()*.

52970 **NAME**

52971 pthread\_create — thread creation

52972 **SYNOPSIS**

```
52973 #include <pthread.h>
52974 int pthread_create(pthread_t *restrict thread,
52975     const pthread_attr_t *restrict attr,
52976     void *(*start_routine)(void*), void *restrict arg);
```

52977 **DESCRIPTION**

52978 The *pthread\_create()* function shall create a new thread, with attributes specified by *attr*, within a  
 52979 process. If *attr* is NULL, the default attributes shall be used. If the attributes specified by *attr* are  
 52980 modified later, the thread's attributes shall not be affected. Upon successful completion,  
 52981 *pthread\_create()* shall store the ID of the created thread in the location referenced by *thread*.

52982 The thread is created executing *start\_routine* with *arg* as its sole argument. If the *start\_routine*  
 52983 returns, the effect shall be as if there was an implicit call to *pthread\_exit()* using the return value  
 52984 of *start\_routine* as the exit status. Note that the thread in which *main()* was originally invoked  
 52985 differs from this. When it returns from *main()*, the effect shall be as if there was an implicit call to  
 52986 *exit()* using the return value of *main()* as the exit status.

52987 The signal state of the new thread shall be initialized as follows:

- 52988 • The signal mask shall be inherited from the creating thread.
- 52989 • The set of signals pending for the new thread shall be empty.

52990 XSI The thread-local current locale and the alternate stack shall not be inherited.

52991 The floating-point environment shall be inherited from the creating thread.

52992 If *pthread\_create()* fails, no new thread is created and the contents of the location referenced by  
 52993 *thread* are undefined.

52994 TCT If `_POSIX_THREAD_CPUTIME` is defined, the new thread shall have a CPU-time clock  
 52995 accessible, and the initial value of this clock shall be set to zero.

52996 The behavior is undefined if the value specified by the *attr* argument to *pthread\_create()* does not  
 52997 refer to an initialized thread attributes object.

52998 **RETURN VALUE**

52999 If successful, the *pthread\_create()* function shall return zero; otherwise, an error number shall be  
 53000 returned to indicate the error.

53001 **ERRORS**

53002 The *pthread\_create()* function shall fail if:

53003 [EAGAIN] The system lacked the necessary resources to create another thread, or the  
 53004 system-imposed limit on the total number of threads in a process  
 53005 {PTHREAD\_THREADS\_MAX} would be exceeded.

53006 [EPERM] The caller does not have appropriate privileges to set the required scheduling  
 53007 parameters or scheduling policy.

53008 The *pthread\_create()* function shall not return an error code of [EINTR].

## EXAMPLES

None.

## APPLICATION USAGE

There is no requirement on the implementation that the ID of the created thread be available before the newly created thread starts executing. The calling thread can obtain the ID of the created thread through the *thread* argument of the *pthread\_create()* function, and the newly created thread can obtain its ID by a call to *pthread\_self()*.

## RATIONALE

A suggested alternative to *pthread\_create()* would be to define two separate operations: create and start. Some applications would find such behavior more natural. Ada, in particular, separates the “creation” of a task from its “activation”.

Splitting the operation was rejected by the standard developers for many reasons:

- The number of calls required to start a thread would increase from one to two and thus place an additional burden on applications that do not require the additional synchronization. The second call, however, could be avoided by the additional complication of a start-up state attribute.
- An extra state would be introduced: “created but not started”. This would require the standard to specify the behavior of the thread operations when the target has not yet started executing.
- For those applications that require such behavior, it is possible to simulate the two separate steps with the facilities that are currently provided. The *start\_routine()* can synchronize by waiting on a condition variable that is signaled by the start operation.

An Ada implementor can choose to create the thread at either of two points in the Ada program: when the task object is created, or when the task is activated (generally at a “begin”). If the first approach is adopted, the *start\_routine()* needs to wait on a condition variable to receive the order to begin “activation”. The second approach requires no such condition variable or extra synchronization. In either approach, a separate Ada task control block would need to be created when the task object is created to hold rendezvous queues, and so on.

An extension of the preceding model would be to allow the state of the thread to be modified between the create and start. This would allow the thread attributes object to be eliminated. This has been rejected because:

- All state in the thread attributes object has to be able to be set for the thread. This would require the definition of functions to modify thread attributes. There would be no reduction in the number of function calls required to set up the thread. In fact, for an application that creates all threads using identical attributes, the number of function calls required to set up the threads would be dramatically increased. Use of a thread attributes object permits the application to make one set of attribute setting function calls. Otherwise, the set of attribute setting function calls needs to be made for each thread creation.
- Depending on the implementation architecture, functions to set thread state would require kernel calls, or for other implementation reasons would not be able to be implemented as macros, thereby increasing the cost of thread creation.
- The ability for applications to segregate threads by class would be lost.

Another suggested alternative uses a model similar to that for process creation, such as “thread fork”. The fork semantics would provide more flexibility and the “create” function can be implemented simply by doing a thread fork followed immediately by a call to the desired “start

routine” for the thread. This alternative has these problems:

- For many implementations, the entire stack of the calling thread would need to be duplicated, since in many architectures there is no way to determine the size of the calling frame.
- Efficiency is reduced since at least some part of the stack has to be copied, even though in most cases the thread never needs the copied context, since it merely calls the desired start routine.

If an implementation detects that the value specified by the *attr* argument to *pthread\_create()* does not refer to an initialized thread attributes object, it is recommended that the function should fail and report an [EINVAL] error.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*fork()*, *pthread\_exit()*, *pthread\_join()*

XBD Section 4.12 (on page 111), **<pthread.h>**

#### CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

#### Issue 6

The *pthread\_create()* function is marked as part of the Threads option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EPERM] mandatory error condition is added.

The thread CPU-time clock semantics are added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the *pthread\_create()* prototype for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION is updated to make it explicit that the floating-point environment is inherited from the creating thread.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/44 is applied, adding text that the alternate stack is not inherited.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/93 is applied, updating the ERRORS section to remove the mandatory [EINVAL] error (“The value specified by *attr* is invalid”), and adding the optional [EINVAL] error (“The attributes specified by *attr* are invalid”).

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/94 is applied, adding the APPLICATION USAGE section.

#### Issue 7

The *pthread\_create()* function is moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized thread attributes object is removed; this condition results in undefined behavior.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0458 [302] is applied.

**53094 NAME**

53095 pthread\_detach — detach a thread

**53096 SYNOPSIS**

```
53097 #include <pthread.h>
53098 int pthread_detach(pthread_t thread);
```

**53099 DESCRIPTION**

53100 The *pthread\_detach()* function shall indicate to the implementation that storage for the thread  
 53101 *thread* can be reclaimed when that thread terminates. If *thread* has not terminated,  
 53102 *pthread\_detach()* shall not cause it to terminate.

53103 The behavior is undefined if the value specified by the *thread* argument to *pthread\_detach()* does  
 53104 not refer to a joinable thread.

**53105 RETURN VALUE**

53106 If the call succeeds, *pthread\_detach()* shall return 0; otherwise, an error number shall be returned  
 53107 to indicate the error.

**53108 ERRORS**

53109 The *pthread\_detach()* function shall not return an error code of [EINTR].

**53110 EXAMPLES**

53111 None.

**53112 APPLICATION USAGE**

53113 None.

**53114 RATIONALE**

53115 The *pthread\_join()* or *pthread\_detach()* functions should eventually be called for every thread that  
 53116 is created so that storage associated with the thread may be reclaimed.

53117 It has been suggested that a “detach” function is not necessary; the *detachstate* thread creation  
 53118 attribute is sufficient, since a thread need never be dynamically detached. However, need arises  
 53119 in at least two cases:

- 53120 1. In a cancellation handler for a *pthread\_join()* it is nearly essential to have a  
 53121 *pthread\_detach()* function in order to detach the thread on which *pthread\_join()* was  
 53122 waiting. Without it, it would be necessary to have the handler do another *pthread\_join()* to  
 53123 attempt to detach the thread, which would both delay the cancellation processing for an  
 53124 unbounded period and introduce a new call to *pthread\_join()*, which might itself need a  
 53125 cancellation handler. A dynamic detach is nearly essential in this case.
- 53126 2. In order to detach the “initial thread” (as may be desirable in processes that set up server  
 53127 threads).

53128 If an implementation detects that the value specified by the *thread* argument to *pthread\_detach()*  
 53129 does not refer to a joinable thread, it is recommended that the function should fail and report an  
 53130 [EINVAL] error.

53131 If an implementation detects use of a thread ID after the end of its lifetime, it is recommended  
 53132 that the function should fail and report an [ESRCH] error.

**53133 FUTURE DIRECTIONS**

53134 None.



**53135 SEE ALSO**

53136 *pthread\_join()*

53137 XBD <pthread.h>

**53138 CHANGE HISTORY**

53139 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

**53140 Issue 6**

53141 The *pthread\_detach()* function is marked as part of the Threads option.

53142 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/95 is applied, updating the ERRORS  
53143 section so that the [EINVAL] and [ESRCH] error cases become optional.

**53144 Issue 7**

53145 The *pthread\_detach()* function is moved from the Threads option to the Base.

53146 Austin Group Interpretation 1003.1-2001 #142 is applied, removing the [ESRCH] error condition.

53147 The [EINVAL] error for a non-joinable thread is removed; this condition results in undefined  
53148 behavior.

53149 **NAME**

53150 pthread\_equal — compare thread IDs

53151 **SYNOPSIS**

53152 #include <pthread.h>

53153 int pthread\_equal(pthread\_t t1, pthread\_t t2);

53154 **DESCRIPTION**

53155 This function shall compare the thread IDs *t1* and *t2*.

53156 **RETURN VALUE**

53157 The *pthread\_equal()* function shall return a non-zero value if *t1* and *t2* are equal; otherwise, zero  
53158 shall be returned.

53159 If either *t1* or *t2* are not valid thread IDs, the behavior is undefined.

53160 **ERRORS**

53161 No errors are defined.

53162 The *pthread\_equal()* function shall not return an error code of [EINTR].

53163 **EXAMPLES**

53164 None.

53165 **APPLICATION USAGE**

53166 None.

53167 **RATIONALE**

53168 Implementations may choose to define a thread ID as a structure. This allows additional  
53169 flexibility and robustness over using an **int**. For example, a thread ID could include a sequence  
53170 number that allows detection of “dangling IDs” (copies of a thread ID that has been detached).  
53171 Since the C language does not support comparison on structure types, the *pthread\_equal()*  
53172 function is provided to compare thread IDs.

53173 **FUTURE DIRECTIONS**

53174 None.

53175 **SEE ALSO**

53176 *pthread\_create()*, *pthread\_self()*

53177 XBD <pthread.h>

53178 **CHANGE HISTORY**

53179 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

53180 **Issue 6**

53181 The *pthread\_equal()* function is marked as part of the Threads option.

53182 **Issue 7**

53183 The *pthread\_equal()* function is moved from the Threads option to the Base.

**53184 NAME**

53185 pthread\_exit — thread termination

**53186 SYNOPSIS**

53187 #include <pthread.h>

53188 void pthread\_exit(void \*value\_ptr);

**53189 DESCRIPTION**

53190 The *pthread\_exit()* function shall terminate the calling thread and make the value *value\_ptr*  
53191 available to any successful join with the terminating thread. Any cancellation cleanup handlers  
53192 that have been pushed and not yet popped shall be popped in the reverse order that they were  
53193 pushed and then executed. After all cancellation cleanup handlers have been executed, if the  
53194 thread has any thread-specific data, appropriate destructor functions shall be called in an  
53195 unspecified order. Thread termination does not release any application visible process resources,  
53196 including, but not limited to, mutexes and file descriptors, nor does it perform any process-level  
53197 cleanup actions, including, but not limited to, calling any *atexit()* routines that may exist.

53198 An implicit call to *pthread\_exit()* is made when a thread other than the thread in which *main()*  
53199 was first invoked returns from the start routine that was used to create it. The function's return  
53200 value shall serve as the thread's exit status.

53201 The behavior of *pthread\_exit()* is undefined if called from a cancellation cleanup handler or  
53202 destructor function that was invoked as a result of either an implicit or explicit call to  
53203 *pthread\_exit()*.

53204 After a thread has terminated, the result of access to local (auto) variables of the thread is  
53205 undefined. Thus, references to local variables of the exiting thread should not be used for the  
53206 *pthread\_exit()* *value\_ptr* parameter value.

53207 The process shall exit with an exit status of 0 after the last thread has been terminated. The  
53208 behavior shall be as if the implementation called *exit()* with a zero argument at thread  
53209 termination time.

**53210 RETURN VALUE**

53211 The *pthread\_exit()* function cannot return to its caller.

**53212 ERRORS**

53213 No errors are defined.

**53214 EXAMPLES**

53215 None.

**53216 APPLICATION USAGE**

53217 None.

**53218 RATIONALE**

53219 The normal mechanism by which a thread terminates is to return from the routine that was  
53220 specified in the *pthread\_create()* call that started it. The *pthread\_exit()* function provides the  
53221 capability for a thread to terminate without requiring a return from the start routine of that  
53222 thread, thereby providing a function analogous to *exit()*.

53223 Regardless of the method of thread termination, any cancellation cleanup handlers that have  
53224 been pushed and not yet popped are executed, and the destructors for any existing thread-  
53225 specific data are executed. This volume of POSIX.1-2008 requires that cancellation cleanup  
53226 handlers be popped and called in order. After all cancellation cleanup handlers have been  
53227 executed, thread-specific data destructors are called, in an unspecified order, for each item of  
53228 thread-specific data that exists in the thread. This ordering is necessary because cancellation  
53229 cleanup handlers may rely on thread-specific data.

53230 As the meaning of the status is determined by the application (except when the thread has been  
53231 canceled, in which case it is PTHREAD\_CANCELED), the implementation has no idea what an  
53232 illegal status value is, which is why no address error checking is done.

53233 **FUTURE DIRECTIONS**

53234 None.

53235 **SEE ALSO**

53236 *exit()*, *pthread\_create()*, *pthread\_join()*

53237 XBD <pthread.h>

53238 **CHANGE HISTORY**

53239 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

53240 **Issue 6**

53241 The *pthread\_exit()* function is marked as part of the Threads option.

53242 **Issue 7**

53243 The *pthread\_exit()* function is moved from the Threads option to the Base.

**53244 NAME**

53245 pthread\_getconcurrency, pthread\_setconcurrency — get and set the level of concurrency

**53246 SYNOPSIS**

```
53247 OB XSI #include <pthread.h>
53248 int pthread_getconcurrency(void);
53249 int pthread_setconcurrency(int new_level);
```

**53250 DESCRIPTION**

53251 Unbound threads in a process may or may not be required to be simultaneously active. By  
 53252 default, the threads implementation ensures that a sufficient number of threads are active so that  
 53253 the process can continue to make progress. While this conserves system resources, it may not  
 53254 produce the most effective level of concurrency.

53255 The *pthread\_setconcurrency()* function allows an application to inform the threads  
 53256 implementation of its desired concurrency level, *new\_level*. The actual level of concurrency  
 53257 provided by the implementation as a result of this function call is unspecified.

53258 If *new\_level* is zero, it causes the implementation to maintain the concurrency level at its  
 53259 discretion as if *pthread\_setconcurrency()* had never been called.

53260 The *pthread\_getconcurrency()* function shall return the value set by a previous call to the  
 53261 *pthread\_setconcurrency()* function. If the *pthread\_setconcurrency()* function was not previously  
 53262 called, this function shall return zero to indicate that the implementation is maintaining the  
 53263 concurrency level.

53264 A call to *pthread\_setconcurrency()* shall inform the implementation of its desired concurrency  
 53265 level. The implementation shall use this as a hint, not a requirement.

53266 If an implementation does not support multiplexing of user threads on top of several kernel-  
 53267 scheduled entities, the *pthread\_setconcurrency()* and *pthread\_getconcurrency()* functions are  
 53268 provided for source code compatibility but they shall have no effect when called. To maintain  
 53269 the function semantics, the *new\_level* parameter is saved when *pthread\_setconcurrency()* is called  
 53270 so that a subsequent call to *pthread\_getconcurrency()* shall return the same value.

**53271 RETURN VALUE**

53272 If successful, the *pthread\_setconcurrency()* function shall return zero; otherwise, an error number  
 53273 shall be returned to indicate the error.

53274 The *pthread\_getconcurrency()* function shall always return the concurrency level set by a previous  
 53275 call to *pthread\_setconcurrency()*. If the *pthread\_setconcurrency()* function has never been called,  
 53276 *pthread\_getconcurrency()* shall return zero.

**53277 ERRORS**

53278 The *pthread\_setconcurrency()* function shall fail if:

53279 [EINVAL] The value specified by *new\_level* is negative.

53280 [EAGAIN] The value specified by *new\_level* would cause a system resource to be  
 53281 exceeded.

53282 The *pthread\_setconcurrency()* function shall not return an error code of [EINTR].

**53283 EXAMPLES**

53284 None.

**53285 APPLICATION USAGE**

53286 Application developers should note that an implementation can always ignore any calls to  
53287 *pthread\_setconcurrency()* and return a constant for *pthread\_getconcurrency()*. For this reason, it is  
53288 not recommended that portable applications use this function.

**53289 RATIONALE**

53290 None.

**53291 FUTURE DIRECTIONS**

53292 These functions may be removed in a future version.

**53293 SEE ALSO**

53294 XBD [<pthread.h>](#)

**53295 CHANGE HISTORY**

53296 First released in Issue 5.

**53297 Issue 7**

53298 SD5-XSH-ERN-184 is applied.

53299 The *pthread\_getconcurrency()* and *pthread\_setconcurrency()* functions are marked obsolescent.

53300 **NAME**

53301 pthread\_getcpuclockid — access a thread CPU-time clock (ADVANCED REALTIME  
53302 THREADS)

53303 **SYNOPSIS**

```
53304 TCT #include <pthread.h>
53305      #include <time.h>
53306      int pthread_getcpuclockid(pthread_t thread_id, clockid_t *clock_id);
```

53307 **DESCRIPTION**

53308 The *pthread\_getcpuclockid()* function shall return in *clock\_id* the clock ID of the CPU-time clock of  
53309 the thread specified by *thread\_id*, if the thread specified by *thread\_id* exists.

53310 **RETURN VALUE**

53311 Upon successful completion, *pthread\_getcpuclockid()* shall return zero; otherwise, an error  
53312 number shall be returned to indicate the error.

53313 **ERRORS**

53314 No errors are defined.

53315 **EXAMPLES**

53316 None.

53317 **APPLICATION USAGE**

53318 The *pthread\_getcpuclockid()* function is part of the Thread CPU-Time Clocks option and need not  
53319 be provided on all implementations.

53320 **RATIONALE**

53321 If an implementation detects use of a thread ID after the end of its lifetime, it is recommended  
53322 that the function should fail and report an [ESRCH] error.

53323 **FUTURE DIRECTIONS**

53324 None.

53325 **SEE ALSO**

53326 *clock\_getcpuclockid()*, *clock\_getres()*, *timer\_create()*

53327 XBD *<pthread.h>*, *<time.h>*

53328 **CHANGE HISTORY**

53329 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

53330 In the SYNOPSIS, the inclusion of *<sys/types.h>* is no longer required.

53331 **Issue 7**

53332 The *pthread\_getcpuclockid()* function is marked only as part of the Thread CPU-Time Clocks  
53333 option as the Threads option is now part of the Base.

53334 Austin Group Interpretation 1003.1-2001 #142 is applied, removing the [ESRCH] error condition.

**NAME**

pthread\_getschedparam, pthread\_setschedparam — dynamic thread scheduling parameters access (**REALTIME THREADS**)

**SYNOPSIS**

```

#include <pthread.h>

int pthread_getschedparam(pthread_t thread, int *restrict policy,
    struct sched_param *restrict param);
int pthread_setschedparam(pthread_t thread, int policy,
    const struct sched_param *param);

```

**DESCRIPTION**

The *pthread\_getschedparam()* and *pthread\_setschedparam()* functions shall, respectively, get and set the scheduling policy and parameters of individual threads within a multi-threaded process to be retrieved and set. For SCHED\_FIFO and SCHED\_RR, the only required member of the **sched\_param** structure is the priority *sched\_priority*. For SCHED\_OTHER, the affected scheduling parameters are implementation-defined.

The *pthread\_getschedparam()* function shall retrieve the scheduling policy and scheduling parameters for the thread whose thread ID is given by *thread* and shall store those values in *policy* and *param*, respectively. The priority value returned from *pthread\_getschedparam()* shall be the value specified by the most recent *pthread\_setschedparam()*, *pthread\_setschedprio()*, or *pthread\_create()* call affecting the target thread. It shall not reflect any temporary adjustments to its priority as a result of any priority inheritance or ceiling functions. The *pthread\_setschedparam()* function shall set the scheduling policy and associated scheduling parameters for the thread whose thread ID is given by *thread* to the policy and associated parameters provided in *policy* and *param*, respectively.

The *policy* parameter may have the value SCHED\_OTHER, SCHED\_FIFO, or SCHED\_RR. The scheduling parameters for the SCHED\_OTHER policy are implementation-defined. The SCHED\_FIFO and SCHED\_RR policies shall have a single scheduling parameter, *priority*.

If **\_POSIX\_THREAD\_SPORADIC\_SERVER** is defined, then the *policy* argument may have the value SCHED\_SPORADIC, with the exception for the *pthread\_setschedparam()* function that if the scheduling policy was not SCHED\_SPORADIC at the time of the call, it is implementation-defined whether the function is supported; in other words, the implementation need not allow the application to dynamically change the scheduling policy to SCHED\_SPORADIC. The sporadic server scheduling policy has the associated parameters *sched\_ss\_low\_priority*, *sched\_ss\_repl\_period*, *sched\_ss\_init\_budget*, *sched\_priority*, and *sched\_ss\_max\_repl*. The specified *sched\_ss\_repl\_period* shall be greater than or equal to the specified *sched\_ss\_init\_budget* for the function to succeed; if it is not, then the function shall fail. The value of *sched\_ss\_max\_repl* shall be within the inclusive range [1, {SS\_REPL\_MAX}] for the function to succeed; if not, the function shall fail. It is unspecified whether the *sched\_ss\_repl\_period* and *sched\_ss\_init\_budget* values are stored as provided by this function or are rounded to align with the resolution of the clock being used.

If the *pthread\_setschedparam()* function fails, the scheduling parameters shall not be changed for the target thread.

**RETURN VALUE**

If successful, the *pthread\_getschedparam()* and *pthread\_setschedparam()* functions shall return zero; otherwise, an error number shall be returned to indicate the error.



**ERRORS**

The *pthread\_getschedparam()* function shall fail if:

[ENOTSUP] An attempt was made to set the policy or scheduling parameters to an unsupported value.

[ENOTSUP] An attempt was made to dynamically change the scheduling policy to SCHED\_SPORADIC, and the implementation does not support this change.

The *pthread\_getschedparam()* function may fail if:

[EINVAL] The value specified by *policy* or one of the scheduling parameters associated with the scheduling policy *policy* is invalid.

[EPERM] The caller does not have appropriate privileges to set either the scheduling parameters or the scheduling policy of the specified thread.

[EPERM] The implementation does not allow the application to modify one of the parameters to the value specified.

These functions shall not return an error code of [EINTR].

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

If an implementation detects use of a thread ID after the end of its lifetime, it is recommended that the function should fail and report an [ESRCH] error.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*pthread\_setschedprio()*, *sched\_getparam()*, *sched\_getscheduler()*

XBD <pthread.h>, <sched.h>

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

**Issue 6**

The *pthread\_getschedparam()* and *pthread\_setschedparam()* functions are marked as part of the Threads and Thread Execution Scheduling options.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Execution Scheduling option.

The Open Group Corrigendum U026/2 is applied, correcting the prototype for the *pthread\_setschedparam()* function so that its second argument is of type **int**.

The SCHED\_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

The **restrict** keyword is added to the *pthread\_getschedparam()* prototype for alignment with the ISO/IEC 9899:1999 standard.

The Open Group Corrigendum U047/1 is applied.

IEEE PASC Interpretation 1003.1 #96 is applied, noting that priority values can also be set by a call to the *pthread\_setschedprio()* function.

53421 **Issue 7**

53422 The *pthread\_getschedparam()* and *pthread\_setschedparam()* functions are marked only as part of the  
53423 Thread Execution Scheduling option as the Threads option is now part of the Base.

53424 Austin Group Interpretation 1003.1-2001 #119 is applied, clarifying the accuracy requirements  
53425 for the *sched\_ss\_repl\_period* and *sched\_ss\_init\_budget* values.

53426 Austin Group Interpretation 1003.1-2001 #142 is applied, removing the [ESRCH] error condition.  
53427 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0459 [314] is applied.

53428 **NAME**

53429 pthread\_getspecific, pthread\_setspecific — thread-specific data management

53430 **SYNOPSIS**

53431 #include &lt;pthread.h&gt;

53432 void \*pthread\_getspecific(pthread\_key\_t key);

53433 int pthread\_setspecific(pthread\_key\_t key, const void \*value);

53434 **DESCRIPTION**53435 The *pthread\_getspecific()* function shall return the value currently bound to the specified *key* on  
53436 behalf of the calling thread.53437 The *pthread\_setspecific()* function shall associate a thread-specific *value* with a *key* obtained via a  
53438 previous call to *pthread\_key\_create()*. Different threads may bind different values to the same  
53439 key. These values are typically pointers to blocks of dynamically allocated memory that have  
53440 been reserved for use by the calling thread.53441 The effect of calling *pthread\_getspecific()* or *pthread\_setspecific()* with a *key* value not obtained  
53442 from *pthread\_key\_create()* or after *key* has been deleted with *pthread\_key\_delete()* is undefined.53443 Both *pthread\_getspecific()* and *pthread\_setspecific()* may be called from a thread-specific data  
53444 destructor function. A call to *pthread\_getspecific()* for the thread-specific data *key* being  
53445 destroyed shall return the value NULL, unless the value is changed (after the destructor starts)  
53446 by a call to *pthread\_setspecific()*. Calling *pthread\_setspecific()* from a thread-specific data  
53447 destructor routine may result either in lost storage (after at least  
53448 PTHREAD\_DESTRUCTOR\_ITERATIONS attempts at destruction) or in an infinite loop.

53449 Both functions may be implemented as macros.

53450 **RETURN VALUE**53451 The *pthread\_getspecific()* function shall return the thread-specific data value associated with the  
53452 given *key*. If no thread-specific data value is associated with *key*, then the value NULL shall be  
53453 returned.53454 If successful, the *pthread\_setspecific()* function shall return zero; otherwise, an error number shall  
53455 be returned to indicate the error.53456 **ERRORS**53457 No errors are returned from *pthread\_getspecific()*.53458 The *pthread\_setspecific()* function shall fail if:

53459 [ENOMEM] Insufficient memory exists to associate the non-NULL value with the key.

53460 The *pthread\_setspecific()* function shall not return an error code of [EINTR].53461 **EXAMPLES**

53462 None.

53463 **APPLICATION USAGE**

53464 None.

53465 **RATIONALE**53466 Performance and ease-of-use of *pthread\_getspecific()* are critical for functions that rely on  
53467 maintaining state in thread-specific data. Since no errors are required to be detected by it, and  
53468 since the only error that could be detected is the use of an invalid key, the function to  
53469 *pthread\_getspecific()* has been designed to favor speed and simplicity over error reporting.53470 If an implementation detects that the value specified by the *key* argument to *pthread\_setspecific()*  
53471 does not refer to a key value obtained from *pthread\_key\_create()* or refers to a key that has been

53472 deleted with *pthread\_key\_delete()*, it is recommended that the function should fail and report an  
53473 [EINVAL] error.

#### 53474 FUTURE DIRECTIONS

53475 None.

#### 53476 SEE ALSO

53477 *pthread\_key\_create()*

53478 XBD <pthread.h>

#### 53479 CHANGE HISTORY

53480 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

##### 53481 Issue 6

53482 The *pthread\_getspecific()* and *pthread\_setspecific()* functions are marked as part of the Threads  
53483 option.

53484 IEEE PASC Interpretation 1003.1c #3 (Part 6) is applied, updating the DESCRIPTION.

53485 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/96 is applied, updating the ERRORS  
53486 section so that the [ENOMEM] error case is changed from “to associate the value with the key”  
53487 to “to associate the non-NULL value with the key”.

##### 53488 Issue 7

53489 Austin Group Interpretation 1003.1-2001 #063 is applied, updating the ERRORS section.

53490 The *pthread\_getspecific()* and *pthread\_setspecific()* functions are moved from the Threads option to  
53491 the Base.

53492 The [EINVAL] error for a key value not obtained from *pthread\_key\_create()* or a key deleted with  
53493 *pthread\_key\_delete()* is removed; this condition results in undefined behavior.

53494 **NAME**

53495 pthread\_join — wait for thread termination

53496 **SYNOPSIS**

53497 #include &lt;pthread.h&gt;

53498 int pthread\_join(pthread\_t thread, void \*\*value\_ptr);

53499 **DESCRIPTION**

53500 The *pthread\_join()* function shall suspend execution of the calling thread until the target *thread*  
 53501 terminates, unless the target *thread* has already terminated. On return from a successful  
 53502 *pthread\_join()* call with a non-NULL *value\_ptr* argument, the value passed to *pthread\_exit()* by  
 53503 the terminating thread shall be made available in the location referenced by *value\_ptr*. When a  
 53504 *pthread\_join()* returns successfully, the target thread has been terminated. The results of multiple  
 53505 simultaneous calls to *pthread\_join()* specifying the same target thread are undefined. If the  
 53506 thread calling *pthread\_join()* is canceled, then the target thread shall not be detached.

53507 It is unspecified whether a thread that has exited but remains unjoined counts against  
 53508 {PTHREAD\_THREADS\_MAX}.

53509 The behavior is undefined if the value specified by the *thread* argument to *pthread\_join()* does not  
 53510 refer to a joinable thread.

53511 The behavior is undefined if the value specified by the *thread* argument to *pthread\_join()* refers to  
 53512 the calling thread.

53513 **RETURN VALUE**

53514 If successful, the *pthread\_join()* function shall return zero; otherwise, an error number shall be  
 53515 returned to indicate the error.

53516 **ERRORS**53517 The *pthread\_join()* function may fail if:

53518 [EDEADLK] A deadlock was detected.

53519 The *pthread\_join()* function shall not return an error code of [EINTR].53520 **EXAMPLES**

53521 An example of thread creation and deletion follows:

```

53522 typedef struct {
53523     int *ar;
53524     long n;
53525 } subarray;

53526 void *
53527 incer(void *arg)
53528 {
53529     long i;

53530     for (i = 0; i < ((subarray *)arg)->n; i++)
53531         ((subarray *)arg)->ar[i]++;
53532 }

53533 int main(void)
53534 {
53535     int          ar[1000000];
53536     pthread_t    th1, th2;
53537     subarray     sb1, sb2;

```

```

53538         sb1.ar = &ar[0];
53539         sb1.n  = 500000;
53540         (void) pthread_create(&th1, NULL, incer, &sb1);

53541         sb2.ar = &ar[500000];
53542         sb2.n  = 500000;
53543         (void) pthread_create(&th2, NULL, incer, &sb2);

53544         (void) pthread_join(th1, NULL);
53545         (void) pthread_join(th2, NULL);
53546         return 0;
53547     }

```

## APPLICATION USAGE

None.

## RATIONALE

The *pthread\_join()* function is a convenience that has proven useful in multi-threaded applications. It is true that a programmer could simulate this function if it were not provided by passing extra state as part of the argument to the *start\_routine()*. The terminating thread would set a flag to indicate termination and broadcast a condition that is part of that state; a joining thread would wait on that condition variable. While such a technique would allow a thread to wait on more complex conditions (for example, waiting for multiple threads to terminate), waiting on individual thread termination is considered widely useful. Also, including the *pthread\_join()* function in no way precludes a programmer from coding such complex waits. Thus, while not a primitive, including *pthread\_join()* in this volume of POSIX.1-2008 was considered valuable.

The *pthread\_join()* function provides a simple mechanism allowing an application to wait for a thread to terminate. After the thread terminates, the application may then choose to clean up resources that were used by the thread. For instance, after *pthread\_join()* returns, any application-provided stack storage could be reclaimed.

The *pthread\_join()* or *pthread\_detach()* function should eventually be called for every thread that is created with the *detachstate* attribute set to *PTHREAD\_CREATE\_JOINABLE* so that storage associated with the thread may be reclaimed.

The interaction between *pthread\_join()* and cancellation is well-defined for the following reasons:

- The *pthread\_join()* function, like all other non-async-cancel-safe functions, can only be called with deferred cancelability type.
- Cancellation cannot occur in the disabled cancelability state.

Thus, only the default cancelability state need be considered. As specified, either the *pthread\_join()* call is canceled, or it succeeds, but not both. The difference is obvious to the application, since either a cancellation handler is run or *pthread\_join()* returns. There are no race conditions since *pthread\_join()* was called in the deferred cancelability state.

If an implementation detects that the value specified by the *thread* argument to *pthread\_join()* does not refer to a joinable thread, it is recommended that the function should fail and report an [EINVAL] error.

If an implementation detects that the value specified by the *thread* argument to *pthread\_join()* refers to the calling thread, it is recommended that the function should fail and report an [EDEADLK] error.

If an implementation detects use of a thread ID after the end of its lifetime, it is recommended

53583 that the function should fail and report an [ESRCH] error.

53584 **FUTURE DIRECTIONS**

53585 None.

53586 **SEE ALSO**

53587 *pthread\_create()*, *wait()*

53588 XBD Section 4.12 (on page 111), *<pthread.h>*

53589 **CHANGE HISTORY**

53590 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

53591 **Issue 6**

53592 The *pthread\_join()* function is marked as part of the Threads option.

53593 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/97 is applied, updating the ERRORS  
53594 section so that the [EINVAL] error is made optional and the words “the implementation has  
53595 detected” are removed from it.

53596 **Issue 7**

53597 The *pthread\_join()* function is moved from the Threads option to the Base.

53598 Austin Group Interpretation 1003.1-2001 #142 is applied, removing the [ESRCH] error condition.

53599 The [EINVAL] error for a non-joinable thread is removed; this condition results in undefined  
53600 behavior.

53601 The [EDEADLK] error for the calling thread is removed; this condition results in undefined  
53602 behavior.

53603 **NAME**

53604 pthread\_key\_create — thread-specific data key creation

53605 **SYNOPSIS**

```
53606 #include <pthread.h>
53607 int pthread_key_create(pthread_key_t *key, void (*destructor)(void*));
```

53608 **DESCRIPTION**

53609 The *pthread\_key\_create()* function shall create a thread-specific data key visible to all threads in  
 53610 the process. Key values provided by *pthread\_key\_create()* are opaque objects used to locate  
 53611 thread-specific data. Although the same key value may be used by different threads, the values  
 53612 bound to the key by *pthread\_setspecific()* are maintained on a per-thread basis and persist for the  
 53613 life of the calling thread.

53614 Upon key creation, the value NULL shall be associated with the new key in all active threads.  
 53615 Upon thread creation, the value NULL shall be associated with all defined keys in the new  
 53616 thread.

53617 An optional destructor function may be associated with each key value. At thread exit, if a key  
 53618 value has a non-NULL destructor pointer, and the thread has a non-NULL value associated with  
 53619 that key, the value of the key is set to NULL, and then the function pointed to is called with the  
 53620 previously associated value as its sole argument. The order of destructor calls is unspecified if  
 53621 more than one destructor exists for a thread when it exits.

53622 If, after all the destructors have been called for all non-NULL values with associated destructors,  
 53623 there are still some non-NULL values with associated destructors, then the process is repeated.  
 53624 If, after at least {PTHREAD\_DESTRUCTOR\_ITERATIONS} iterations of destructor calls for  
 53625 outstanding non-NULL values, there are still some non-NULL values with associated  
 53626 destructors, implementations may stop calling destructors, or they may continue calling  
 53627 destructors until no non-NULL values with associated destructors exist, even though this might  
 53628 result in an infinite loop.

53629 **RETURN VALUE**

53630 If successful, the *pthread\_key\_create()* function shall store the newly created key value at *\*key* and  
 53631 shall return zero. Otherwise, an error number shall be returned to indicate the error.

53632 **ERRORS**

53633 The *pthread\_key\_create()* function shall fail if:

53634 [EAGAIN] The system lacked the necessary resources to create another thread-specific  
 53635 data key, or the system-imposed limit on the total number of keys per process  
 53636 {PTHREAD\_KEYS\_MAX} has been exceeded.

53637 [ENOMEM] Insufficient memory exists to create the key.

53638 The *pthread\_key\_create()* function shall not return an error code of [EINTR].

53639 **EXAMPLES**

53640 The following example demonstrates a function that initializes a thread-specific data key when it  
 53641 is first called, and associates a thread-specific object with each calling thread, initializing this  
 53642 object when necessary.

```
53643 static pthread_key_t key;
53644 static pthread_once_t key_once = PTHREAD_ONCE_INIT;

53645 static void
53646 make_key( )
53647 {
```



```

53648     (void) pthread_key_create(&key, NULL);
53649 }
53650 func()
53651 {
53652     void *ptr;
53653     (void) pthread_once(&key_once, make_key);
53654     if ((ptr = pthread_getspecific(key)) == NULL) {
53655         ptr = malloc(OBJECT_SIZE);
53656         ...
53657         (void) pthread_setspecific(key, ptr);
53658     }
53659     ...
53660 }

```

Note that the key has to be initialized before *pthread\_getspecific()* or *pthread\_setspecific()* can be used. The *pthread\_key\_create()* call could either be explicitly made in a module initialization routine, or it can be done implicitly by the first call to a module as in this example. Any attempt to use the key before it is initialized is a programming error, making the code below incorrect.

```

53665 static pthread_key_t key;
53666 func()
53667 {
53668     void *ptr;
53669     /* KEY NOT INITIALIZED!!! THIS WON'T WORK!!! */
53670     if ((ptr = pthread_getspecific(key)) == NULL &&
53671         pthread_setspecific(key, NULL) != 0) {
53672         pthread_key_create(&key, NULL);
53673         ...
53674     }
53675 }

```

## APPLICATION USAGE

None.

## RATIONALE

### Destructor Functions

Normally, the value bound to a key on behalf of a particular thread is a pointer to storage allocated dynamically on behalf of the calling thread. The destructor functions specified with *pthread\_key\_create()* are intended to be used to free this storage when the thread exits. Thread cancellation cleanup handlers cannot be used for this purpose because thread-specific data may persist outside the lexical scope in which the cancellation cleanup handlers operate.

If the value associated with a key needs to be updated during the lifetime of the thread, it may be necessary to release the storage associated with the old value before the new value is bound. Although the *pthread\_setspecific()* function could do this automatically, this feature is not needed often enough to justify the added complexity. Instead, the programmer is responsible for freeing the stale storage:

```

53690 pthread_getspecific(key, &old);
53691 new = allocate();
53692 destructor(old);

```

53693 pthread\_setspecific(key, new);

53694 **Note:** The above example could leak storage if run with asynchronous cancellation enabled. No such  
 53695 problems occur in the default cancellation state if no cancellation points occur between the get  
 53696 and set.

53697 There is no notion of a destructor-safe function. If an application does not call *pthread\_exit()*  
 53698 from a signal handler, or if it blocks any signal whose handler may call *pthread\_exit()* while  
 53699 calling async-unsafe functions, all functions may be safely called from destructors.

#### 53700 Non-Idempotent Data Key Creation

53701 There were requests to make *pthread\_key\_create()* idempotent with respect to a given *key* address  
 53702 parameter. This would allow applications to call *pthread\_key\_create()* multiple times for a given  
 53703 *key* address and be guaranteed that only one key would be created. Doing so would require the  
 53704 key value to be previously initialized (possibly at compile time) to a known null value and  
 53705 would require that implicit mutual-exclusion be performed based on the address and contents of  
 53706 the *key* parameter in order to guarantee that exactly one key would be created.

53707 Unfortunately, the implicit mutual-exclusion would not be limited to only *pthread\_key\_create()*.  
 53708 On many implementations, implicit mutual-exclusion would also have to be performed by  
 53709 *pthread\_getspecific()* and *pthread\_setspecific()* in order to guard against using incompletely stored  
 53710 or not-yet-visible key values. This could significantly increase the cost of important operations,  
 53711 particularly *pthread\_getspecific()*.

53712 Thus, this proposal was rejected. The *pthread\_key\_create()* function performs no implicit  
 53713 synchronization. It is the responsibility of the programmer to ensure that it is called exactly once  
 53714 per key before use of the key. Several straightforward mechanisms can already be used to  
 53715 accomplish this, including calling explicit module initialization functions, using mutexes, and  
 53716 using *pthread\_once()*. This places no significant burden on the programmer, introduces no  
 53717 possibly confusing *ad hoc* implicit synchronization mechanism, and potentially allows  
 53718 commonly used thread-specific data operations to be more efficient.

#### 53719 FUTURE DIRECTIONS

53720 None.

#### 53721 SEE ALSO

53722 *pthread\_getspecific()*, *pthread\_key\_delete()*

53723 XBD <pthread.h>

#### 53724 CHANGE HISTORY

53725 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

#### 53726 Issue 6

53727 The *pthread\_key\_create()* function is marked as part of the Threads option.

53728 IEEE PASC Interpretation 1003.1c #8 is applied, updating the DESCRIPTION.

#### 53729 Issue 7

53730 The *pthread\_key\_create()* function is moved from the Threads option to the Base.

**NAME**

pthread\_key\_delete — thread-specific data key deletion

**SYNOPSIS**

```
#include <pthread.h>
```

```
int pthread_key_delete(pthread_key_t key);
```

**DESCRIPTION**

The *pthread\_key\_delete()* function shall delete a thread-specific data key previously returned by *pthread\_key\_create()*. The thread-specific data values associated with *key* need not be NULL at the time *pthread\_key\_delete()* is called. It is the responsibility of the application to free any application storage or perform any cleanup actions for data structures related to the deleted key or associated thread-specific data in any threads; this cleanup can be done either before or after *pthread\_key\_delete()* is called. Any attempt to use *key* following the call to *pthread\_key\_delete()* results in undefined behavior.

The *pthread\_key\_delete()* function shall be callable from within destructor functions. No destructor functions shall be invoked by *pthread\_key\_delete()*. Any destructor function that may have been associated with *key* shall no longer be called upon thread exit.

**RETURN VALUE**

If successful, the *pthread\_key\_delete()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

**ERRORS**

The *pthread\_key\_delete()* function shall not return an error code of [EINTR].

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

A thread-specific data key deletion function has been included in order to allow the resources associated with an unused thread-specific data key to be freed. Unused thread-specific data keys can arise, among other scenarios, when a dynamically loaded module that allocated a key is unloaded.

Conforming applications are responsible for performing any cleanup actions needed for data structures associated with the key to be deleted, including data referenced by thread-specific data values. No such cleanup is done by *pthread\_key\_delete()*. In particular, destructor functions are not called. There are several reasons for this division of responsibility:

1. The associated destructor functions used to free thread-specific data at thread exit time are only guaranteed to work correctly when called in the thread that allocated the thread-specific data. (Destructors themselves may utilize thread-specific data.) Thus, they cannot be used to free thread-specific data in other threads at key deletion time. Attempting to have them called by other threads at key deletion time would require other threads to be asynchronously interrupted. But since interrupted threads could be in an arbitrary state, including holding locks necessary for the destructor to run, this approach would fail. In general, there is no safe mechanism whereby an implementation could free thread-specific data at key deletion time.
2. Even if there were a means of safely freeing thread-specific data associated with keys to be deleted, doing so would require that implementations be able to enumerate the threads with non-NULL data and potentially keep them from creating more thread-

specific data while the key deletion is occurring. This special case could cause extra synchronization in the normal case, which would otherwise be unnecessary.

For an application to know that it is safe to delete a key, it has to know that all the threads that might potentially ever use the key do not attempt to use it again. For example, it could know this if all the client threads have called a cleanup procedure declaring that they are through with the module that is being shut down, perhaps by setting a reference count to zero.

If an implementation detects that the value specified by the *key* argument to *pthread\_key\_delete()* does not refer to a key value obtained from *pthread\_key\_create()* or refers to a key that has been deleted with *pthread\_key\_delete()*, it is recommended that the function should fail and report an [EINVAL] error.

#### 53787 FUTURE DIRECTIONS

53788 None.

#### 53789 SEE ALSO

53790 *pthread\_key\_create()*

53791 XBD <pthread.h>

#### 53792 CHANGE HISTORY

53793 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

#### 53794 Issue 6

53795 The *pthread\_key\_delete()* function is marked as part of the Threads option.

#### 53796 Issue 7

53797 The *pthread\_key\_delete()* function is moved from the Threads option to the Base.

53798 The [EINVAL] error for a key value not obtained from *pthread\_key\_create()* or a key deleted with  
53799 *pthread\_key\_delete()* is removed; this condition results in undefined behavior.

53800 **NAME**

53801 pthread\_kill — send a signal to a thread

53802 **SYNOPSIS**

```
53803 CX      #include <signal.h>
53804      int pthread_kill(pthread_t thread, int sig);
```

53805 **DESCRIPTION**53806 The *pthread\_kill()* function shall request that a signal be delivered to the specified thread.53807 As in *kill()*, if *sig* is zero, error checking shall be performed but no signal shall actually be sent.53808 **RETURN VALUE**

53809 Upon successful completion, the function shall return a value of zero. Otherwise, the function  
 53810 shall return an error number. If the *pthread\_kill()* function fails, no signal shall be sent.

53811 **ERRORS**53812 The *pthread\_kill()* function shall fail if:53813 [EINVAL] The value of the *sig* argument is an invalid or unsupported signal number.53814 The *pthread\_kill()* function shall not return an error code of [EINTR].53815 **EXAMPLES**

53816 None.

53817 **APPLICATION USAGE**

53818 The *pthread\_kill()* function provides a mechanism for asynchronously directing a signal at a  
 53819 thread in the calling process. This could be used, for example, by one thread to affect broadcast  
 53820 delivery of a signal to a set of threads.

53821 Note that *pthread\_kill()* only causes the signal to be handled in the context of the given thread;  
 53822 the signal action (termination or stopping) affects the process as a whole.

53823 **RATIONALE**

53824 If an implementation detects use of a thread ID after the end of its lifetime, it is recommended  
 53825 that the function should fail and report an [ESRCH] error.

53826 Existing implementations vary on the result of a *pthread\_kill()* with a thread ID indicating an +  
 53827 inactive thread (a terminated thread that has not been detached or joined). Some indicate success +  
 53828 on such a call, while others give an error of [ESRCH]. Since the definition of thread lifetime in +  
 53829 this volume of POSIX.1-2008 covers inactive threads, the [ESRCH] error as described is +  
 53830 inappropriate in this case. In particular, this means that an application cannot have one thread +  
 53831 check for termination of another with *pthread\_kill()*.

53832 **FUTURE DIRECTIONS**

53833 A future version of this standard may require that *pthread\_kill()* not fail with [ESRCH] in the |  
 53834 case of sending signals to an inactive thread (a terminated thread not yet detached or joined), |  
 53835 even though no signal will be delivered because the thread is no longer running.

53836 **SEE ALSO**53837 *kill()*, *pthread\_self()*, *raise()*53838 XBD *<signal.h>*

53839 **CHANGE HISTORY**

53840 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

53841 **Issue 6**

53842 The *pthread\_kill()* function is marked as part of the Threads option.

53843 The APPLICATION USAGE section is added.

53844 **Issue 7**

53845 The *pthread\_kill()* function is moved from the Threads option to the Base.

53846 Austin Group Interpretation 1003.1-2001 #142 is applied, removing the [ESRCH] error condition.

**53847 NAME**

53848 pthread\_mutex\_consistent — mark state protected by robust mutex as consistent

**53849 SYNOPSIS**

53850 #include <pthread.h>

53851 int pthread\_mutex\_consistent(pthread\_mutex\_t \*mutex);

**53852 DESCRIPTION**

53853 If *mutex* is a robust mutex in an inconsistent state, the *pthread\_mutex\_consistent()* function can be  
53854 used to mark the state protected by the mutex referenced by *mutex* as consistent again.

53855 If an owner of a robust mutex terminates while holding the mutex, the mutex becomes  
53856 inconsistent and the next thread that acquires the mutex lock shall be notified of the state by the  
53857 return value [EOWNERDEAD]. In this case, the mutex does not become normally usable again  
53858 until the state is marked consistent.

53859 If the thread which acquired the mutex lock with the return value [EOWNERDEAD] terminates  
53860 before calling either *pthread\_mutex\_consistent()* or *pthread\_mutex\_unlock()*, the next thread that  
53861 acquires the mutex lock shall be notified about the state of the mutex by the return value  
53862 [EOWNERDEAD].

53863 The behavior is undefined if the value specified by the *mutex* argument to  
53864 *pthread\_mutex\_consistent()* does not refer to an initialized mutex.

**53865 RETURN VALUE**

53866 Upon successful completion, the *pthread\_mutex\_consistent()* function shall return zero.  
53867 Otherwise, an error value shall be returned to indicate the error.

**53868 ERRORS**

53869 The *pthread\_mutex\_consistent()* function shall fail if:

53870 [EINVAL] The mutex object referenced by *mutex* is not robust or does not protect an  
53871 inconsistent state.

53872 These functions shall not return an error code of [EINTR].

**53873 EXAMPLES**

53874 None.

**53875 APPLICATION USAGE**

53876 The *pthread\_mutex\_consistent()* function is only responsible for notifying the implementation that  
53877 the state protected by the mutex has been recovered and that normal operations with the mutex  
53878 can be resumed. It is the responsibility of the application to recover the state so it can be reused.  
53879 If the application is not able to perform the recovery, it can notify the implementation that the  
53880 situation is unrecoverable by a call to *pthread\_mutex\_unlock()* without a prior call to  
53881 *pthread\_mutex\_consistent()*, in which case subsequent threads that attempt to lock the mutex will  
53882 fail to acquire the lock and be returned [ENOTRECOVERABLE].

**53883 RATIONALE**

53884 If an implementation detects that the value specified by the *mutex* argument to  
53885 *pthread\_mutex\_consistent()* does not refer to an initialized mutex, it is recommended that the  
53886 function should fail and report an [EINVAL] error.

**53887 FUTURE DIRECTIONS**

53888 None.

53889

53890

53891

53892

53893

SEE ALSO

pthread\_mutex\_lock(), pthread\_mutexattr\_getrobust()

XBD <pthread.h>

CHANGE HISTORY

First released in Issue 7.



**53894 NAME**

53895 pthread\_mutex\_destroy, pthread\_mutex\_init — destroy and initialize a mutex

**53896 SYNOPSIS**

```
53897 #include <pthread.h>
53898 int pthread_mutex_destroy(pthread_mutex_t *mutex);
53899 int pthread_mutex_init(pthread_mutex_t *restrict mutex,
53900     const pthread_mutexattr_t *restrict attr);
53901 pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
```

**53902 DESCRIPTION**

53903 The *pthread\_mutex\_destroy()* function shall destroy the mutex object referenced by *mutex*; the  
 53904 mutex object becomes, in effect, uninitialized. An implementation may cause  
 53905 *pthread\_mutex\_destroy()* to set the object referenced by *mutex* to an invalid value.

53906 A destroyed mutex object can be reinitialized using *pthread\_mutex\_init()*; the results of otherwise  
 53907 referencing the object after it has been destroyed are undefined.

53908 It shall be safe to destroy an initialized mutex that is unlocked. Attempting to destroy a locked  
 53909 mutex, or a mutex that another thread is attempting to lock, or a mutex that is being used in a  
 53910 *pthread\_cond\_timedwait()* or *pthread\_cond\_wait()* call by another thread, results in undefined  
 53911 behavior.

53912 The *pthread\_mutex\_init()* function shall initialize the mutex referenced by *mutex* with attributes  
 53913 specified by *attr*. If *attr* is NULL, the default mutex attributes are used; the effect shall be the  
 53914 same as passing the address of a default mutex attributes object. Upon successful initialization,  
 53915 the state of the mutex becomes initialized and unlocked.

53916 See [Section 2.9.9](#) (on page 523) for further requirements.

53917 Attempting to initialize an already initialized mutex results in undefined behavior.

53918 In cases where default mutex attributes are appropriate, the macro  
 53919 PTHREAD\_MUTEX\_INITIALIZER can be used to initialize mutexes. The effect shall be  
 53920 equivalent to dynamic initialization by a call to *pthread\_mutex\_init()* with parameter *attr*  
 53921 specified as NULL, except that no error checks are performed.

53922 The behavior is undefined if the value specified by the *mutex* argument to  
 53923 *pthread\_mutex\_destroy()* does not refer to an initialized mutex.

53924 The behavior is undefined if the value specified by the *attr* argument to *pthread\_mutex\_init()*  
 53925 does not refer to an initialized mutex attributes object.

**53926 RETURN VALUE**

53927 If successful, the *pthread\_mutex\_destroy()* and *pthread\_mutex\_init()* functions shall return zero;  
 53928 otherwise, an error number shall be returned to indicate the error.

**53929 ERRORS**

53930 The *pthread\_mutex\_init()* function shall fail if:

- |       |          |                                                                                            |
|-------|----------|--------------------------------------------------------------------------------------------|
| 53931 | [EAGAIN] | The system lacked the necessary resources (other than memory) to initialize another mutex. |
| 53932 |          |                                                                                            |
| 53933 | [ENOMEM] | Insufficient memory exists to initialize the mutex.                                        |
| 53934 | [EPERM]  | The caller does not have the privilege to perform the operation.                           |

53935 The *pthread\_mutex\_init()* function may fail if:

53936 [EINVAL] The attributes object referenced by *attr* has the robust mutex attribute set

53937 without the process-shared attribute being set.

53938 These functions shall not return an error code of [EINTR].

#### 53939 EXAMPLES

53940 None.

#### 53941 APPLICATION USAGE

53942 None.

#### 53943 RATIONALE

53944 If an implementation detects that the value specified by the *mutex* argument to

53945 *pthread\_mutex\_destroy()* does not refer to an initialized mutex, it is recommended that the

53946 function should fail and report an [EINVAL] error.

53947 If an implementation detects that the value specified by the *mutex* argument to

53948 *pthread\_mutex\_destroy()* or *pthread\_mutex\_init()* refers to a locked mutex or a mutex that is

53949 referenced (for example, while being used in a *pthread\_cond\_timedwait()* or *pthread\_cond\_wait()*)

53950 by another thread, or detects that the value specified by the *mutex* argument to

53951 *pthread\_mutex\_init()* refers to an already initialized mutex, it is recommended that the function

53952 should fail and report an [EBUSY] error.

53953 If an implementation detects that the value specified by the *attr* argument to

53954 *pthread\_mutex\_init()* does not refer to an initialized mutex attributes object, it is recommended

53955 that the function should fail and report an [EINVAL] error.

#### 53956 Alternate Implementations Possible

53957 This volume of POSIX.1-2008 supports several alternative implementations of mutexes. An

53958 implementation may store the lock directly in the object of type **pthread\_mutex\_t**. Alternatively,

53959 an implementation may store the lock in the heap and merely store a pointer, handle, or unique

53960 ID in the mutex object. Either implementation has advantages or may be required on certain

53961 hardware configurations. So that portable code can be written that is invariant to this choice, this

53962 volume of POSIX.1-2008 does not define assignment or equality for this type, and it uses the

53963 term “initialize” to reinforce the (more restrictive) notion that the lock may actually reside in the

53964 mutex object itself.

53965 Note that this precludes an over-specification of the type of the mutex or condition variable and

53966 motivates the opaqueness of the type.

53967 An implementation is permitted, but not required, to have *pthread\_mutex\_destroy()* store an

53968 illegal value into the mutex. This may help detect erroneous programs that try to lock (or

53969 otherwise reference) a mutex that has already been destroyed.

#### 53970 Tradeoff Between Error Checks and Performance Supported

53971 Many error conditions that can occur are not required to be detected by the implementation in

53972 order to let implementations trade off performance *versus* degree of error checking according to

53973 the needs of their specific applications and execution environment. As a general rule, conditions

53974 caused by the system (such as insufficient memory) are required to be detected, but conditions

53975 caused by an erroneously coded application (such as failing to provide adequate

53976 synchronization to prevent a mutex from being deleted while in use) are specified to result in

53977 undefined behavior.

53978 A wide range of implementations is thus made possible. For example, an implementation

intended for application debugging may implement all of the error checks, but an implementation running a single, provably correct application under very tight performance constraints in an embedded computer might implement minimal checks. An implementation might even be provided in two versions, similar to the options that compilers provide: a full-checking, but slower version; and a limited-checking, but faster version. To forbid this optionality would be a disservice to users.

By carefully limiting the use of “undefined behavior” only to things that an erroneous (badly coded) application might do, and by defining that resource-not-available errors are mandatory, this volume of POSIX.1-2008 ensures that a fully-conforming application is portable across the full range of implementations, while not forcing all implementations to add overhead to check for numerous things that a correct program never does. When the behavior is undefined, no error number is specified to be returned on implementations that do detect the condition. This is because undefined behavior means *anything* can happen, which includes returning with any value (which might happen to be a valid, but different, error number). However, since the error number might be useful to application developers when diagnosing problems during application development, a recommendation is made in rationale that implementors should return a particular error number if their implementation does detect the condition.

### Why No Limits are Defined

Defining symbols for the maximum number of mutexes and condition variables was considered but rejected because the number of these objects may change dynamically. Furthermore, many implementations place these objects into application memory; thus, there is no explicit maximum.

### Static Initializers for Mutexes and Condition Variables

Providing for static initialization of statically allocated synchronization objects allows modules with private static synchronization variables to avoid runtime initialization tests and overhead. Furthermore, it simplifies the coding of self-initializing modules. Such modules are common in C libraries, where for various reasons the design calls for self-initialization instead of requiring an explicit module initialization function to be called. An example use of static initialization follows.

Without static initialization, a self-initializing routine *foo()* might look as follows:

```
static pthread_once_t foo_once = PTHREAD_ONCE_INIT;
static pthread_mutex_t foo_mutex;

void foo_init()
{
    pthread_mutex_init(&foo_mutex, NULL);
}

void foo()
{
    pthread_once(&foo_once, foo_init);
    pthread_mutex_lock(&foo_mutex);
    /* Do work. */
    pthread_mutex_unlock(&foo_mutex);
}
```

With static initialization, the same routine could be coded as follows:

```
static pthread_mutex_t foo_mutex = PTHREAD_MUTEX_INITIALIZER;
```

```
54024 void foo()  
54025 {  
54026     pthread_mutex_lock(&foo_mutex);  
54027     /* Do work. */  
54028     pthread_mutex_unlock(&foo_mutex);  
54029 }
```

54030 Note that the static initialization both eliminates the need for the initialization test inside  
54031 *pthread\_once()* and the fetch of *&foo\_mutex* to learn the address to be passed to  
54032 *pthread\_mutex\_lock()* or *pthread\_mutex\_unlock()*.

54033 Thus, the C code written to initialize static objects is simpler on all systems and is also faster on a  
54034 large class of systems; those where the (entire) synchronization object can be stored in  
54035 application memory.

54036 Yet the locking performance question is likely to be raised for machines that require mutexes to  
54037 be allocated out of special memory. Such machines actually have to have mutexes and possibly  
54038 condition variables contain pointers to the actual hardware locks. For static initialization to work  
54039 on such machines, *pthread\_mutex\_lock()* also has to test whether or not the pointer to the actual  
54040 lock has been allocated. If it has not, *pthread\_mutex\_lock()* has to initialize it before use. The  
54041 reservation of such resources can be made when the program is loaded, and hence return codes  
54042 have not been added to mutex locking and condition variable waiting to indicate failure to  
54043 complete initialization.

54044 This runtime test in *pthread\_mutex\_lock()* would at first seem to be extra work; an extra test is  
54045 required to see whether the pointer has been initialized. On most machines this would actually  
54046 be implemented as a fetch of the pointer, testing the pointer against zero, and then using the  
54047 pointer if it has already been initialized. While the test might seem to add extra work, the extra  
54048 effort of testing a register is usually negligible since no extra memory references are actually  
54049 done. As more and more machines provide caches, the real expenses are memory references, not  
54050 instructions executed.

54051 Alternatively, depending on the machine architecture, there are often ways to eliminate *all*  
54052 overhead in the most important case: on the lock operations that occur *after* the lock has been  
54053 initialized. This can be done by shifting more overhead to the less frequent operation:  
54054 initialization. Since out-of-line mutex allocation also means that an address has to be  
54055 dereferenced to find the actual lock, one technique that is widely applicable is to have static  
54056 initialization store a bogus value for that address; in particular, an address that causes a machine  
54057 fault to occur. When such a fault occurs upon the first attempt to lock such a mutex, validity  
54058 checks can be done, and then the correct address for the actual lock can be filled in. Subsequent  
54059 lock operations incur no extra overhead since they do not “fault”. This is merely one technique  
54060 that can be used to support static initialization, while not adversely affecting the performance of  
54061 lock acquisition. No doubt there are other techniques that are highly machine-dependent.

54062 The locking overhead for machines doing out-of-line mutex allocation is thus similar for  
54063 modules being implicitly initialized, where it is improved for those doing mutex allocation  
54064 entirely inline. The inline case is thus made much faster, and the out-of-line case is not  
54065 significantly worse.

54066 Besides the issue of locking performance for such machines, a concern is raised that it is possible  
54067 that threads would serialize contending for initialization locks when attempting to finish  
54068 initializing statically allocated mutexes. (Such finishing would typically involve taking an  
54069 internal lock, allocating a structure, storing a pointer to the structure in the mutex, and releasing  
54070 the internal lock.) First, many implementations would reduce such serialization by hashing on  
54071 the mutex address. Second, such serialization can only occur a bounded number of times. In

particular, it can happen at most as many times as there are statically allocated synchronization objects. Dynamically allocated objects would still be initialized via *pthread\_mutex\_init()* or *pthread\_cond\_init()*.

Finally, if none of the above optimization techniques for out-of-line allocation yields sufficient performance for an application on some implementation, the application can avoid static initialization altogether by explicitly initializing all synchronization objects with the corresponding *pthread\_\*\_init()* functions, which are supported by all implementations. An implementation can also document the tradeoffs and advise which initialization technique is more efficient for that particular implementation.

### Destroying Mutexes

A mutex can be destroyed immediately after it is unlocked. However, since attempting to destroy a locked mutex, or a mutex that another thread is attempting to lock, or a mutex that is being used in a *pthread\_cond\_timedwait()* or *pthread\_cond\_wait()* call by another thread, results in undefined behavior, care must be taken to ensure that no other thread may be referencing the mutex.

### Robust Mutexes

Implementations are required to provide robust mutexes for mutexes with the process-shared attribute set to `PTHREAD_PROCESS_SHARED`. Implementations are allowed, but not required, to provide robust mutexes when the process-shared attribute is set to `PTHREAD_PROCESS_PRIVATE`.

### FUTURE DIRECTIONS

None.

### SEE ALSO

*pthread\_mutex\_getprioceiling()*, *pthread\_mutexattr\_getrobust()*, *pthread\_mutex\_lock()*,  
*pthread\_mutex\_timedlock()*, *pthread\_mutexattr\_getpshared()*

XBD [`<pthread.h>`](#)

### CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

#### Issue 6

The *pthread\_mutex\_destroy()* and *pthread\_mutex\_init()* functions are marked as part of the Threads option.

The *pthread\_mutex\_timedlock()* function is added to the SEE ALSO section for alignment with IEEE Std 1003.1d-1999.

IEEE PASC Interpretation 1003.1c #34 is applied, updating the DESCRIPTION.

The **restrict** keyword is added to the *pthread\_mutex\_init()* prototype for alignment with the ISO/IEC 9899: 1999 standard.

#### Issue 7

Changes are made from The Open Group Technical Standard, 2006, Extended API Set Part 3.

The *pthread\_mutex\_destroy()* and *pthread\_mutex\_init()* functions are moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized mutex or an uninitialized mutex attributes object is removed; this condition results in undefined behavior.

54114 The [EBUSY] error for a locked mutex, a mutex that is referenced, or an already initialized mutex  
54115 is removed; this condition results in undefined behavior.

54116 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0460 [70,428] is applied.



54117 **NAME**

54118 pthread\_mutex\_getprioceiling, pthread\_mutex\_setprioceiling — get and set the priority ceiling  
 54119 of a mutex (**REALTIME THREADS**)

54120 **SYNOPSIS**

```
54121 RPP|TPP #include <pthread.h>
54122
54122 int pthread_mutex_getprioceiling(const pthread_mutex_t *restrict mutex,
54123 int *restrict prioceiling);
54124
54124 int pthread_mutex_setprioceiling(pthread_mutex_t *restrict mutex,
54125 int prioceiling, int *restrict old_ceiling);
```

54126 **DESCRIPTION**

54127 The *pthread\_mutex\_getprioceiling()* function shall return the current priority ceiling of the mutex.

54128 The *pthread\_mutex\_setprioceiling()* function shall attempt to lock the mutex as if by a call to  
 54129 *pthread\_mutex\_lock()*, except that the process of locking the mutex need not adhere to the priority  
 54130 protect protocol. On acquiring the mutex it shall change the mutex's priority ceiling and then  
 54131 release the mutex as if by a call to *pthread\_mutex\_unlock()*. When the change is successful, the  
 54132 previous value of the priority ceiling shall be returned in *old\_ceiling*.

54133 If the *pthread\_mutex\_setprioceiling()* function fails, the mutex priority ceiling shall not be  
 54134 changed.

54135 **RETURN VALUE**

54136 If successful, the *pthread\_mutex\_getprioceiling()* and *pthread\_mutex\_setprioceiling()* functions shall  
 54137 return zero; otherwise, an error number shall be returned to indicate the error.

54138 **ERRORS**

54139 These functions shall fail if:

54140 [EINVAL] The protocol attribute of *mutex* is PTHREAD\_PRIO\_NONE.

54141 [EPERM] The implementation requires appropriate privileges to perform the operation  
 54142 and the caller does not have appropriate privileges.

54143 The *pthread\_mutex\_setprioceiling()* function shall fail if:

54144 [EAGAIN] The mutex could not be acquired because the maximum number of recursive  
 54145 locks for *mutex* has been exceeded.

54146 [EDEADLK] The mutex type is PTHREAD\_MUTEX\_ERRORCHECK and the current  
 54147 thread already owns the mutex.

54148 [EINVAL] The mutex was created with the protocol attribute having the value  
 54149 PTHREAD\_PRIO\_PROTECT and the calling thread's priority is higher than  
 54150 the mutex's current priority ceiling, and the implementation adheres to the  
 54151 priority protect protocol in the process of locking the mutex.

54152 [ENOTRECOVERABLE]

54153 The mutex is a robust mutex and the state protected by the mutex is not  
 54154 recoverable.

54155 [EOWNERDEAD]

54156 The mutex is a robust mutex and the process containing the previous owning  
 54157 thread terminated while holding the mutex lock. The mutex lock shall be  
 54158 acquired by the calling thread and it is up to the new owner to make the state  
 54159 consistent (see *pthread\_mutex\_lock()*).

54160 The *pthread\_mutex\_setprioceiling()* function may fail if:

54161 [EDEADLK] A deadlock condition was detected.

54162 [EINVAL] The priority requested by *prioceiling* is out of range.

54163 [EOWNERDEAD]

54164 The mutex is a robust mutex and the previous owning thread terminated

54165 while holding the mutex lock. The mutex lock shall be acquired by the calling

54166 thread and it is up to the new owner to make the state consistent (see

54167 *pthread\_mutex\_lock()*).

54168 These functions shall not return an error code of [EINTR].

#### 54169 EXAMPLES

54170 None.

#### 54171 APPLICATION USAGE

54172 None.

#### 54173 RATIONALE

54174 None.

#### 54175 FUTURE DIRECTIONS

54176 None.

#### 54177 SEE ALSO

54178 *pthread\_mutex\_destroy()*, *pthread\_mutex\_lock()*, *pthread\_mutex\_timedlock()*

54179 XBD <pthread.h>

#### 54180 CHANGE HISTORY

54181 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

54182 Marked as part of the Realtime Threads Feature Group.

#### 54183 Issue 6

54184 The *pthread\_mutex\_getprioceiling()* and *pthread\_mutex\_setprioceiling()* functions are marked as

54185 part of the Threads and Thread Priority Protection options.

54186 The [ENOSYS] error conditions have been removed.

54187 The *pthread\_mutex\_timedlock()* function is added to the SEE ALSO section for alignment with

54188 IEEE Std 1003.1d-1999.

54189 The **restrict** keyword is added to the *pthread\_mutex\_getprioceiling()* and

54190 *pthread\_mutex\_setprioceiling()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

#### 54191 Issue 7

54192 SD5-XSH-ERN-39 is applied.

54193 Austin Group Interpretation 1003.1-2001 #052 is applied, adding [EDEADLK] as a “may fail”

54194 error.

54195 SD5-XSH-ERN-158 is applied, updating the ERRORS section to include a “shall fail” error case

54196 for when the protocol attribute of *mutex* is PTHREAD\_PRIO\_NONE.

54197 The *pthread\_mutex\_getprioceiling()* and *pthread\_mutex\_setprioceiling()* functions are moved from

54198 the Threads option to require support of either the Robust Mutex Priority Protection option or

54199 the Non-Robust Mutex Priority Protection option.



54200  
54201

The DESCRIPTION and ERRORS sections are updated to account properly for all of the various mutex types.

54202 **NAME**

54203 pthread\_mutex\_init — destroy and initialize a mutex

54204 **SYNOPSIS**

54205 #include &lt;pthread.h&gt;

54206 int pthread\_mutex\_init(pthread\_mutex\_t \*restrict mutex,

54207 const pthread\_mutexattr\_t \*restrict attr);

54208 pthread\_mutex\_t mutex = PTHREAD\_MUTEX\_INITIALIZER;

54209 **DESCRIPTION**54210 Refer to *pthread\_mutex\_destroy()*.

**NAME**

pthread\_mutex\_lock, pthread\_mutex\_trylock, pthread\_mutex\_unlock — lock and unlock a mutex

**SYNOPSIS**

```
#include <pthread.h>

int pthread_mutex_lock(pthread_mutex_t *mutex);
int pthread_mutex_trylock(pthread_mutex_t *mutex);
int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

**DESCRIPTION**

The mutex object referenced by *mutex* shall be locked by a call to *pthread\_mutex\_lock()* that returns zero or [EOWNERDEAD]. If the mutex is already locked by another thread, the calling thread shall block until the mutex becomes available. This operation shall return with the mutex object referenced by *mutex* in the locked state with the calling thread as its owner. If a thread attempts to relock a mutex that it has already locked, *pthread\_mutex\_lock()* shall behave as described in the **Relock** column of the following table. If a thread attempts to unlock a mutex that it has not locked or a mutex which is unlocked, *pthread\_mutex\_unlock()* shall behave as described in the **Unlock When Not Owner** column of the following table.

| Mutex Type | Robustness | Relock                   | Unlock When Not Owner |
|------------|------------|--------------------------|-----------------------|
| NORMAL     | non-robust | deadlock                 | undefined behavior    |
| NORMAL     | robust     | deadlock                 | error returned        |
| ERRORCHECK | either     | error returned           | error returned        |
| RECURSIVE  | either     | recursive<br>(see below) | error returned        |
| DEFAULT    | non-robust | undefined<br>behavior†   | undefined behavior†   |
| DEFAULT    | robust     | undefined<br>behavior†   | error returned        |

† If the mutex type is PTHREAD\_MUTEX\_DEFAULT, the behavior of *pthread\_mutex\_lock()* may correspond to one of the three other standard mutex types as described in the table above. If it does not correspond to one of those three, the behavior is undefined for the cases marked †.

Where the table indicates recursive behavior, the mutex shall maintain the concept of a lock count. When a thread successfully acquires a mutex for the first time, the lock count shall be set to one. Every time a thread relocks this mutex, the lock count shall be incremented by one. Each time the thread unlocks the mutex, the lock count shall be decremented by one. When the lock count reaches zero, the mutex shall become available for other threads to acquire.

The *pthread\_mutex\_trylock()* function shall be equivalent to *pthread\_mutex\_lock()*, except that if the mutex object referenced by *mutex* is currently locked (by any thread, including the current thread), the call shall return immediately. If the mutex type is PTHREAD\_MUTEX\_RECURSIVE and the mutex is currently owned by the calling thread, the mutex lock count shall be incremented by one and the *pthread\_mutex\_trylock()* function shall immediately return success.

The *pthread\_mutex\_unlock()* function shall release the mutex object referenced by *mutex*. The manner in which a mutex is released is dependent upon the mutex's type attribute. If there are threads blocked on the mutex object referenced by *mutex* when *pthread\_mutex\_unlock()* is called, resulting in the mutex becoming available, the scheduling policy shall determine which thread shall acquire the mutex.

54257 (In the case of PTHREAD\_MUTEX\_RECURSIVE mutexes, the mutex shall become available  
54258 when the count reaches zero and the calling thread no longer has any locks on this mutex.)

54259 If a signal is delivered to a thread waiting for a mutex, upon return from the signal handler the  
54260 thread shall resume waiting for the mutex as if it was not interrupted.

54261 If *mutex* is a robust mutex and the process containing the owning thread terminated while  
54262 holding the mutex lock, a call to *pthread\_mutex\_lock()* shall return the error value  
54263 [EOWNERDEAD]. If *mutex* is a robust mutex and the owning thread terminated while holding  
54264 the mutex lock, a call to *pthread\_mutex\_lock()* may return the error value [EOWNERDEAD] even  
54265 if the process in which the owning thread resides has not terminated. In these cases, the mutex is  
54266 locked by the thread but the state it protects is marked as inconsistent. The application should  
54267 ensure that the state is made consistent for reuse and when that is complete call  
54268 *pthread\_mutex\_consistent()*. If the application is unable to recover the state, it should unlock the  
54269 mutex without a prior call to *pthread\_mutex\_consistent()*, after which the mutex is marked  
54270 permanently unusable.

54271 If *mutex* does not refer to an initialized mutex object, the behavior of *pthread\_mutex\_lock()*,  
54272 *pthread\_mutex\_trylock()*, and *pthread\_mutex\_unlock()* is undefined.

#### 54273 RETURN VALUE

54274 If successful, the *pthread\_mutex\_lock()*, *pthread\_mutex\_trylock()*, and *pthread\_mutex\_unlock()*  
54275 functions shall return zero; otherwise, an error number shall be returned to indicate the error.

#### 54276 ERRORS

54277 The *pthread\_mutex\_lock()* and *pthread\_mutex\_trylock()* functions shall fail if:

54278 [EAGAIN] The mutex could not be acquired because the maximum number of recursive  
54279 locks for *mutex* has been exceeded.

54280 RPP|TPP [EINVAL] The *mutex* was created with the protocol attribute having the value  
54281 PTHREAD\_PRIO\_PROTECT and the calling thread's priority is higher than  
54282 the mutex's current priority ceiling.

54283 [ENOTRECOVERABLE]  
54284 The state protected by the mutex is not recoverable.

54285 [EOWNERDEAD]  
54286 The mutex is a robust mutex and the process containing the previous owning  
54287 thread terminated while holding the mutex lock. The mutex lock shall be  
54288 acquired by the calling thread and it is up to the new owner to make the state  
54289 consistent.

54290 The *pthread\_mutex\_lock()* function shall fail if:

54291 [EDEADLK] The mutex type is PTHREAD\_MUTEX\_ERRORCHECK and the current  
54292 thread already owns the mutex.

54293 The *pthread\_mutex\_trylock()* function shall fail if:

54294 [EBUSY] The *mutex* could not be acquired because it was already locked.

54295 The *pthread\_mutex\_unlock()* function shall fail if:

54296 [EPERM] The mutex type is PTHREAD\_MUTEX\_ERRORCHECK or  
54297 PTHREAD\_MUTEX\_RECURSIVE, or the mutex is a robust mutex, and the  
54298 current thread does not own the mutex.

54299 The *pthread\_mutex\_lock()* and *pthread\_mutex\_trylock()* functions may fail if:

54300 [EOWNERDEAD]  
54301 The mutex is a robust mutex and the previous owning thread terminated  
54302 while holding the mutex lock. The mutex lock shall be acquired by the calling  
54303 thread and it is up to the new owner to make the state consistent.

54304 The *pthread\_mutex\_lock()* function may fail if:

54305 [EDEADLK] A deadlock condition was detected.

54306 These functions shall not return an error code of [EINTR].

#### 54307 **EXAMPLES**

54308 None.

#### 54309 **APPLICATION USAGE**

54310 Applications that have assumed that non-zero return values are errors will need updating for  
54311 use with robust mutexes, since a valid return for a thread acquiring a mutex which is protecting  
54312 a currently inconsistent state is [EOWNERDEAD]. Applications that do not check the error  
54313 returns, due to ruling out the possibility of such errors arising, should not use robust mutexes. If  
54314 an application is supposed to work with normal and robust mutexes it should check all return  
54315 values for error conditions and if necessary take appropriate action.

#### 54316 **RATIONALE**

54317 Mutex objects are intended to serve as a low-level primitive from which other thread  
54318 synchronization functions can be built. As such, the implementation of mutexes should be as  
54319 efficient as possible, and this has ramifications on the features available at the interface.

54320 The mutex functions and the particular default settings of the mutex attributes have been  
54321 motivated by the desire to not preclude fast, inlined implementations of mutex locking and  
54322 unlocking.

54323 Since most attributes only need to be checked when a thread is going to be blocked, the use of  
54324 attributes does not slow the (common) mutex-locking case.

54325 Likewise, while being able to extract the thread ID of the owner of a mutex might be desirable, it  
54326 would require storing the current thread ID when each mutex is locked, and this could incur  
54327 unacceptable levels of overhead. Similar arguments apply to a *mutex\_tryunlock* operation.

54328 For further rationale on the extended mutex types, see XRAT [Threads Extensions](#) (on page 3631).

54329 If an implementation detects that the value specified by the *mutex* argument does not refer to an  
54330 initialized mutex object, it is recommended that the function should fail and report an [EINVAL]  
54331 error.

#### 54332 **FUTURE DIRECTIONS**

54333 None.

#### 54334 **SEE ALSO**

54335 *pthread\_mutex\_consistent()*, *pthread\_mutex\_destroy()*, *pthread\_mutex\_timedlock()*,  
54336 *pthread\_mutexattr\_getrobust()*

54337 XBD [Section 4.12](#) (on page 111), [<pthread.h>](#)

#### 54338 **CHANGE HISTORY**

54339 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

**Issue 6**

The *pthread\_mutex\_lock()*, *pthread\_mutex\_trylock()*, and *pthread\_mutex\_unlock()* functions are marked as part of the Threads option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The behavior when attempting to relock a mutex is defined.

The *pthread\_mutex\_timedlock()* function is added to the SEE ALSO section for alignment with IEEE Std 1003.1d-1999.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/98 is applied, updating the ERRORS section so that the [EDEADLK] error includes detection of a deadlock condition. The RATIONALE section is also reworded to take into account non-XSI-conformant systems.

**Issue 7**

SD5-XSH-ERN-43 is applied, marking the “shall fail” case of the [EINVAL] error as dependent on the Thread Priority Protection option.

Changes are made from The Open Group Technical Standard, 2006, Extended API Set Part 3.

The *pthread\_mutex\_lock()*, *pthread\_mutex\_trylock()*, and *pthread\_mutex\_unlock()* functions are moved from the Threads option to the Base.

The following extended mutex types are moved from the XSI option to the Base:

PTHREAD\_MUTEX\_NORMAL  
PTHREAD\_MUTEX\_ERRORCHECK  
PTHREAD\_MUTEX\_RECURSIVE  
PTHREAD\_MUTEX\_DEFAULT

The DESCRIPTION is updated to clarify the behavior when *mutex* does not refer to an initialized mutex.

The ERRORS section is updated to account properly for all of the various mutex types.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0461 [121], XSH/TC1-2008/0462 [92,428], and XSH/TC1-2008/0463 [121] are applied.

54367 **NAME**

54368 pthread\_mutex\_setprioceiling — change the priority ceiling of a mutex (REALTIME  
54369 THREADS)

54370 **SYNOPSIS**

```
54371 RPP|TPP #include <pthread.h>  
54372 int pthread_mutex_setprioceiling(pthread_mutex_t *restrict mutex,  
54373 int prioceiling, int *restrict old_ceiling);
```

54374 **DESCRIPTION**

54375 Refer to *pthread\_mutex\_getprioceiling()*.

**NAME**

pthread\_mutex\_timedlock — lock a mutex

**SYNOPSIS**

```
#include <pthread.h>
#include <time.h>
```

```
int pthread_mutex_timedlock(pthread_mutex_t *restrict mutex,
    const struct timespec *restrict abstime);
```

**DESCRIPTION**

The *pthread\_mutex\_timedlock()* function shall lock the mutex object referenced by *mutex*. If the mutex is already locked, the calling thread shall block until the mutex becomes available as in the *pthread\_mutex\_lock()* function. If the mutex cannot be locked without waiting for another thread to unlock the mutex, this wait shall be terminated when the specified timeout expires.

The timeout shall expire when the absolute time specified by *abstime* passes, as measured by the clock on which timeouts are based (that is, when the value of that clock equals or exceeds *abstime*), or if the absolute time specified by *abstime* has already been passed at the time of the call.

The timeout shall be based on the CLOCK\_REALTIME clock. The resolution of the timeout shall be the resolution of the clock on which it is based. The **timespec** data type is defined in the **<time.h>** header.

Under no circumstance shall the function fail with a timeout if the mutex can be locked immediately. The validity of the *abstime* parameter need not be checked if the mutex can be locked immediately.

RPI | TPI

As a consequence of the priority inheritance rules (for mutexes initialized with the Prio\_Inherit protocol), if a timed mutex wait is terminated because its timeout expires, the priority of the owner of the mutex shall be adjusted as necessary to reflect the fact that this thread is no longer among the threads waiting for the mutex.

If *mutex* is a robust mutex and the process containing the owning thread terminated while holding the mutex lock, a call to *pthread\_mutex\_timedlock()* shall return the error value [EOWNERDEAD]. If *mutex* is a robust mutex and the owning thread terminated while holding the mutex lock, a call to *pthread\_mutex\_timedlock()* may return the error value [EOWNERDEAD] even if the process in which the owning thread resides has not terminated. In these cases, the mutex is locked by the thread but the state it protects is marked as inconsistent. The application should ensure that the state is made consistent for reuse and when that is complete call *pthread\_mutex\_consistent()*. If the application is unable to recover the state, it should unlock the mutex without a prior call to *pthread\_mutex\_consistent()*, after which the mutex is marked permanently unusable.

If *mutex* does not refer to an initialized mutex object, the behavior is undefined.

**RETURN VALUE**

If successful, the *pthread\_mutex\_timedlock()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

**ERRORS**

The *pthread\_mutex\_timedlock()* function shall fail if:

[EAGAIN]           The mutex could not be acquired because the maximum number of recursive locks for *mutex* has been exceeded.



54420 [EDEADLK] The mutex type is PTHREAD\_MUTEX\_ERRORCHECK and the current  
 54421 thread already owns the mutex.

54422 [EINVAL] The mutex was created with the protocol attribute having the value  
 54423 PTHREAD\_PRIO\_PROTECT and the calling thread's priority is higher than  
 54424 the mutex' current priority ceiling.

54425 [EINVAL] The process or thread would have blocked, and the *abstime* parameter  
 54426 specified a nanoseconds field value less than zero or greater than or equal to  
 54427 1 000 million.

54428 [ENOTRECOVERABLE]  
 54429 The state protected by the mutex is not recoverable.

54430 [EOWNERDEAD]  
 54431 The mutex is a robust mutex and the process containing the previous owning  
 54432 thread terminated while holding the mutex lock. The mutex lock shall be  
 54433 acquired by the calling thread and it is up to the new owner to make the state  
 54434 consistent.

54435 [ETIMEDOUT] The mutex could not be locked before the specified timeout expired.

54436 The *pthread\_mutex\_timedlock()* function may fail if:

54437 [EDEADLK] A deadlock condition was detected.

54438 [EOWNERDEAD]  
 54439 The mutex is a robust mutex and the previous owning thread terminated  
 54440 while holding the mutex lock. The mutex lock shall be acquired by the calling  
 54441 thread and it is up to the new owner to make the state consistent.

54442 This function shall not return an error code of [EINTR].

**EXAMPLES**

54443 None.  
 54444

**APPLICATION USAGE**

54446 Applications that have assumed that non-zero return values are errors will need updating for  
 54447 use with robust mutexes, since a valid return for a thread acquiring a mutex which is protecting  
 54448 a currently inconsistent state is [EOWNERDEAD]. Applications that do not check the error  
 54449 returns, due to ruling out the possibility of such errors arising, should not use robust mutexes. If  
 54450 an application is supposed to work with normal and robust mutexes, it should check all return  
 54451 values for error conditions and if necessary take appropriate action.

**RATIONALE**

54452 Refer to *pthread\_mutex\_lock()*.  
 54453

**FUTURE DIRECTIONS**

54454 None.  
 54455

**SEE ALSO**

54456 *pthread\_mutex\_destroy()*, *pthread\_mutex\_lock()*, *time()*  
 54457  
 54458 XBD Section 4.12 (on page 111), **<pthread.h>**, **<time.h>**

**CHANGE HISTORY**

54459 First released in Issue 6. Derived from IEEE Std 1003.1d-1999.  
 54460  
 54461 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/99 is applied, marking the last paragraph  
 54462 in the DESCRIPTION as part of the Thread Priority Inheritance option.

- 54463 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/100 is applied, updating the ERRORS  
54464 section so that the [EDEADLK] error includes detection of a deadlock condition.
- 54465 **Issue 7**
- 54466 Changes are made from The Open Group Technical Standard, 2006, Extended API Set Part 3.
- 54467 The *pthread\_mutex\_timedlock()* function is moved from the Timeouts option to the Base.
- 54468 Functionality relating to the Timers option is moved to the Base.
- 54469 The DESCRIPTION is updated to clarify the behavior when *mutex* does not refer to an initialized  
54470 mutex.
- 54471 The ERRORS section is updated to account properly for all of the various mutex types.

54472 **NAME**

54473 pthread\_mutex\_trylock, pthread\_mutex\_unlock — lock and unlock a mutex

54474 **SYNOPSIS**

54475 #include &lt;pthread.h&gt;

54476 int pthread\_mutex\_trylock(pthread\_mutex\_t \*mutex);

54477 int pthread\_mutex\_unlock(pthread\_mutex\_t \*mutex);

54478 **DESCRIPTION**54479 Refer to *pthread\_mutex\_lock()*.

**54480 NAME**

54481 pthread\_mutexattr\_destroy, pthread\_mutexattr\_init — destroy and initialize the mutex  
 54482 attributes object

**54483 SYNOPSIS**

54484 #include <pthread.h>  
 54485 int pthread\_mutexattr\_destroy(pthread\_mutexattr\_t \*attr);  
 54486 int pthread\_mutexattr\_init(pthread\_mutexattr\_t \*attr);

**54487 DESCRIPTION**

54488 The *pthread\_mutexattr\_destroy()* function shall destroy a mutex attributes object; the object  
 54489 becomes, in effect, uninitialized. An implementation may cause *pthread\_mutexattr\_destroy()* to  
 54490 set the object referenced by *attr* to an invalid value.

54491 A destroyed *attr* attributes object can be reinitialized using *pthread\_mutexattr\_init()*; the results of  
 54492 otherwise referencing the object after it has been destroyed are undefined.

54493 The *pthread\_mutexattr\_init()* function shall initialize a mutex attributes object *attr* with the  
 54494 default value for all of the attributes defined by the implementation.

54495 Results are undefined if *pthread\_mutexattr\_init()* is called specifying an already initialized *attr*  
 54496 attributes object.

54497 After a mutex attributes object has been used to initialize one or more mutexes, any function  
 54498 affecting the attributes object (including destruction) shall not affect any previously initialized  
 54499 mutexes.

54500 The behavior is undefined if the value specified by the *attr* argument to  
 54501 *pthread\_mutexattr\_destroy()* does not refer to an initialized mutex attributes object.

**54502 RETURN VALUE**

54503 Upon successful completion, *pthread\_mutexattr\_destroy()* and *pthread\_mutexattr\_init()* shall  
 54504 return zero; otherwise, an error number shall be returned to indicate the error.

**54505 ERRORS**

54506 The *pthread\_mutexattr\_init()* function shall fail if:

54507 [ENOMEM] Insufficient memory exists to initialize the mutex attributes object.

54508 These functions shall not return an error code of [EINTR].

**54509 EXAMPLES**

54510 None.

**54511 APPLICATION USAGE**

54512 None.

**54513 RATIONALE**

54514 If an implementation detects that the value specified by the *attr* argument to  
 54515 *pthread\_mutexattr\_destroy()* does not refer to an initialized mutex attributes object, it is  
 54516 recommended that the function should fail and report an [EINVAL] error.

54517 See *pthread\_attr\_destroy()* for a general explanation of attributes. Attributes objects allow  
 54518 implementations to experiment with useful extensions and permit extension of this volume of  
 54519 POSIX.1-2008 without changing the existing functions. Thus, they provide for future  
 54520 extensibility of this volume of POSIX.1-2008 and reduce the temptation to standardize  
 54521 prematurely on semantics that are not yet widely implemented or understood.

54522 Examples of possible additional mutex attributes that have been discussed are *spin\_only*,  
 54523 *limited\_spin*, *no\_spin*, *recursive*, and *metered*. (To explain what the latter attributes might mean:

recursive mutexes would allow for multiple re-locking by the current owner; metered mutexes would transparently keep records of queue length, wait time, and so on.) Since there is not yet wide agreement on the usefulness of these resulting from shared implementation and usage experience, they are not yet specified in this volume of POSIX.1-2008. Mutex attributes objects, however, make it possible to test out these concepts for possible standardization at a later time.

### Mutex Attributes and Performance

Care has been taken to ensure that the default values of the mutex attributes have been defined such that mutexes initialized with the defaults have simple enough semantics so that the locking and unlocking can be done with the equivalent of a test-and-set instruction (plus possibly a few other basic instructions).

There is at least one implementation method that can be used to reduce the cost of testing at lock-time if a mutex has non-default attributes. One such method that an implementation can employ (and this can be made fully transparent to fully conforming POSIX applications) is to secretly pre-lock any mutexes that are initialized to non-default attributes. Any later attempt to lock such a mutex causes the implementation to branch to the “slow path” as if the mutex were unavailable; then, on the slow path, the implementation can do the “real work” to lock a non-default mutex. The underlying unlock operation is more complicated since the implementation never really wants to release the pre-lock on this kind of mutex. This illustrates that, depending on the hardware, there may be certain optimizations that can be used so that whatever mutex attributes are considered “most frequently used” can be processed most efficiently.

### Process Shared Memory and Synchronization

The existence of memory mapping functions in this volume of POSIX.1-2008 leads to the possibility that an application may allocate the synchronization objects from this section in memory that is accessed by multiple processes (and therefore, by threads of multiple processes).

In order to permit such usage, while at the same time keeping the usual case (that is, usage within a single process) efficient, a *process-shared* option has been defined.

If an implementation supports the `_POSIX_THREAD_PROCESS_SHARED` option, then the *process-shared* attribute can be used to indicate that mutexes or condition variables may be accessed by threads of multiple processes.

The default setting of `PTHREAD_PROCESS_PRIVATE` has been chosen for the *process-shared* attribute so that the most efficient forms of these synchronization objects are created by default.

Synchronization variables that are initialized with the `PTHREAD_PROCESS_PRIVATE` *process-shared* attribute may only be operated on by threads in the process that initialized them. Synchronization variables that are initialized with the `PTHREAD_PROCESS_SHARED` *process-shared* attribute may be operated on by any thread in any process that has access to it. In particular, these processes may exist beyond the lifetime of the initializing process. For example, the following code implements a simple counting semaphore in a mapped file that may be used by many processes.

```
/* sem.h */
struct semaphore {
    pthread_mutex_t lock;
    pthread_cond_t nonzero;
    unsigned count;
};
typedef struct semaphore semaphore_t;
```

```

54569 semaphore_t *semaphore_create(char *semaphore_name);
54570 semaphore_t *semaphore_open(char *semaphore_name);
54571 void semaphore_post(semaphore_t *semap);
54572 void semaphore_wait(semaphore_t *semap);
54573 void semaphore_close(semaphore_t *semap);

54574 /* sem.c */
54575 #include <sys/types.h>
54576 #include <sys/stat.h>
54577 #include <sys/mman.h>
54578 #include <fcntl.h>
54579 #include <pthread.h>
54580 #include "sem.h"

54581 semaphore_t *
54582 semaphore_create(char *semaphore_name)
54583 {
54584     int fd;
54585     semaphore_t *semap;
54586     pthread_mutexattr_t psharedm;
54587     pthread_condattr_t psharedc;

54588     fd = open(semaphore_name, O_RDWR | O_CREAT | O_EXCL, 0666);
54589     if (fd < 0)
54590         return (NULL);
54591     (void) ftruncate(fd, sizeof(semaphore_t));
54592     (void) pthread_mutexattr_init(&psharedm);
54593     (void) pthread_mutexattr_setpshared(&psharedm,
54594         PTHREAD_PROCESS_SHARED);
54595     (void) pthread_condattr_init(&psharedc);
54596     (void) pthread_condattr_setpshared(&psharedc,
54597         PTHREAD_PROCESS_SHARED);
54598     semap = (semaphore_t *) mmap(NULL, sizeof(semaphore_t),
54599         PROT_READ | PROT_WRITE, MAP_SHARED,
54600         fd, 0);
54601     close (fd);
54602     (void) pthread_mutex_init(&semap->lock, &psharedm);
54603     (void) pthread_cond_init(&semap->nonzero, &psharedc);
54604     semap->count = 0;
54605     return (semap);
54606 }

54607 semaphore_t *
54608 semaphore_open(char *semaphore_name)
54609 {
54610     int fd;
54611     semaphore_t *semap;

54612     fd = open(semaphore_name, O_RDWR, 0666);
54613     if (fd < 0)
54614         return (NULL);
54615     semap = (semaphore_t *) mmap(NULL, sizeof(semaphore_t),
54616         PROT_READ | PROT_WRITE, MAP_SHARED,
54617         fd, 0);

```

```

54618         close (fd);
54619         return (semap);
54620     }

54621     void
54622     semaphore_post(semaphore_t *semap)
54623     {
54624         pthread_mutex_lock(&semap->lock);
54625         if (semap->count == 0)
54626             pthread_cond_signal(&semap->nonzero);
54627         semap->count++;
54628         pthread_mutex_unlock(&semap->lock);
54629     }

54630     void
54631     semaphore_wait(semaphore_t *semap)
54632     {
54633         pthread_mutex_lock(&semap->lock);
54634         while (semap->count == 0)
54635             pthread_cond_wait(&semap->nonzero, &semap->lock);
54636         semap->count--;
54637         pthread_mutex_unlock(&semap->lock);
54638     }

54639     void
54640     semaphore_close(semaphore_t *semap)
54641     {
54642         munmap((void *) semap, sizeof(semaphore_t));
54643     }

```

The following code is for three separate processes that create, post, and wait on a semaphore in the file **/tmp/semaphore**. Once the file is created, the post and wait programs increment and decrement the counting semaphore (waiting and waking as required) even though they did not initialize the semaphore.

```

54648     /* create.c */
54649     #include "pthread.h"
54650     #include "sem.h"

54651     int
54652     main()
54653     {
54654         semaphore_t *semap;

54655         semap = semaphore_create("/tmp/semaphore");
54656         if (semap == NULL)
54657             exit(1);
54658         semaphore_close(semap);
54659         return (0);
54660     }

54661     /* post */
54662     #include "pthread.h"
54663     #include "sem.h"

54664     int

```

```

54665     main()
54666     {
54667         semaphore_t *semaph;
54668
54669         semaph = semaphore_open("/tmp/semaphore");
54670         if (semaph == NULL)
54671             exit(1);
54672         semaphore_post(semaph);
54673         semaphore_close(semaph);
54674         return (0);
54675     }
54676
54677     /* wait */
54678     #include "pthread.h"
54679     #include "sem.h"
54680
54681     int
54682     main()
54683     {
54684         semaphore_t *semaph;
54685
54686         semaph = semaphore_open("/tmp/semaphore");
54687         if (semaph == NULL)
54688             exit(1);
54689         semaphore_wait(semaph);
54690         semaphore_close(semaph);
54691         return (0);
54692     }

```

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*pthread\_cond\_destroy()*, *pthread\_create()*, *pthread\_mutex\_destroy()*

XBD <pthread.h>

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

**Issue 6**

The *pthread\_mutexattr\_destroy()* and *pthread\_mutexattr\_init()* functions are marked as part of the Threads option.

IEEE PASC Interpretation 1003.1c #27 is applied, updating the ERRORS section.

**Issue 7**

The *pthread\_mutexattr\_destroy()* and *pthread\_mutexattr\_init()* functions are moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized mutex attributes object is removed; this condition results in undefined behavior.



**54705 NAME**

54706 pthread\_mutexattr\_getprioceiling, pthread\_mutexattr\_setprioceiling — get and set the  
 54707 prioceiling attribute of the mutex attributes object (**REALTIME THREADS**)

**54708 SYNOPSIS**

```
54709 RPP|TPP #include <pthread.h>
54710
54710 int pthread_mutexattr_getprioceiling(const pthread_mutexattr_t
54711     *restrict attr, int *restrict prioceiling);
54712 int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *attr,
54713     int prioceiling);
```

**54714 DESCRIPTION**

54715 The *pthread\_mutexattr\_getprioceiling()* and *pthread\_mutexattr\_setprioceiling()* functions,  
 54716 respectively, shall get and set the priority ceiling attribute of a mutex attributes object pointed to  
 54717 by *attr* which was previously created by the function *pthread\_mutexattr\_init()*.

54718 The *prioceiling* attribute contains the priority ceiling of initialized mutexes. The values of  
 54719 *prioceiling* are within the maximum range of priorities defined by SCHED\_FIFO.

54720 The *prioceiling* attribute defines the priority ceiling of initialized mutexes, which is the minimum  
 54721 priority level at which the critical section guarded by the mutex is executed. In order to avoid  
 54722 priority inversion, the priority ceiling of the mutex shall be set to a priority higher than or equal  
 54723 to the highest priority of all the threads that may lock that mutex. The values of *prioceiling* are  
 54724 within the maximum range of priorities defined under the SCHED\_FIFO scheduling policy.

54725 The behavior is undefined if the value specified by the *attr* argument to  
 54726 *pthread\_mutexattr\_getprioceiling()* or *pthread\_mutexattr\_setprioceiling()* does not refer to an  
 54727 initialized mutex attributes object.

**54728 RETURN VALUE**

54729 Upon successful completion, the *pthread\_mutexattr\_getprioceiling()* and  
 54730 *pthread\_mutexattr\_setprioceiling()* functions shall return zero; otherwise, an error number shall be  
 54731 returned to indicate the error.

**54732 ERRORS**

54733 These functions may fail if:

54734 [EINVAL] The value specified by *prioceiling* is invalid.

54735 [EPERM] The caller does not have the privilege to perform the operation.

54736 These functions shall not return an error code of [EINTR].

**54737 EXAMPLES**

54738 None.

**54739 APPLICATION USAGE**

54740 None.

**54741 RATIONALE**

54742 If an implementation detects that the value specified by the *attr* argument to  
 54743 *pthread\_mutexattr\_getprioceiling()* or *pthread\_mutexattr\_setprioceiling()* does not refer to an  
 54744 initialized mutex attributes object, it is recommended that the function should fail and report an  
 54745 [EINVAL] error.

54746 **FUTURE DIRECTIONS**

54747 None.

54748 **SEE ALSO**54749 *pthread\_cond\_destroy()*, *pthread\_create()*, *pthread\_mutex\_destroy()*

54750 XBD &lt;pthread.h&gt;

54751 **CHANGE HISTORY**

54752 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

54753 Marked as part of the Realtime Threads Feature Group.

54754 **Issue 6**54755 The *pthread\_mutexattr\_getprioceiling()* and *pthread\_mutexattr\_setprioceiling()* functions are marked  
54756 as part of the Threads and Thread Priority Protection options.54757 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
54758 implementation does not support the Thread Priority Protection option.54759 The [ENOTSUP] error condition has been removed since these functions do not have a *protocol*  
54760 argument.54761 The **restrict** keyword is added to the *pthread\_mutexattr\_getprioceiling()* prototype for alignment  
54762 with the ISO/IEC 9899:1999 standard.54763 **Issue 7**54764 The *pthread\_mutexattr\_getprioceiling()* and *pthread\_mutexattr\_setprioceiling()* functions are moved  
54765 from the Threads option to require support of either the Robust Mutex Priority Protection option  
54766 or the Non-Robust Mutex Priority Protection option.54767 The [EINVAL] error for an uninitialized mutex attributes object is removed; this condition  
54768 results in undefined behavior.

54769 **NAME**

54770 pthread\_mutexattr\_getprotocol, pthread\_mutexattr\_setprotocol — get and set the protocol  
 54771 attribute of the mutex attributes object (**REALTIME THREADS**)

54772 **SYNOPSIS**

```
54773 MC1 #include <pthread.h>
54774
54775 int pthread_mutexattr_getprotocol(const pthread_mutexattr_t
54776     *restrict attr, int *restrict protocol);
54777 int pthread_mutexattr_setprotocol(pthread_mutexattr_t *attr,
54778     int protocol);
```

54778 **DESCRIPTION**

54779 The *pthread\_mutexattr\_getprotocol()* and *pthread\_mutexattr\_setprotocol()* functions, respectively,  
 54780 shall get and set the protocol attribute of a mutex attributes object pointed to by *attr* which was  
 54781 previously created by the function *pthread\_mutexattr\_init()*.

54782 The *protocol* attribute defines the protocol to be followed in utilizing mutexes. The value of  
 54783 *protocol* may be one of:

```
54784 RPI | TPI PTHREAD_PRIO_INHERIT
54785 MC1 PTHREAD_PRIO_NONE
54786 RPP | TPP PTHREAD_PRIO_PROTECT
```

54787 which are defined in the **<pthread.h>** header. The default value of the attribute shall be  
 54788 PTHREAD\_PRIO\_NONE.

54789 When a thread owns a mutex with the PTHREAD\_PRIO\_NONE *protocol* attribute, its priority  
 54790 and scheduling shall not be affected by its mutex ownership.

54791 RPI When a thread is blocking higher priority threads because of owning one or more robust  
 54792 mutexes with the PTHREAD\_PRIO\_INHERIT *protocol* attribute, it shall execute at the higher of  
 54793 its priority or the priority of the highest priority thread waiting on any of the robust mutexes  
 54794 owned by this thread and initialized with this protocol.

54795 TPI When a thread is blocking higher priority threads because of owning one or more non-robust  
 54796 mutexes with the PTHREAD\_PRIO\_INHERIT *protocol* attribute, it shall execute at the higher of  
 54797 its priority or the priority of the highest priority thread waiting on any of the non-robust  
 54798 mutexes owned by this thread and initialized with this protocol.

54799 RPP When a thread owns one or more robust mutexes initialized with the  
 54800 PTHREAD\_PRIO\_PROTECT protocol, it shall execute at the higher of its priority or the highest  
 54801 of the priority ceilings of all the robust mutexes owned by this thread and initialized with this  
 54802 attribute, regardless of whether other threads are blocked on any of these robust mutexes or not.

54803 TPP When a thread owns one or more non-robust mutexes initialized with the  
 54804 PTHREAD\_PRIO\_PROTECT protocol, it shall execute at the higher of its priority or the highest  
 54805 of the priority ceilings of all the non-robust mutexes owned by this thread and initialized with  
 54806 this attribute, regardless of whether other threads are blocked on any of these non-robust  
 54807 mutexes or not.

54808 While a thread is holding a mutex which has been initialized with the  
 54809 PTHREAD\_PRIO\_INHERIT or PTHREAD\_PRIO\_PROTECT protocol attributes, it shall not be  
 54810 subject to being moved to the tail of the scheduling queue at its priority in the event that its  
 54811 original priority is changed, such as by a call to *sched\_setparam()*. Likewise, when a thread

54812 unlocks a mutex that has been initialized with the PTHREAD\_PRIO\_INHERIT or  
 54813 PTHREAD\_PRIO\_PROTECT protocol attributes, it shall not be subject to being moved to the tail  
 54814 of the scheduling queue at its priority in the event that its original priority is changed.

54815 If a thread simultaneously owns several mutexes initialized with different protocols, it shall  
 54816 execute at the highest of the priorities that it would have obtained by each of these protocols.

54817 RPI | TPI When a thread makes a call to *pthread\_mutex\_lock()*, the mutex was initialized with the protocol  
 54818 attribute having the value PTHREAD\_PRIO\_INHERIT, when the calling thread is blocked  
 54819 because the mutex is owned by another thread, that owner thread shall inherit the priority level  
 54820 of the calling thread as long as it continues to own the mutex. The implementation shall update  
 54821 its execution priority to the maximum of its assigned priority and all its inherited priorities.  
 54822 Furthermore, if this owner thread itself becomes blocked on another mutex with the *protocol*  
 54823 attribute having the value PTHREAD\_PRIO\_INHERIT, the same priority inheritance effect shall  
 54824 be propagated to this other owner thread, in a recursive manner.

54825 The behavior is undefined if the value specified by the *attr* argument to  
 54826 *pthread\_mutexattr\_getprotocol()* or *pthread\_mutexattr\_setprotocol()* does not refer to an initialized  
 54827 mutex attributes object.

#### 54828 RETURN VALUE

54829 Upon successful completion, the *pthread\_mutexattr\_getprotocol()* and  
 54830 *pthread\_mutexattr\_setprotocol()* functions shall return zero; otherwise, an error number shall be  
 54831 returned to indicate the error.

#### 54832 ERRORS

54833 The *pthread\_mutexattr\_setprotocol()* function shall fail if:

54834 [ENOTSUP] The value specified by *protocol* is an unsupported value.

54835 The *pthread\_mutexattr\_getprotocol()* and *pthread\_mutexattr\_setprotocol()* functions may fail if:

54836 [EINVAL] The value specified by *protocol* is invalid.

54837 [EPERM] The caller does not have the privilege to perform the operation.

54838 These functions shall not return an error code of [EINTR].

#### 54839 EXAMPLES

54840 None.

#### 54841 APPLICATION USAGE

54842 None.

#### 54843 RATIONALE

54844 If an implementation detects that the value specified by the *attr* argument to  
 54845 *pthread\_mutexattr\_getprotocol()* or *pthread\_mutexattr\_setprotocol()* does not refer to an initialized  
 54846 mutex attributes object, it is recommended that the function should fail and report an [EINVAL]  
 54847 error.

#### 54848 FUTURE DIRECTIONS

54849 None.

#### 54850 SEE ALSO

54851 *pthread\_cond\_destroy()*, *pthread\_create()*, *pthread\_mutex\_destroy()*

54852 XBD <pthread.h>

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

Marked as part of the Realtime Threads Feature Group.

**Issue 6**

The *pthread\_mutexattr\_getprotocol()* and *pthread\_mutexattr\_setprotocol()* functions are marked as part of the Threads option and either the Thread Priority Protection or Thread Priority Inheritance options.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Thread Priority Protection or Thread Priority Inheritance options.

The **restrict** keyword is added to the *pthread\_mutexattr\_getprotocol()* prototype for alignment with the ISO/IEC 9899:1999 standard.

**Issue 7**

SD5-XSH-ERN-135 is applied, updating the DESCRIPTION to define a default value for the *protocol* attribute.

SD5-XSH-ERN-188 is applied, updating the DESCRIPTION.

The *pthread\_mutexattr\_getprotocol()* and *pthread\_mutexattr\_setprotocol()* functions are moved from the Threads option to require support of either the Non-Robust Mutex Priority Protection option or the Non-Robust Mutex Priority Inheritance option or the Robust Mutex Priority Protection option or the Robust Mutex Priority Inheritance option.

The [EINVAL] error for an uninitialized mutex attributes object is removed; this condition results in undefined behavior.

54875 **NAME**

54876 pthread\_mutexattr\_getpshared, pthread\_mutexattr\_setpshared — get and set the process-shared  
 54877 attribute

54878 **SYNOPSIS**

```
54879 TSH #include <pthread.h>
54880
54880 int pthread_mutexattr_getpshared(const pthread_mutexattr_t
54881     *restrict attr, int *restrict pshared);
54882 int pthread_mutexattr_setpshared(pthread_mutexattr_t *attr,
54883     int pshared);
```

54884 **DESCRIPTION**

54885 The *pthread\_mutexattr\_getpshared()* function shall obtain the value of the *process-shared* attribute  
 54886 from the attributes object referenced by *attr*.

54887 The *pthread\_mutexattr\_setpshared()* function shall set the *process-shared* attribute in an initialized  
 54888 attributes object referenced by *attr*.

54889 The *process-shared* attribute is set to `PTHREAD_PROCESS_SHARED` to permit a mutex to be  
 54890 operated upon by any thread that has access to the memory where the mutex is allocated, even if  
 54891 the mutex is allocated in memory that is shared by multiple processes. See [Section 2.9.9](#) (on page  
 54892 523) for further requirements. The default value of the attribute shall be  
 54893 `PTHREAD_PROCESS_PRIVATE`.

54894 The behavior is undefined if the value specified by the *attr* argument to  
 54895 *pthread\_mutexattr\_getpshared()* or *pthread\_mutexattr\_setpshared()* does not refer to an initialized  
 54896 mutex attributes object.

54897 **RETURN VALUE**

54898 Upon successful completion, *pthread\_mutexattr\_setpshared()* shall return zero; otherwise, an error  
 54899 number shall be returned to indicate the error.

54900 Upon successful completion, *pthread\_mutexattr\_getpshared()* shall return zero and store the value  
 54901 of the *process-shared* attribute of *attr* into the object referenced by the *pshared* parameter.  
 54902 Otherwise, an error number shall be returned to indicate the error.

54903 **ERRORS**

54904 The *pthread\_mutexattr\_setpshared()* function may fail if:

54905 [EINVAL] The new value specified for the attribute is outside the range of legal values  
 54906 for that attribute.

54907 These functions shall not return an error code of [EINTR].

54908 **EXAMPLES**

54909 None.

54910 **APPLICATION USAGE**

54911 None.

54912 **RATIONALE**

54913 If an implementation detects that the value specified by the *attr* argument to  
 54914 *pthread\_mutexattr\_getpshared()* or *pthread\_mutexattr\_setpshared()* does not refer to an initialized  
 54915 mutex attributes object, it is recommended that the function should fail and report an [EINVAL]  
 54916 error.

54917 **FUTURE DIRECTIONS**

54918 None.

54919 **SEE ALSO**54920 *pthread\_cond\_destroy()*, *pthread\_create()*, *pthread\_mutex\_destroy()*, *pthread\_mutexattr\_destroy()*

54921 XBD &lt;pthread.h&gt;

54922 **CHANGE HISTORY**

54923 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

54924 **Issue 6**54925 The *pthread\_mutexattr\_getpshared()* and *pthread\_mutexattr\_setpshared()* functions are marked as  
54926 part of the Threads and Thread Process-Shared Synchronization options.54927 The **restrict** keyword is added to the *pthread\_mutexattr\_getpshared()* prototype for alignment  
54928 with the ISO/IEC 9899:1999 standard.54929 **Issue 7**54930 The *pthread\_mutexattr\_getpshared()* and *pthread\_mutexattr\_setpshared()* functions are marked only |  
54931 as part of the Thread Process-Shared Synchronization option as the Threads option is now part |  
54932 of the Base.54933 The [EINVAL] error for an uninitialized mutex attributes object is removed; this condition  
54934 results in undefined behavior.



**NAME**

pthread\_mutexattr\_getrobust, pthread\_mutexattr\_setrobust — get and set the mutex robust attribute

**SYNOPSIS**

```
#include <pthread.h>

int pthread_mutexattr_getrobust(const pthread_mutexattr_t *restrict
    attr, int *restrict robust);
int pthread_mutexattr_setrobust(pthread_mutexattr_t *attr,
    int robust);
```

**DESCRIPTION**

The *pthread\_mutexattr\_getrobust()* and *pthread\_mutexattr\_setrobust()* functions, respectively, shall get and set the mutex *robust* attribute. This attribute is set in the *robust* parameter. Valid values for *robust* include:

**PTHREAD\_MUTEX\_STALLED**

No special actions are taken if the owner of the mutex is terminated while holding the mutex lock. This can lead to deadlocks if no other thread can unlock the mutex.

This is the default value.

**PTHREAD\_MUTEX\_ROBUST**

If the process containing the owning thread of a robust mutex terminates while holding the mutex lock, the next thread that acquires the mutex shall be notified about the termination by the return value [EOWNERDEAD] from the locking function. If the owning thread of a robust mutex terminates while holding the mutex lock, the next thread that attempts to acquire the mutex may be notified about the termination by the return value [EOWNERDEAD]. The notified thread can then attempt to make the state protected by the mutex consistent again, and if successful can mark the mutex state as consistent by calling *pthread\_mutex\_consistent()*. After a subsequent successful call to *pthread\_mutex\_unlock()*, the mutex lock shall be released and can be used normally by other threads. If the mutex is unlocked without a call to *pthread\_mutex\_consistent()*, it shall be in a permanently unusable state and all attempts to lock the mutex shall fail with the error [ENOTRECOVERABLE]. The only permissible operation on such a mutex is *pthread\_mutex\_destroy()*.

The behavior is undefined if the value specified by the *attr* argument to *pthread\_mutexattr\_getrobust()* or *pthread\_mutexattr\_setrobust()* does not refer to an initialized mutex attributes object.

**RETURN VALUE**

Upon successful completion, the *pthread\_mutexattr\_getrobust()* function shall return zero and store the value of the *robust* attribute of *attr* into the object referenced by the *robust* parameter. Otherwise, an error value shall be returned to indicate the error. If successful, the *pthread\_mutexattr\_setrobust()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

**ERRORS**

The *pthread\_mutexattr\_setrobust()* function shall fail if:

[EINVAL] The value of *robust* is invalid.

These functions shall not return an error code of [EINTR].



**EXAMPLES**

None.

**APPLICATION USAGE**

The actions required to make the state protected by the mutex consistent again are solely dependent on the application. If it is not possible to make the state of a mutex consistent, robust mutexes can be used to notify this situation by calling *pthread\_mutex\_unlock()* without a prior call to *pthread\_mutex\_consistent()*.

If the state is declared inconsistent by calling *pthread\_mutex\_unlock()* without a prior call to *pthread\_mutex\_consistent()*, a possible approach could be to destroy the mutex and then reinitialize it. However, it should be noted that this is possible only in certain situations where the state protected by the mutex has to be reinitialized and coordination achieved with other threads blocked on the mutex, because otherwise a call to a locking function with a reference to a mutex object invalidated by a call to *pthread\_mutex\_destroy()* results in undefined behavior.

**RATIONALE**

If an implementation detects that the value specified by the *attr* argument to *pthread\_mutexattr\_getrobust()* or *pthread\_mutexattr\_setrobust()* does not refer to an initialized mutex attributes object, it is recommended that the function should fail and report an [EINVAL] error.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*pthread\_mutex\_consistent()*, *pthread\_mutex\_destroy()*, *pthread\_mutex\_lock()*

XBD <pthread.h>

**CHANGE HISTORY**

First released in Issue 7.

55003 **NAME**

55004 pthread\_mutexattr\_gettype, pthread\_mutexattr\_settype — get and set the mutex type attribute

55005 **SYNOPSIS**

```
55006 #include <pthread.h>
55007 int pthread_mutexattr_gettype(const pthread_mutexattr_t *restrict attr,
55008 int *restrict type);
55009 int pthread_mutexattr_settype(pthread_mutexattr_t *attr, int type);
```

55010 **DESCRIPTION**

55011 The *pthread\_mutexattr\_gettype()* and *pthread\_mutexattr\_settype()* functions, respectively, shall get  
 55012 and set the mutex *type* attribute. This attribute is set in the *type* parameter to these functions. The  
 55013 default value of the *type* attribute is PTHREAD\_MUTEX\_DEFAULT.

55014 The type of mutex is contained in the *type* attribute of the mutex attributes. Valid mutex types  
 55015 include:

```
55016 PTHREAD_MUTEX_NORMAL
55017 PTHREAD_MUTEX_ERRORCHECK
55018 PTHREAD_MUTEX_RECURSIVE
55019 PTHREAD_MUTEX_DEFAULT
```

55020 The mutex type affects the behavior of calls which lock and unlock the mutex. See  
 55021 *pthread\_mutex\_lock()* for details. An implementation may map PTHREAD\_MUTEX\_DEFAULT  
 55022 to one of the other mutex types.

55023 The behavior is undefined if the value specified by the *attr* argument to  
 55024 *pthread\_mutexattr\_gettype()* or *pthread\_mutexattr\_settype()* does not refer to an initialized mutex  
 55025 attributes object.

55026 **RETURN VALUE**

55027 Upon successful completion, the *pthread\_mutexattr\_gettype()* function shall return zero and store  
 55028 the value of the *type* attribute of *attr* into the object referenced by the *type* parameter. Otherwise,  
 55029 an error shall be returned to indicate the error.

55030 If successful, the *pthread\_mutexattr\_settype()* function shall return zero; otherwise, an error  
 55031 number shall be returned to indicate the error.

55032 **ERRORS**

55033 The *pthread\_mutexattr\_settype()* function shall fail if:

55034 [EINVAL] The value *type* is invalid.

55035 These functions shall not return an error code of [EINTR].

55036 **EXAMPLES**

55037 None.

55038 **APPLICATION USAGE**

55039 It is advised that an application should not use a PTHREAD\_MUTEX\_RECURSIVE mutex with  
 55040 condition variables because the implicit unlock performed for a *pthread\_cond\_timedwait()* or  
 55041 *pthread\_cond\_wait()* may not actually release the mutex (if it had been locked multiple times). If  
 55042 this happens, no other thread can satisfy the condition of the predicate.

55043 **RATIONALE**

55044 If an implementation detects that the value specified by the *attr* argument to  
 55045 *pthread\_mutexattr\_gettype()* or *pthread\_mutexattr\_settype()* does not refer to an initialized mutex  
 55046 attributes object, it is recommended that the function should fail and report an [EINVAL] error.

55047 **FUTURE DIRECTIONS**

55048 None.

55049 **SEE ALSO**55050 *pthread\_cond\_timedwait()*, *pthread\_mutex\_lock()*55051 XBD **<pthread.h>**55052 **CHANGE HISTORY**

55053 First released in Issue 5.

55054 **Issue 6**

55055 The Open Group Corrigendum U033/3 is applied. The SYNOPSIS for  
55056 *pthread\_mutexattr\_gettype()* is updated so that the first argument is of type **const**  
55057 **pthread\_mutexattr\_t** \*.

55058 The **restrict** keyword is added to the *pthread\_mutexattr\_gettype()* prototype for alignment with  
55059 the ISO/IEC 9899:1999 standard.

55060 **Issue 7**

55061 The *pthread\_mutexattr\_gettype()* and *pthread\_mutexattr\_settype()* functions are moved from the  
55062 XSI option to the Base.

55063 The [EINVAL] error for an uninitialized mutex attributes object is removed; this condition  
55064 results in undefined behavior.

55065 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0464 [121] is applied.

55066 **NAME**

55067 pthread\_mutexattr\_init — initialize the mutex attributes object

55068 **SYNOPSIS**

55069 #include &lt;pthread.h&gt;

55070 int pthread\_mutexattr\_init(pthread\_mutexattr\_t \*attr);

55071 **DESCRIPTION**55072 Refer to *pthread\_mutexattr\_destroy()*.

55073 **NAME**

55074 pthread\_mutexattr\_setprioceiling — set the prioceiling attribute of the mutex attributes object  
55075 (**REALTIME THREADS**)

55076 **SYNOPSIS**

```
55077 RPP|TPP #include <pthread.h>  
55078 int pthread_mutexattr_setprioceiling(pthread_mutexattr_t *attr,  
55079 int prioceiling);
```

55080 **DESCRIPTION**

55081 Refer to *pthread\_mutexattr\_getprioceiling()*.

55082 **NAME**

55083 pthread\_mutexattr\_setprotocol — set the protocol attribute of the mutex attributes object  
55084 (**REALTIME THREADS**)

55085 **SYNOPSIS**

```
55086 MC1 #include <pthread.h>  
55087 int pthread_mutexattr_setprotocol(pthread_mutexattr_t *attr,  
55088     int protocol);
```

55089 **DESCRIPTION**

55090 Refer to *pthread\_mutexattr\_getprotocol()*.

55091 **NAME**

55092 pthread\_mutexattr\_setpshared — set the process-shared attribute

55093 **SYNOPSIS**

```
55094 TSH      #include <pthread.h>
55095          int pthread_mutexattr_setpshared(pthread_mutexattr_t *attr,
55096          int pshared);
```

55097 **DESCRIPTION**55098 Refer to *pthread\_mutexattr\_getpshared()*.

55099 **NAME**

55100 pthread\_mutexattr\_setrobust — get and set the mutex robust attribute

55101 **SYNOPSIS**

55102 #include &lt;pthread.h&gt;

55103 int pthread\_mutexattr\_setrobust(pthread\_mutexattr\_t \*attr,

55104 int robust);

55105 **DESCRIPTION**55106 Refer to *pthread\_mutexattr\_getrobust()*.



55107 **NAME**

55108 pthread\_mutexattr\_settype — set the mutex type attribute

55109 **SYNOPSIS**

55110 #include &lt;pthread.h&gt;

55111 int pthread\_mutexattr\_settype(pthread\_mutexattr\_t \*attr, int type);

55112 **DESCRIPTION**55113 Refer to *pthread\_mutexattr\_gettype()*.

## 55114 NAME

55115 pthread\_once — dynamic package initialization

## 55116 SYNOPSIS

```
55117 #include <pthread.h>
55118 int pthread_once(pthread_once_t *once_control,
55119                 void (*init_routine)(void));
55120 pthread_once_t once_control = PTHREAD_ONCE_INIT;
```

## 55121 DESCRIPTION

55122 The first call to *pthread\_once()* by any thread in a process, with a given *once\_control*, shall call the  
 55123 *init\_routine* with no arguments. Subsequent calls of *pthread\_once()* with the same *once\_control*  
 55124 shall not call the *init\_routine*. On return from *pthread\_once()*, *init\_routine* shall have completed.  
 55125 The *once\_control* parameter shall determine whether the associated initialization routine has been  
 55126 called.

55127 The *pthread\_once()* function is not a cancellation point. However, if *init\_routine* is a cancellation  
 55128 point and is canceled, the effect on *once\_control* shall be as if *pthread\_once()* was never called.

55129 If the call to *init\_routine* is terminated by a call to *longjmp()*, *\_longjmp()*, or *siglongjmp()*, the +  
 55130 behavior is undefined. +

55131 The constant PTHREAD\_ONCE\_INIT is defined in the **<pthread.h>** header.

55132 The behavior of *pthread\_once()* is undefined if *once\_control* has automatic storage duration or is  
 55133 not initialized by PTHREAD\_ONCE\_INIT.

## 55134 RETURN VALUE

55135 Upon successful completion, *pthread\_once()* shall return zero; otherwise, an error number shall  
 55136 be returned to indicate the error.

## 55137 ERRORS

55138 The *pthread\_once()* function shall not return an error code of [EINTR].

## 55139 EXAMPLES

55140 None.

## 55141 APPLICATION USAGE

55142 If *init\_routine* recursively calls *pthread\_once()* with the same *once\_control*, the recursive call will |  
 55143 not call the specified *init\_routine*, and thus the specified *init\_routine* will not complete, and thus |  
 55144 the recursive call to *pthread\_once()* will not return. Use of *longjmp()*, *\_longjmp()*, or *siglongjmp()* |  
 55145 within an *init\_routine* to jump to a point outside of *init\_routine* prevents *init\_routine* from |  
 55146 returning.

## 55147 RATIONALE

55148 Some C libraries are designed for dynamic initialization. That is, the global initialization for the  
 55149 library is performed when the first procedure in the library is called. In a single-threaded  
 55150 program, this is normally implemented using a static variable whose value is checked on entry  
 55151 to a routine, as follows:

```
55152 static int random_is_initialized = 0;
55153 extern void initialize_random();
55154 int random_function()
55155 {
55156     if (random_is_initialized == 0) {
55157         initialize_random();
55158         random_is_initialized = 1;
```

```

55159     }
55160     ... /* Operations performed after initialization. */
55161 }

```

55162 To keep the same structure in a multi-threaded program, a new primitive is needed. Otherwise,  
 55163 library initialization has to be accomplished by an explicit call to a library-exported initialization  
 55164 function prior to any use of the library.

55165 For dynamic library initialization in a multi-threaded process, if an initialization flag is used the  
 55166 flag needs to be protected against modification by multiple threads simultaneously calling into  
 55167 the library. This can be done by using a mutex (initialized by assigning  
 55168 PTHREAD\_MUTEX\_INITIALIZER). However, the better solution is to use *pthread\_once()* which  
 55169 is designed for exactly this purpose, as follows:

```

55170 #include <pthread.h>
55171 static pthread_once_t random_is_initialized = PTHREAD_ONCE_INIT;
55172 extern void initialize_random();
55173
55174 int random_function()
55175 {
55176     (void) pthread_once(&random_is_initialized, initialize_random);
55177     ... /* Operations performed after initialization. */
55178 }

```

55178 If an implementation detects that the value specified by the *once\_control* argument to  
 55179 *pthread\_once()* does not refer to a **pthread\_once\_t** object initialized by PTHREAD\_ONCE\_INIT,  
 55180 it is recommended that the function should fail and report an [EINVAL] error.

#### 55181 FUTURE DIRECTIONS

55182 None.

#### 55183 SEE ALSO

55184 XBD [<pthread.h>](#)

#### 55185 CHANGE HISTORY

55186 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

#### 55187 Issue 6

55188 The *pthread\_once()* function is marked as part of the Threads option.

55189 The [EINVAL] error is added as a “may fail” case for if either argument is invalid.

#### 55190 Issue 7

55191 The *pthread\_once()* function is moved from the Threads option to the Base.

55192 The [EINVAL] error for an uninitialized **pthread\_once\_t** object is removed; this condition results  
 55193 in undefined behavior.

## 55194 NAME

55195 pthread\_rwlock\_destroy, pthread\_rwlock\_init — destroy and initialize a read-write lock object

## 55196 SYNOPSIS

```
55197 #include <pthread.h>
55198 int pthread_rwlock_destroy(pthread_rwlock_t *rwlock);
55199 int pthread_rwlock_init(pthread_rwlock_t *restrict rwlock,
55200     const pthread_rwlockattr_t *restrict attr);
55201 pthread_rwlock_t rwlock = PTHREAD_RWLOCK_INITIALIZER;
```

## 55202 DESCRIPTION

55203 The *pthread\_rwlock\_destroy()* function shall destroy the read-write lock object referenced by  
 55204 *rwlock* and release any resources used by the lock. The effect of subsequent use of the lock is  
 55205 undefined until the lock is reinitialized by another call to *pthread\_rwlock\_init()*. An  
 55206 implementation may cause *pthread\_rwlock\_destroy()* to set the object referenced by *rwlock* to an  
 55207 invalid value. Results are undefined if *pthread\_rwlock\_destroy()* is called when any thread holds  
 55208 *rwlock*. Attempting to destroy an uninitialized read-write lock results in undefined behavior.

55209 The *pthread\_rwlock\_init()* function shall allocate any resources required to use the read-write lock  
 55210 referenced by *rwlock* and initializes the lock to an unlocked state with attributes referenced by  
 55211 *attr*. If *attr* is NULL, the default read-write lock attributes shall be used; the effect is the same as  
 55212 passing the address of a default read-write lock attributes object. Once initialized, the lock can be  
 55213 used any number of times without being reinitialized. Results are undefined if  
 55214 *pthread\_rwlock\_init()* is called specifying an already initialized read-write lock. Results are  
 55215 undefined if a read-write lock is used without first being initialized.

55216 If the *pthread\_rwlock\_init()* function fails, *rwlock* shall not be initialized and the contents of *rwlock*  
 55217 are undefined.

55218 See [Section 2.9.9](#) (on page 523) for further requirements. |

55219 In cases where default read-write lock attributes are appropriate, the macro  
 55220 PTHREAD\_RWLOCK\_INITIALIZER can be used to initialize read-write locks. The effect shall  
 55221 be equivalent to dynamic initialization by a call to *pthread\_rwlock\_init()* with the *attr* parameter  
 55222 specified as NULL, except that no error checks are performed.

55223 The behavior is undefined if the value specified by the *attr* argument to *pthread\_rwlock\_init()*  
 55224 does not refer to an initialized read-write lock attributes object.

## 55225 RETURN VALUE

55226 If successful, the *pthread\_rwlock\_destroy()* and *pthread\_rwlock\_init()* functions shall return zero;  
 55227 otherwise, an error number shall be returned to indicate the error.

## 55228 ERRORS

55229 The *pthread\_rwlock\_init()* function shall fail if:

55230 [EAGAIN] The system lacked the necessary resources (other than memory) to initialize  
 55231 another read-write lock.

55232 [ENOMEM] Insufficient memory exists to initialize the read-write lock.

55233 [EPERM] The caller does not have the privilege to perform the operation.

55234 These functions shall not return an error code of [EINTR].

**EXAMPLES**

None.

**APPLICATION USAGE**

Applications using these and related read-write lock functions may be subject to priority inversion, as discussed in XBD [Section 3.291](#) (on page 80).

**RATIONALE**

If an implementation detects that the value specified by the *rwlock* argument to *pthread\_rwlock\_destroy()* does not refer to an initialized read-write lock object, it is recommended that the function should fail and report an [EINVAL] error.

If an implementation detects that the value specified by the *attr* argument to *pthread\_rwlock\_init()* does not refer to an initialized read-write lock attributes object, it is recommended that the function should fail and report an [EINVAL] error.

If an implementation detects that the value specified by the *rwlock* argument to *pthread\_rwlock\_destroy()* or *pthread\_rwlock\_init()* refers to a locked read-write lock object, or detects that the value specified by the *rwlock* argument to *pthread\_rwlock\_init()* refers to an already initialized read-write lock object, it is recommended that the function should fail and report an [EBUSY] error.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[pthread\\_rwlock\\_rdlock\(\)](#), [pthread\\_rwlock\\_timedrdlock\(\)](#), [pthread\\_rwlock\\_timedwrlock\(\)](#),  
[pthread\\_rwlock\\_trywrlock\(\)](#), [pthread\\_rwlock\\_unlock\(\)](#)

XBD [Section 3.291](#) (on page 80), [<pthread.h>](#)

**CHANGE HISTORY**

First released in Issue 5.

**Issue 6**

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is now part of the Threads option (previously it was part of the Read-Write Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension). The initializer macro is also deleted from the SYNOPSIS.

- The DESCRIPTION is updated as follows:

- It explicitly notes allocation of resources upon initialization of a read-write lock object.

- A paragraph is added specifying that copies of read-write lock objects may not be used.

- An [EINVAL] error is added to the ERRORS section for *pthread\_rwlock\_init()*, indicating that the *rwlock* value is invalid.

- The SEE ALSO section is updated.

The **restrict** keyword is added to the *pthread\_rwlock\_init()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/45 is applied, adding APPLICATION USAGE relating to priority inversion.

55278 **Issue 7**

55279 Austin Group Interpretation 1003.1-2001 #048 is applied, adding the  
55280 PTHREAD\_RWLOCK\_INITIALIZER macro.

55281 The *pthread\_rwlock\_destroy()* and *pthread\_rwlock\_init()* functions are moved from the Threads  
55282 option to the Base.

55283 The [EINVAL] error for an uninitialized read-write lock object or read-write lock attributes  
55284 object is removed; this condition results in undefined behavior.

55285 The [EBUSY] error for a locked read-write lock object or an already initialized read-write lock  
55286 object is removed; this condition results in undefined behavior.

55287 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0465 [70] is applied.

**55288 NAME**

55289 pthread\_rwlock\_rdlock, pthread\_rwlock\_tryrdlock — lock a read-write lock object for reading

**55290 SYNOPSIS**

```
55291 #include <pthread.h>
55292 int pthread_rwlock_rdlock(pthread_rwlock_t *rwlock);
55293 int pthread_rwlock_tryrdlock(pthread_rwlock_t *rwlock);
```

**55294 DESCRIPTION**

55295 The *pthread\_rwlock\_rdlock()* function shall apply a read lock to the read-write lock referenced by  
 55296 *rwlock*. The calling thread acquires the read lock if a writer does not hold the lock and there are  
 55297 no writers blocked on the lock.

55298 TPS If the Thread Execution Scheduling option is supported, and the threads involved in the lock are  
 55299 executing with the scheduling policies SCHED\_FIFO or SCHED\_RR, the calling thread shall not  
 55300 acquire the lock if a writer holds the lock or if writers of higher or equal priority are blocked on  
 55301 the lock; otherwise, the calling thread shall acquire the lock.

55302 TPS TSP If the Thread Execution Scheduling option is supported, and the threads involved in the lock are  
 55303 executing with the SCHED\_SPORADIC scheduling policy, the calling thread shall not acquire  
 55304 the lock if a writer holds the lock or if writers of higher or equal priority are blocked on the lock;  
 55305 otherwise, the calling thread shall acquire the lock.

55306 If the Thread Execution Scheduling option is not supported, it is implementation-defined  
 55307 whether the calling thread acquires the lock when a writer does not hold the lock and there are  
 55308 writers blocked on the lock. If a writer holds the lock, the calling thread shall not acquire the  
 55309 read lock. If the read lock is not acquired, the calling thread shall block until it can acquire the  
 55310 lock. The calling thread may deadlock if at the time the call is made it holds a write lock.

55311 A thread may hold multiple concurrent read locks on *rwlock* (that is, successfully call the  
 55312 *pthread\_rwlock\_rdlock()* function *n* times). If so, the application shall ensure that the thread  
 55313 performs matching unlocks (that is, it calls the *pthread\_rwlock\_unlock()* function *n* times).

55314 The maximum number of simultaneous read locks that an implementation guarantees can be  
 55315 applied to a read-write lock shall be implementation-defined. The *pthread\_rwlock\_rdlock()*  
 55316 function may fail if this maximum would be exceeded.

55317 The *pthread\_rwlock\_tryrdlock()* function shall apply a read lock as in the *pthread\_rwlock\_rdlock()*  
 55318 function, with the exception that the function shall fail if the equivalent *pthread\_rwlock\_rdlock()*  
 55319 call would have blocked the calling thread. In no case shall the *pthread\_rwlock\_tryrdlock()*  
 55320 function ever block; it always either acquires the lock or fails and returns immediately.

55321 Results are undefined if any of these functions are called with an uninitialized read-write lock.

55322 If a signal is delivered to a thread waiting for a read-write lock for reading, upon return from the  
 55323 signal handler the thread resumes waiting for the read-write lock for reading as if it was not  
 55324 interrupted.

**55325 RETURN VALUE**

55326 If successful, the *pthread\_rwlock\_rdlock()* function shall return zero; otherwise, an error number  
 55327 shall be returned to indicate the error.

55328 The *pthread\_rwlock\_tryrdlock()* function shall return zero if the lock for reading on the read-write  
 55329 lock object referenced by *rwlock* is acquired. Otherwise, an error number shall be returned to  
 55330 indicate the error.

## ERRORS

The *pthread\_rwlock\_tryrdlock()* function shall fail if:

[EBUSY]           The read-write lock could not be acquired for reading because a writer holds the lock or a writer with the appropriate priority was blocked on it.

The *pthread\_rwlock\_rdlock()* and *pthread\_rwlock\_tryrdlock()* functions may fail if:

[EAGAIN]           The read lock could not be acquired because the maximum number of read locks for *rwlock* has been exceeded.

The *pthread\_rwlock\_rdlock()* function may fail if:

[EDEADLK]           A deadlock condition was detected or the current thread already owns the read-write lock for writing.

These functions shall not return an error code of [EINTR].

## EXAMPLES

None.

## APPLICATION USAGE

Applications using these functions may be subject to priority inversion, as discussed in XBD [Section 3.291](#) (on page 80).

## RATIONALE

If an implementation detects that the value specified by the *rwlock* argument to *pthread\_rwlock\_rdlock()* or *pthread\_rwlock\_tryrdlock()* does not refer to an initialized read-write lock object, it is recommended that the function should fail and report an [EINVAL] error.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[\*pthread\\_rwlock\\_destroy\(\)\*](#), [\*pthread\\_rwlock\\_timedrdlock\(\)\*](#), [\*pthread\\_rwlock\\_timedwrlock\(\)\*](#), [\*pthread\\_rwlock\\_trywrlock\(\)\*](#), [\*pthread\\_rwlock\\_unlock\(\)\*](#)

XBD [Section 3.291](#) (on page 80), [Section 4.12](#) (on page 111), [`<pthread.h>`](#)

## CHANGE HISTORY

First released in Issue 5.

## Issue 6

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is now part of the Threads option (previously it was part of the Read-Write Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension).
- The DESCRIPTION is updated as follows:
  - Conditions under which writers have precedence over readers are specified.
  - Failure of *pthread\_rwlock\_tryrdlock()* is clarified.
  - A paragraph on the maximum number of read locks is added.
- In the ERRORS sections, [EBUSY] is modified to take into account write priority, and [EDEADLK] is deleted as a *pthread\_rwlock\_tryrdlock()* error.



- 55370                   • The SEE ALSO section is updated.
- 55371                   IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/101 is applied, updating the ERRORS
- 55372                   section so that the [EDEADLK] error includes detection of a deadlock condition.
- 55373   **Issue 7**
- 55374                   The *pthread\_rwlock\_rdlock()* and *pthread\_rwlock\_tryrdlock()* functions are moved from the Threads
- 55375                   option to the Base.
- 55376                   The [EINVAL] error for an uninitialized read-write lock object is removed; this condition results
- 55377                   in undefined behavior.

55378 **NAME**

55379 pthread\_rwlock\_timedrdlock — lock a read-write lock for reading

55380 **SYNOPSIS**

```
55381 #include <pthread.h>
55382 #include <time.h>

55383 int pthread_rwlock_timedrdlock(pthread_rwlock_t *restrict rwlock,
55384                               const struct timespec *restrict abstime);
```

55385 **DESCRIPTION**

55386 The *pthread\_rwlock\_timedrdlock()* function shall apply a read lock to the read-write lock  
 55387 referenced by *rwlock* as in the *pthread\_rwlock\_rdlock()* function. However, if the lock cannot be  
 55388 acquired without waiting for other threads to unlock the lock, this wait shall be terminated  
 55389 when the specified timeout expires. The timeout shall expire when the absolute time specified  
 55390 by *abstime* passes, as measured by the clock on which timeouts are based (that is, when the value  
 55391 of that clock equals or exceeds *abstime*), or if the absolute time specified by *abstime* has already  
 55392 been passed at the time of the call.

55393 The timeout shall be based on the CLOCK\_REALTIME clock. The resolution of the timeout shall  
 55394 be the resolution of the CLOCK\_REALTIME clock. The **timespec** data type is defined in the  
 55395 **<time.h>** header. Under no circumstances shall the function fail with a timeout if the lock can be  
 55396 acquired immediately. The validity of the *abstime* parameter need not be checked if the lock can  
 55397 be immediately acquired.

55398 If a signal that causes a signal handler to be executed is delivered to a thread blocked on a read-  
 55399 write lock via a call to *pthread\_rwlock\_timedrdlock()*, upon return from the signal handler the  
 55400 thread shall resume waiting for the lock as if it was not interrupted.

55401 The calling thread may deadlock if at the time the call is made it holds a write lock on *rwlock*.  
 55402 The results are undefined if this function is called with an uninitialized read-write lock.

55403 **RETURN VALUE**

55404 The *pthread\_rwlock\_timedrdlock()* function shall return zero if the lock for reading on the read-  
 55405 write lock object referenced by *rwlock* is acquired. Otherwise, an error number shall be returned  
 55406 to indicate the error.

55407 **ERRORS**

55408 The *pthread\_rwlock\_timedrdlock()* function shall fail if:

55409 [ETIMEDOUT] The lock could not be acquired before the specified timeout expired.

55410 The *pthread\_rwlock\_timedrdlock()* function may fail if:

55411 [EAGAIN] The read lock could not be acquired because the maximum number of read  
 55412 locks for lock would be exceeded.

55413 [EDEADLK] A deadlock condition was detected or the calling thread already holds a write  
 55414 lock on *rwlock*.

55415 [EINVAL] The *abstime* nanosecond value is less than zero or greater than or equal to 1 000  
 55416 million.

55417 This function shall not return an error code of [EINTR].

**EXAMPLES**

None.

**APPLICATION USAGE**

Applications using this function may be subject to priority inversion, as discussed in XBD Section 3.291 (on page 80).

**RATIONALE**

If an implementation detects that the value specified by the *rwlock* argument to *pthread\_rwlock\_timedrdlock()* does not refer to an initialized read-write lock object, it is recommended that the function should fail and report an [EINVAL] error.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*pthread\_rwlock\_destroy()*, *pthread\_rwlock\_rdlock()*, *pthread\_rwlock\_timedwrlock()*, *pthread\_rwlock\_trywrlock()*, *pthread\_rwlock\_unlock()*

XBD Section 3.291 (on page 80), Section 4.12 (on page 111), **<pthread.h>**, **<time.h>**

**CHANGE HISTORY**

First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/102 is applied, updating the ERRORS section so that the [EDEADLK] error includes detection of a deadlock condition.

**Issue 7**

The *pthread\_rwlock\_timedrdlock()* function is moved from the Timeouts option to the Base.

The [EINVAL] error for an uninitialized read-write lock object is removed; this condition results in undefined behavior.

**NAME**

pthread\_rwlock\_timedwrlock — lock a read-write lock for writing

**SYNOPSIS**

```
#include <pthread.h>
#include <time.h>

int pthread_rwlock_timedwrlock(pthread_rwlock_t *restrict rwlock,
                               const struct timespec *restrict abstime);
```

**DESCRIPTION**

The *pthread\_rwlock\_timedwrlock()* function shall apply a write lock to the read-write lock referenced by *rwlock* as in the *pthread\_rwlock\_wrlock()* function. However, if the lock cannot be acquired without waiting for other threads to unlock the lock, this wait shall be terminated when the specified timeout expires. The timeout shall expire when the absolute time specified by *abstime* passes, as measured by the clock on which timeouts are based (that is, when the value of that clock equals or exceeds *abstime*), or if the absolute time specified by *abstime* has already been passed at the time of the call.

The timeout shall be based on the CLOCK\_REALTIME clock. The resolution of the timeout shall be the resolution of the CLOCK\_REALTIME clock. The **timespec** data type is defined in the **<time.h>** header. Under no circumstances shall the function fail with a timeout if the lock can be acquired immediately. The validity of the *abstime* parameter need not be checked if the lock can be immediately acquired.

If a signal that causes a signal handler to be executed is delivered to a thread blocked on a read-write lock via a call to *pthread\_rwlock\_timedwrlock()*, upon return from the signal handler the thread shall resume waiting for the lock as if it was not interrupted.

The calling thread may deadlock if at the time the call is made it holds the read-write lock. The results are undefined if this function is called with an uninitialized read-write lock.

**RETURN VALUE**

The *pthread\_rwlock\_timedwrlock()* function shall return zero if the lock for writing on the read-write lock object referenced by *rwlock* is acquired. Otherwise, an error number shall be returned to indicate the error.

**ERRORS**

The *pthread\_rwlock\_timedwrlock()* function shall fail if:

[ETIMEDOUT] The lock could not be acquired before the specified timeout expired.

The *pthread\_rwlock\_timedwrlock()* function may fail if:

[EDEADLK] A deadlock condition was detected or the calling thread already holds the *rwlock*.

[EINVAL] The *abstime* nanosecond value is less than zero or greater than or equal to 1 000 million.

This function shall not return an error code of [EINTR].

**55479 EXAMPLES**

55480 None.

**55481 APPLICATION USAGE**

55482 Applications using this function may be subject to priority inversion, as discussed in XBD  
55483 [Section 3.291](#) (on page 80).

**55484 RATIONALE**

55485 If an implementation detects that the value specified by the *rwlock* argument to  
55486 *pthread\_rwlock\_timedwrlock()* does not refer to an initialized read-write lock object, it is  
55487 recommended that the function should fail and report an [EINVAL] error.

**55488 FUTURE DIRECTIONS**

55489 None.

**55490 SEE ALSO**

55491 [pthread\\_rwlock\\_destroy\(\)](#), [pthread\\_rwlock\\_rdlock\(\)](#), [pthread\\_rwlock\\_timedrdlock\(\)](#),  
55492 [pthread\\_rwlock\\_trywrlock\(\)](#), [pthread\\_rwlock\\_unlock\(\)](#)

55493 XBD [Section 3.291](#) (on page 80), [Section 4.12](#) (on page 111), [<pthread.h>](#), [<time.h>](#)

**55494 CHANGE HISTORY**

55495 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

55496 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/103 is applied, updating the ERRORS  
55497 section so that the [EDEADLK] error includes detection of a deadlock condition.

**55498 Issue 7**

55499 The *pthread\_rwlock\_timedwrlock()* function is moved from the Timeouts option to the Base.

55500 The [EINVAL] error for an uninitialized read-write lock object is removed; this condition results  
55501 in undefined behavior.

# pthread\_rwlock\_tryrdlock()

System Interfaces

## 55502 NAME

55503 pthread\_rwlock\_tryrdlock — lock a read-write lock object for reading

## 55504 SYNOPSIS

55505 #include <pthread.h>

55506 int pthread\_rwlock\_tryrdlock(pthread\_rwlock\_t \*rwlock);

## 55507 DESCRIPTION

55508 Refer to *pthread\_rwlock\_rdlock()*.

**55509 NAME**

55510 pthread\_rwlock\_trywrlock, pthread\_rwlock\_wrlock — lock a read-write lock object for writing

**55511 SYNOPSIS**

```
55512 #include <pthread.h>
55513 int pthread_rwlock_trywrlock(pthread_rwlock_t *rwlock);
55514 int pthread_rwlock_wrlock(pthread_rwlock_t *rwlock);
```

**55515 DESCRIPTION**

55516 The *pthread\_rwlock\_trywrlock()* function shall apply a write lock like the *pthread\_rwlock\_wrlock()*  
 55517 function, with the exception that the function shall fail if any thread currently holds *rwlock* (for  
 55518 reading or writing).

55519 The *pthread\_rwlock\_wrlock()* function shall apply a write lock to the read-write lock referenced by  
 55520 *rwlock*. The calling thread shall acquire the write lock if no thread (reader or writer) holds the  
 55521 read-write lock *rwlock*. Otherwise, if another thread holds the read-write lock *rwlock*, the calling  
 55522 thread shall block until it can acquire the lock. If a deadlock condition occurs or the calling  
 55523 thread already owns the read-write lock for writing or reading, the call shall either deadlock or  
 55524 return [EDEADLK].

55525 Results are undefined if any of these functions are called with an uninitialized read-write lock.

55526 If a signal is delivered to a thread waiting for a read-write lock for writing, upon return from the  
 55527 signal handler the thread resumes waiting for the read-write lock for writing as if it was not  
 55528 interrupted.

**55529 RETURN VALUE**

55530 The *pthread\_rwlock\_trywrlock()* function shall return zero if the lock for writing on the read-write  
 55531 lock object referenced by *rwlock* is acquired. Otherwise, an error number shall be returned to  
 55532 indicate the error.

55533 If successful, the *pthread\_rwlock\_wrlock()* function shall return zero; otherwise, an error number  
 55534 shall be returned to indicate the error.

**55535 ERRORS**

55536 The *pthread\_rwlock\_trywrlock()* function shall fail if:

55537 [EBUSY] The read-write lock could not be acquired for writing because it was already  
 55538 locked for reading or writing.

55539 The *pthread\_rwlock\_wrlock()* function may fail if:

55540 [EDEADLK] A deadlock condition was detected or the current thread already owns the  
 55541 read-write lock for writing or reading.

55542 These functions shall not return an error code of [EINTR].

**55543 EXAMPLES**

55544 None.

**55545 APPLICATION USAGE**

55546 Applications using these functions may be subject to priority inversion, as discussed in XBD  
 55547 [Section 3.291](#) (on page 80).

**55548 RATIONALE**

55549 If an implementation detects that the value specified by the *rwlock* argument to  
 55550 *pthread\_rwlock\_trywrlock()* or *pthread\_rwlock\_wrlock()* does not refer to an initialized read-write  
 55551 lock object, it is recommended that the function should fail and report an [EINVAL] error.

55552 **FUTURE DIRECTIONS**

55553 None.

55554 **SEE ALSO**

55555 *pthread\_rwlock\_destroy()*, *pthread\_rwlock\_rdlock()*, *pthread\_rwlock\_timedrdlock()*,  
55556 *pthread\_rwlock\_timedwrlock()*, *pthread\_rwlock\_unlock()*

55557 XBD [Section 3.291](#) (on page 80), [Section 4.12](#) (on page 111), [<pthread.h>](#)

55558 **CHANGE HISTORY**

55559 First released in Issue 5.

55560 **Issue 6**

55561 The following changes are made for alignment with IEEE Std 1003.1j-2000:

- 55562 • The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is  
55563 now part of the Threads option (previously it was part of the Read-Write Locks option in  
55564 IEEE Std 1003.1j-2000 and also part of the XSI extension).
- 55565 • The [EDEADLK] error is deleted as a *pthread\_rwlock\_trywrlock()* error.
- 55566 • The SEE ALSO section is updated.

55567 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/104 is applied, updating the ERRORS  
55568 section so that the [EDEADLK] error includes detection of a deadlock condition.

55569 **Issue 7**

55570 The *pthread\_rwlock\_trywrlock()* and *pthread\_rwlock\_wrlock()* functions are moved from the  
55571 Threads option to the Base.

55572 The [EINVAL] error for an uninitialized read-write lock object is removed; this condition results  
55573 in undefined behavior.



55574 **NAME**

55575 pthread\_rwlock\_unlock — unlock a read-write lock object

55576 **SYNOPSIS**

55577 #include &lt;pthread.h&gt;

55578 int pthread\_rwlock\_unlock(pthread\_rwlock\_t \*rwlock);

55579 **DESCRIPTION**

55580 The *pthread\_rwlock\_unlock()* function shall release a lock held on the read-write lock object  
55581 referenced by *rwlock*. Results are undefined if the read-write lock *rwlock* is not held by the  
55582 calling thread.

55583 If this function is called to release a read lock from the read-write lock object and there are other  
55584 read locks currently held on this read-write lock object, the read-write lock object remains in the  
55585 read locked state. If this function releases the last read lock for this read-write lock object, the  
55586 read-write lock object shall be put in the unlocked state with no owners.

55587 If this function is called to release a write lock for this read-write lock object, the read-write lock  
55588 object shall be put in the unlocked state.

55589 If there are threads blocked on the lock when it becomes available, the scheduling policy shall  
55590 TPS determine which thread(s) shall acquire the lock. If the Thread Execution Scheduling option is  
55591 supported, when threads executing with the scheduling policies SCHED\_FIFO, SCHED\_RR, or  
55592 SCHED\_SPORADIC are waiting on the lock, they shall acquire the lock in priority order when  
55593 the lock becomes available. For equal priority threads, write locks shall take precedence over  
55594 read locks. If the Thread Execution Scheduling option is not supported, it is implementation-  
55595 defined whether write locks take precedence over read locks.

55596 Results are undefined if this function is called with an uninitialized read-write lock.

55597 **RETURN VALUE**

55598 If successful, the *pthread\_rwlock\_unlock()* function shall return zero; otherwise, an error number  
55599 shall be returned to indicate the error.

55600 **ERRORS**

55601 The *pthread\_rwlock\_unlock()* function shall not return an error code of [EINTR].

55602 **EXAMPLES**

55603 None.

55604 **APPLICATION USAGE**

55605 None.

55606 **RATIONALE**

55607 If an implementation detects that the value specified by the *rwlock* argument to  
55608 *pthread\_rwlock\_unlock()* does not refer to an initialized read-write lock object, it is recommended  
55609 that the function should fail and report an [EINVAL] error.

55610 If an implementation detects that the value specified by the *rwlock* argument to  
55611 *pthread\_rwlock\_unlock()* refers to a read-write lock object for which the current thread does not  
55612 hold a lock, it is recommended that the function should fail and report an [EPERM] error.

55613 **FUTURE DIRECTIONS**

55614 None.

**SEE ALSO**

*pthread\_rwlock\_destroy(), pthread\_rwlock\_rdlock(), pthread\_rwlock\_timedrdlock(),  
pthread\_rwlock\_timedwrlock(), pthread\_rwlock\_trywrlock()*

XBD Section 4.12 (on page 111), **<pthread.h>**

**CHANGE HISTORY**

First released in Issue 5.

**Issue 6**

The following changes are made for alignment with IEEE Std 1003.1j-2000:

- The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is now part of the Threads option (previously it was part of the Read-Write Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension).
- The DESCRIPTION is updated as follows:
  - The conditions under which writers have precedence over readers are specified.
  - The concept of read-write lock owner is deleted.
- The SEE ALSO section is updated.

**Issue 7**

SD5-XSH-ERN-183 is applied.

The *pthread\_rwlock\_unlock()* function is moved from the Threads option to the Base.

The [EINVAL] error for an uninitialized read-write lock object is removed; this condition results in undefined behavior.

The [EPERM] error for a read-write lock object for which the current thread does not hold a lock is removed; this condition results in undefined behavior.

55637 **NAME**

55638 pthread\_rwlock\_wrlock — lock a read-write lock object for writing

55639 **SYNOPSIS**

55640 #include &lt;pthread.h&gt;

55641 int pthread\_rwlock\_wrlock(pthread\_rwlock\_t \*rwlock);

55642 **DESCRIPTION**55643 Refer to *pthread\_rwlock\_trywrlock()*.

## 55644 NAME

55645 pthread\_rwlockattr\_destroy, pthread\_rwlockattr\_init — destroy and initialize the read-write  
55646 lock attributes object

## 55647 SYNOPSIS

```
55648 #include <pthread.h>
55649 int pthread_rwlockattr_destroy(pthread_rwlockattr_t *attr);
55650 int pthread_rwlockattr_init(pthread_rwlockattr_t *attr);
```

## 55651 DESCRIPTION

55652 The *pthread\_rwlockattr\_destroy()* function shall destroy a read-write lock attributes object. A  
55653 destroyed *attr* attributes object can be reinitialized using *pthread\_rwlockattr\_init()*; the results of  
55654 otherwise referencing the object after it has been destroyed are undefined. An implementation  
55655 may cause *pthread\_rwlockattr\_destroy()* to set the object referenced by *attr* to an invalid value.

55656 The *pthread\_rwlockattr\_init()* function shall initialize a read-write lock attributes object *attr* with  
55657 the default value for all of the attributes defined by the implementation.

55658 Results are undefined if *pthread\_rwlockattr\_init()* is called specifying an already initialized *attr*  
55659 attributes object.

55660 After a read-write lock attributes object has been used to initialize one or more read-write locks,  
55661 any function affecting the attributes object (including destruction) shall not affect any previously  
55662 initialized read-write locks.

55663 The behavior is undefined if the value specified by the *attr* argument to  
55664 *pthread\_rwlockattr\_destroy()* does not refer to an initialized read-write lock attributes object.

## 55665 RETURN VALUE

55666 If successful, the *pthread\_rwlockattr\_destroy()* and *pthread\_rwlockattr\_init()* functions shall return  
55667 zero; otherwise, an error number shall be returned to indicate the error.

## 55668 ERRORS

55669 The *pthread\_rwlockattr\_init()* function shall fail if:

55670 [ENOMEM] Insufficient memory exists to initialize the read-write lock attributes object.

55671 These functions shall not return an error code of [EINTR].

## 55672 EXAMPLES

55673 None.

## 55674 APPLICATION USAGE

55675 None.

## 55676 RATIONALE

55677 If an implementation detects that the value specified by the *attr* argument to  
55678 *pthread\_rwlockattr\_destroy()* does not refer to an initialized read-write lock attributes object, it is  
55679 recommended that the function should fail and report an [EINVAL] error.

## 55680 FUTURE DIRECTIONS

55681 None.

## 55682 SEE ALSO

55683 *pthread\_rwlock\_destroy()*, *pthread\_rwlockattr\_getpshared()*

55684 XBD <pthread.h>

**CHANGE HISTORY**

55685 First released in Issue 5.

**Issue 6**

55688 The following changes are made for alignment with IEEE Std 1003.1j-2000:

- 55689 • The margin code in the SYNOPSIS is changed to THR to indicate that the functionality is
- 55690 now part of the Threads option (previously it was part of the Read-Write Locks option in
- 55691 IEEE Std 1003.1j-2000 and also part of the XSI extension).
- 55692 • The SEE ALSO section is updated.

**Issue 7**

55693 The *pthread\_rwlockattr\_destroy()* and *pthread\_rwlockattr\_init()* functions are moved from the

55694 Threads option to the Base.

55695

55696 The [EINVAL] error for an uninitialized read-write lock attributes object is removed; this

55697 condition results in undefined behavior.

## NAME

pthread\_rwlockattr\_getpshared, pthread\_rwlockattr\_setpshared — get and set the process-shared attribute of the read-write lock attributes object

## SYNOPSIS

```
TSH #include <pthread.h>

int pthread_rwlockattr_getpshared(const pthread_rwlockattr_t
    *restrict attr, int *restrict pshared);
int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *attr,
    int pshared);
```

## DESCRIPTION

The *pthread\_rwlockattr\_getpshared()* function shall obtain the value of the *process-shared* attribute from the initialized attributes object referenced by *attr*. The *pthread\_rwlockattr\_setpshared()* function shall set the *process-shared* attribute in an initialized attributes object referenced by *attr*.

The *process-shared* attribute shall be set to `PTHREAD_PROCESS_SHARED` to permit a read-write lock to be operated upon by any thread that has access to the memory where the read-write lock is allocated, even if the read-write lock is allocated in memory that is shared by multiple processes. See [Section 2.9.9](#) (on page 523) for further requirements. The default value of the *process-shared* attribute shall be `PTHREAD_PROCESS_PRIVATE`.

Additional attributes, their default values, and the names of the associated functions to get and set those attribute values are implementation-defined.

The behavior is undefined if the value specified by the *attr* argument to *pthread\_rwlockattr\_getpshared()* or *pthread\_rwlockattr\_setpshared()* does not refer to an initialized read-write lock attributes object.

## RETURN VALUE

Upon successful completion, the *pthread\_rwlockattr\_getpshared()* function shall return zero and store the value of the *process-shared* attribute of *attr* into the object referenced by the *pshared* parameter. Otherwise, an error number shall be returned to indicate the error.

If successful, the *pthread\_rwlockattr\_setpshared()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

## ERRORS

The *pthread\_rwlockattr\_setpshared()* function may fail if:

[EINVAL] The new value specified for the attribute is outside the range of legal values for that attribute.

These functions shall not return an error code of [EINTR].

## EXAMPLES

None.

## APPLICATION USAGE

None.

## RATIONALE

None.

**55738 FUTURE DIRECTIONS**

55739 None.

**55740 SEE ALSO**

55741 *pthread\_rwlock\_destroy()*, *pthread\_rwlockattr\_destroy()*

55742 XBD <pthread.h>

**55743 CHANGE HISTORY**

55744 First released in Issue 5.

**55745 Issue 6**

55746 The following changes are made for alignment with IEEE Std 1003.1j-2000:

- 55747 • The margin code in the SYNOPSIS is changed to THR TSH to indicate that the  
55748 functionality is now part of the Threads option (previously it was part of the Read-Write  
55749 Locks option in IEEE Std 1003.1j-2000 and also part of the XSI extension).
- 55750 • The DESCRIPTION notes that additional attributes are implementation-defined.
- 55751 • The SEE ALSO section is updated.

55752 The **restrict** keyword is added to the *pthread\_rwlockattr\_getpshared()* prototype for alignment  
55753 with the ISO/IEC 9899:1999 standard.

**55754 Issue 7**

55755 The *pthread\_rwlockattr\_getpshared()* and *pthread\_rwlockattr\_setpshared()* functions are marked |  
55756 only as part of the Thread Process-Shared Synchronization option as the Threads option is now |  
55757 part of the Base.

55758 The [EINVAL] error for an uninitialized read-write lock attributes object is removed; this  
55759 condition results in undefined behavior.

55760 **NAME**

55761 pthread\_rwlockattr\_init — initialize the read-write lock attributes object

55762 **SYNOPSIS**

55763 #include &lt;pthread.h&gt;

55764 int pthread\_rwlockattr\_init(pthread\_rwlockattr\_t \*attr);

55765 **DESCRIPTION**55766 Refer to *pthread\_rwlockattr\_destroy()*.



55767 **NAME**

55768 pthread\_rwlockattr\_setpshared — set the process-shared attribute of the read-write lock  
55769 attributes object

55770 **SYNOPSIS**

```
55771 TSH    #include <pthread.h>
55772        int pthread_rwlockattr_setpshared(pthread_rwlockattr_t *attr,
55773        int pshared);
```

55774 **DESCRIPTION**

55775 Refer to *pthread\_rwlockattr\_getpshared()*.

55776 **NAME**

55777 pthread\_self — get the calling thread ID

55778 **SYNOPSIS**

55779 #include <pthread.h>

55780 pthread\_t pthread\_self(void);

55781 **DESCRIPTION**

55782 The *pthread\_self()* function shall return the thread ID of the calling thread.

55783 **RETURN VALUE**

55784 The *pthread\_self()* function shall always be successful and no return value is reserved to indicate  
55785 an error.

55786 **ERRORS**

55787 No errors are defined.

55788 **EXAMPLES**

55789 None.

55790 **APPLICATION USAGE**

55791 None.

55792 **RATIONALE**

55793 The *pthread\_self()* function provides a capability similar to the *getpid()* function for processes  
55794 and the rationale is the same: the creation call does not provide the thread ID to the created  
55795 thread.

55796 **FUTURE DIRECTIONS**

55797 None.

55798 **SEE ALSO**

55799 *pthread\_create()*, *pthread\_equal()*

55800 XBD <pthread.h>

55801 **CHANGE HISTORY**

55802 First released in Issue 5. Included for alignment with the POSIX Threads Extension.

55803 **Issue 6**

55804 The *pthread\_self()* function is marked as part of the Threads option.

55805 **Issue 7**

55806 Austin Group Interpretation 1003.1-2001 #063 is applied, updating the RETURN VALUE section.

55807 The *pthread\_self()* function is moved from the Threads option to the Base.

**55808 NAME**

55809 pthread\_setcancelstate, pthread\_setcanceltype, pthread\_testcancel — set cancelability state

**55810 SYNOPSIS**

```
55811 #include <pthread.h>
55812 int pthread_setcancelstate(int state, int *oldstate);
55813 int pthread_setcanceltype(int type, int *oldtype);
55814 void pthread_testcancel(void);
```

**55815 DESCRIPTION**

55816 The *pthread\_setcancelstate()* function shall atomically both set the calling thread's cancelability  
 55817 state to the indicated *state* and return the previous cancelability state at the location referenced  
 55818 by *oldstate*. Legal values for *state* are PTHREAD\_CANCEL\_ENABLE and  
 55819 PTHREAD\_CANCEL\_DISABLE.

55820 The *pthread\_setcanceltype()* function shall atomically both set the calling thread's cancelability  
 55821 type to the indicated *type* and return the previous cancelability type at the location referenced by  
 55822 *oldtype*. Legal values for *type* are PTHREAD\_CANCEL\_DEFERRED and  
 55823 PTHREAD\_CANCEL\_ASYNCHRONOUS.

55824 The cancelability state and type of any newly created threads, including the thread in which  
 55825 *main()* was first invoked, shall be PTHREAD\_CANCEL\_ENABLE and  
 55826 PTHREAD\_CANCEL\_DEFERRED respectively.

55827 The *pthread\_testcancel()* function shall create a cancellation point in the calling thread. The  
 55828 *pthread\_testcancel()* function shall have no effect if cancelability is disabled.

**55829 RETURN VALUE**

55830 If successful, the *pthread\_setcancelstate()* and *pthread\_setcanceltype()* functions shall return zero;  
 55831 otherwise, an error number shall be returned to indicate the error.

**55832 ERRORS**

55833 The *pthread\_setcancelstate()* function may fail if:

55834 [EINVAL] The specified state is not PTHREAD\_CANCEL\_ENABLE or  
 55835 PTHREAD\_CANCEL\_DISABLE.

55836 The *pthread\_setcanceltype()* function may fail if:

55837 [EINVAL] The specified type is not PTHREAD\_CANCEL\_DEFERRED or  
 55838 PTHREAD\_CANCEL\_ASYNCHRONOUS.

55839 These functions shall not return an error code of [EINTR].

**55840 EXAMPLES**

55841 None.

**55842 APPLICATION USAGE**

55843 In order to write a signal handler for an asynchronous signal which can run safely in a  
 55844 cancellable thread, *pthread\_setcancelstate()* must be used to disable cancellation for the duration  
 55845 of any calls that the signal handler makes which are cancellation points. However, the standard  
 55846 does not permit strictly conforming applications to call *pthread\_setcancelstate()* from a signal  
 55847 handler since it is not currently required to be async-signal-safe. On implementations where  
 55848 *pthread\_setcancelstate()* is not async-signal-safe, alternatives are to ensure either that the  
 55849 corresponding signals are blocked during execution of functions that are not async-cancel-safe  
 55850 or that cancellation is disabled during times when those signals could be delivered.  
 55851 Implementations are strongly encouraged to make *pthread\_setcancelstate()* async-signal-safe.

**RATIONALE**

The *pthread\_setcancelstate()* and *pthread\_setcanceltype()* functions control the points at which a thread may be asynchronously canceled. For cancellation control to be usable in modular fashion, some rules need to be followed.

An object can be considered to be a generalization of a procedure. It is a set of procedures and global variables written as a unit and called by clients not known by the object. Objects may depend on other objects.

First, cancelability should only be disabled on entry to an object, never explicitly enabled. On exit from an object, the cancelability state should always be restored to its value on entry to the object.

This follows from a modularity argument: if the client of an object (or the client of an object that uses that object) has disabled cancelability, it is because the client does not want to be concerned about cleaning up if the thread is canceled while executing some sequence of actions. If an object is called in such a state and it enables cancelability and a cancellation request is pending for that thread, then the thread is canceled, contrary to the wish of the client that disabled.

Second, the cancelability type may be explicitly set to either *deferred* or *asynchronous* upon entry to an object. But as with the cancelability state, on exit from an object the cancelability type should always be restored to its value on entry to the object.

Finally, only functions that are cancel-safe may be called from a thread that is asynchronously cancelable.

**FUTURE DIRECTIONS**

The *pthread\_setcancelstate()* function may be added to the table of async-signal-safe functions in [Section 2.4.3](#) (on page 490).

**SEE ALSO**

[\*pthread\\_cancel\(\)\*](#)

XBD [\*<pthread.h>\*](#)

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Threads Extension.

**Issue 6**

The *pthread\_setcancelstate()*, *pthread\_setcanceltype()*, and *pthread\_testcancel()* functions are marked as part of the Threads option.

**Issue 7**

The *pthread\_setcancelstate()*, *pthread\_setcanceltype()*, and *pthread\_testcancel()* functions are moved from the Threads option to the Base.

55886 **NAME**

55887 pthread\_setconcurrency — set the level of concurrency

55888 **SYNOPSIS**

```
55889 OB XSI #include <pthread.h>  
55890 int pthread_setconcurrency(int new_level);
```

55891 **DESCRIPTION**55892 Refer to *pthread\_getconcurrency()*.

55893 **NAME**

55894 pthread\_setschedparam — dynamic thread scheduling parameters access (**REALTIME**  
55895 **THREADS**)

55896 **SYNOPSIS**

```
55897 TPS    #include <pthread.h>
55898
55898     int pthread_setschedparam(pthread_t thread, int policy,
55899                               const struct sched_param *param);
```

55900 **DESCRIPTION**

55901 Refer to *pthread\_getschedparam()*.

**55902 NAME**

55903 pthread\_setschedprio — dynamic thread scheduling parameters access (REALTIME  
55904 THREADS)

**55905 SYNOPSIS**

```
55906 TPS #include <pthread.h>  
55907 int pthread_setschedprio(pthread_t thread, int prio);
```

**55908 DESCRIPTION**

55909 The *pthread\_setschedprio()* function shall set the scheduling priority for the thread whose thread  
55910 ID is given by *thread* to the value given by *prio*. See [Scheduling Policies](#) (on page 506) for a  
55911 description on how this function call affects the ordering of the thread in the thread list for its  
55912 new priority.

55913 If the *pthread\_setschedprio()* function fails, the scheduling priority of the target thread shall not be  
55914 changed.

**55915 RETURN VALUE**

55916 If successful, the *pthread\_setschedprio()* function shall return zero; otherwise, an error number  
55917 shall be returned to indicate the error.

**55918 ERRORS**

55919 The *pthread\_setschedprio()* function may fail if:

55920 [EINVAL] The value of *prio* is invalid for the scheduling policy of the specified thread.

55921 [EPERM] The caller does not have appropriate privileges to set the scheduling priority  
55922 of the specified thread.

55923 The *pthread\_setschedprio()* function shall not return an error code of [EINTR].

**55924 EXAMPLES**

55925 None.

**55926 APPLICATION USAGE**

55927 None.

**55928 RATIONALE**

55929 The *pthread\_setschedprio()* function provides a way for an application to temporarily raise its  
55930 priority and then lower it again, without having the undesired side-effect of yielding to other  
55931 threads of the same priority. This is necessary if the application is to implement its own  
55932 strategies for bounding priority inversion, such as priority inheritance or priority ceilings. This  
55933 capability is especially important if the implementation does not support the Thread Priority  
55934 Protection or Thread Priority Inheritance options, but even if those options are supported it is  
55935 needed if the application is to bound priority inheritance for other resources, such as  
55936 semaphores.

55937 The standard developers considered that while it might be preferable conceptually to solve this  
55938 problem by modifying the specification of *pthread\_setschedparam()*, it was too late to make such a  
55939 change, as there may be implementations that would need to be changed. Therefore, this new  
55940 function was introduced.

55941 If an implementation detects use of a thread ID after the end of its lifetime, it is recommended  
55942 that the function should fail and report an [ESRCH] error.

55943 **FUTURE DIRECTIONS**

55944 None.

55945 **SEE ALSO**

55946 [Scheduling Policies](#) (on page 506), [pthread\\_getschedparam\(\)](#)

55947 XBD [<pthread.h>](#)

55948 **CHANGE HISTORY**

55949 First released in Issue 6. Included as a response to IEEE PASC Interpretation 1003.1 #96.

55950 **Issue 7**

55951 The `pthread_setschedprio()` function is marked only as part of the Thread Execution Scheduling  
55952 option as the Threads option is now part of the Base.

55953 Austin Group Interpretation 1003.1-2001 #069 is applied, updating the [EPERM] error.

55954 Austin Group Interpretation 1003.1-2001 #142 is applied, removing the [ESRCH] error condition.

55955 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0466 [314] is applied.



55956 **NAME**

55957 pthread\_setspecific — thread-specific data management

55958 **SYNOPSIS**

55959 #include &lt;pthread.h&gt;

55960 int pthread\_setspecific(pthread\_key\_t key, const void \*value);

55961 **DESCRIPTION**55962 Refer to *pthread\_getspecific()*.

55963 **NAME**

55964 pthread\_sigmask, sigprocmask — examine and change blocked signals

55965 **SYNOPSIS**

```

55966 CX      #include <signal.h>
55967
55967      int pthread_sigmask(int how, const sigset_t *restrict set,
55968                          sigset_t *restrict oset);
55969
55969      int sigprocmask(int how, const sigset_t *restrict set,
55970                      sigset_t *restrict oset);

```

55971 **DESCRIPTION**

55972 The *pthread\_sigmask()* function shall examine or change (or both) the calling thread's signal  
 55973 mask, regardless of the number of threads in the process. The function shall be equivalent to  
 55974 *sigprocmask()*, without the restriction that the call be made in a single-threaded process.

55975 In a single-threaded process, the *sigprocmask()* function shall examine or change (or both) the  
 55976 signal mask of the calling thread.

55977 If the argument *set* is not a null pointer, it points to a set of signals to be used to change the  
 55978 currently blocked set.

55979 The argument *how* indicates the way in which the set is changed, and the application shall  
 55980 ensure it consists of one of the following values:

55981 SIG\_BLOCK      The resulting set shall be the union of the current set and the signal set  
 55982 pointed to by *set*.

55983 SIG\_SETMASK    The resulting set shall be the signal set pointed to by *set*.

55984 SIG\_UNBLOCK    The resulting set shall be the intersection of the current set and the  
 55985 complement of the signal set pointed to by *set*.

55986 If the argument *oset* is not a null pointer, the previous mask shall be stored in the location  
 55987 pointed to by *oset*. If *set* is a null pointer, the value of the argument *how* is not significant and the  
 55988 thread's signal mask shall be unchanged; thus the call can be used to enquire about currently  
 55989 blocked signals.

55990 If there are any pending unblocked signals after the call to *sigprocmask()*, at least one of those  
 55991 signals shall be delivered before the call to *sigprocmask()* returns.

55992 It is not possible to block those signals which cannot be ignored. This shall be enforced by the  
 55993 system without causing an error to be indicated.

55994 If any of the SIGFPE, SIGILL, SIGSEGV, or SIGBUS signals are generated while they are blocked,  
 55995 the result is undefined, unless the signal was generated by the action of another process, or by  
 55996 one of the functions *kill()*, *pthread\_kill()*, *raise()*, or *sigqueue()*.

55997 If *sigprocmask()* fails, the thread's signal mask shall not be changed.

55998 The use of the *sigprocmask()* function is unspecified in a multi-threaded process.

55999 **RETURN VALUE**

56000 Upon successful completion *pthread\_sigmask()* shall return 0; otherwise, it shall return the  
 56001 corresponding error number.

56002 Upon successful completion, *sigprocmask()* shall return 0; otherwise, -1 shall be returned, *errno*  
 56003 shall be set to indicate the error, and the signal mask of the process shall be unchanged.

**ERRORS**

The *pthread\_sigmask()* and *sigprocmask()* functions shall fail if:

[EINVAL] The value of the *how* argument is not equal to one of the defined values.

The *pthread\_sigmask()* function shall not return an error code of [EINTR].

**EXAMPLES****Signaling in a Multi-Threaded Process**

This example shows the use of *pthread\_sigmask()* in order to deal with signals in a multi-threaded process. It provides a fairly general framework that could be easily adapted/extended.

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <signal.h>
#include <string.h>
#include <errno.h>
...

static sigset_t  signal_mask; /* signals to block          */

int main (int argc, char *argv[])
{
    pthread_t  sig_thr_id;      /* signal handler thread ID */
    int        rc;              /* return code               */

    sigemptyset (&signal_mask);
    sigaddset (&signal_mask, SIGINT);
    sigaddset (&signal_mask, SIGTERM);
    rc = pthread_sigmask (SIG_BLOCK, &signal_mask, NULL);
    if (rc != 0) {
        /* handle error */
        ...
    }
    /* any newly created threads inherit the signal mask */

    rc = pthread_create (&sig_thr_id, NULL, signal_thread, NULL);
    if (rc != 0) {
        /* handle error */
        ...
    }

    /* APPLICATION CODE */
    ...
}

void *signal_thread (void *arg)
{
    int        sig_caught;      /* signal caught             */
    int        rc;              /* returned code             */

    rc = sigwait (&signal_mask, &sig_caught);
    if (rc != 0) {
        /* handle error */
    }
}
```

```

56049         switch (sig_caught)
56050         {
56051             case SIGINT:      /* process SIGINT */
56052                 ...
56053                 break;
56054             case SIGTERM:     /* process SIGTERM */
56055                 ...
56056                 break;
56057             default:          /* should normally not happen */
56058                 fprintf (stderr, "\nUnexpected signal %d\n", sig_caught);
56059                 break;
56060         }
56061     }

```

## APPLICATION USAGE

None.

## RATIONALE

When a thread's signal mask is changed in a signal-catching function that is installed by *sigaction()*, the restoration of the signal mask on return from the signal-catching function overrides that change (see *sigaction()*). If the signal-catching function was installed with *signal()*, it is unspecified whether this occurs.

See *kill()* for a discussion of the requirement on delivery of signals.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*exec*, *kill()*, *sigaction()*, *sigaddset()*, *sigdelset()*, *sigemptyset()*, *sigfillset()*, *sigismember()*, *sigpending()*, *sigqueue()*, *sigsuspend()*

XBD <signal.h>

## CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

### Issue 5

The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

The *pthread\_sigmask()* function is added for alignment with the POSIX Threads Extension.

### Issue 6

The *pthread\_sigmask()* function is marked as part of the Threads option.

The SYNOPSIS for *sigprocmask()* is marked as a CX extension to note that the presence of this function in the <signal.h> header is an extension to the ISO C standard.

The following changes are made for alignment with the ISO POSIX-1:1996 standard:

- The DESCRIPTION is updated to explicitly state the functions which may generate the signal.

The normative text is updated to avoid use of the term “must” for application requirements.

The **restrict** keyword is added to the *pthread\_sigmask()* and *sigprocmask()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/105 is applied, updating “process’ signal mask” to “thread’s signal mask” in the DESCRIPTION and RATIONALE sections.

56093 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/106 is applied, adding the example to the  
56094 EXAMPLES section.

56095 **Issue 7**

56096 The *pthread\_sigmask()* function is moved from the Threads option to the Base.

56097 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0467 [319] is applied.

56098 **NAME**

56099 pthread\_spin\_destroy, pthread\_spin\_init — destroy or initialize a spin lock object

56100 **SYNOPSIS**

```
56101 #include <pthread.h>
56102 int pthread_spin_destroy(pthread_spinlock_t *lock);
56103 int pthread_spin_init(pthread_spinlock_t *lock, int pshared);
```

56104 **DESCRIPTION**

56105 The *pthread\_spin\_destroy()* function shall destroy the spin lock referenced by *lock* and release any  
 56106 resources used by the lock. The effect of subsequent use of the lock is undefined until the lock is  
 56107 reinitialized by another call to *pthread\_spin\_init()*. The results are undefined if  
 56108 *pthread\_spin\_destroy()* is called when a thread holds the lock, or if this function is called with an  
 56109 uninitialized thread spin lock.

56110 The *pthread\_spin\_init()* function shall allocate any resources required to use the spin lock  
 56111 referenced by *lock* and initialize the lock to an unlocked state.

56112 TSH If the Thread Process-Shared Synchronization option is supported and the value of *pshared* is  
 56113 PTHREAD\_PROCESS\_SHARED, the implementation shall permit the spin lock to be operated  
 56114 upon by any thread that has access to the memory where the spin lock is allocated, even if it is  
 56115 allocated in memory that is shared by multiple processes.

56116 See [Section 2.9.9](#) (on page 523) for further requirements.

56117 The results are undefined if *pthread\_spin\_init()* is called specifying an already initialized spin  
 56118 lock. The results are undefined if a spin lock is used without first being initialized.

56119 If the *pthread\_spin\_init()* function fails, the lock is not initialized and the contents of *lock* are  
 56120 undefined.

56121 Only the object referenced by *lock* may be used for performing synchronization.

56122 The result of referring to copies of that object in calls to *pthread\_spin\_destroy()*,  
 56123 *pthread\_spin\_lock()*, *pthread\_spin\_trylock()*, or *pthread\_spin\_unlock()* is undefined.

56124 **RETURN VALUE**

56125 Upon successful completion, these functions shall return zero; otherwise, an error number shall  
 56126 be returned to indicate the error.

56127 **ERRORS**

56128 The *pthread\_spin\_init()* function shall fail if:

56129 [EAGAIN] The system lacks the necessary resources to initialize another spin lock.

56130 [ENOMEM] Insufficient memory exists to initialize the lock.

56131 These functions shall not return an error code of [EINTR].

56132 **EXAMPLES**

56133 None.

56134 **APPLICATION USAGE**

56135 None.

56136 **RATIONALE**

56137 If an implementation detects that the value specified by the *lock* argument to  
 56138 *pthread\_spin\_destroy()* does not refer to an initialized spin lock object, it is recommended that the  
 56139 function should fail and report an [EINVAL] error.

56140 If an implementation detects that the value specified by the *lock* argument to

56141 *pthread\_spin\_destroy()* or *pthread\_spin\_init()* refers to a locked spin lock object, or detects that the  
56142 value specified by the *lock* argument to *pthread\_spin\_init()* refers to an already initialized spin  
56143 lock object, it is recommended that the function should fail and report an [EBUSY] error.

#### 56144 FUTURE DIRECTIONS

56145 None.

#### 56146 SEE ALSO

56147 *pthread\_spin\_lock()*, *pthread\_spin\_unlock()*

56148 XBD <pthread.h>

#### 56149 CHANGE HISTORY

56150 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

56151 In the SYNOPSIS, the inclusion of <sys/types.h> is no longer required.

#### 56152 Issue 7

56153 The *pthread\_spin\_destroy()* and *pthread\_spin\_init()* functions are moved from the Spin Locks  
56154 option to the Base.

56155 The [EINVAL] error for an uninitialized spin lock object is removed; this condition results in  
56156 undefined behavior.

56157 The [EBUSY] error for a locked spin lock object or an already initialized spin lock object is  
56158 removed; this condition results in undefined behavior.

## 56159 NAME

56160 pthread\_spin\_lock, pthread\_spin\_trylock — lock a spin lock object

## 56161 SYNOPSIS

```
56162 #include <pthread.h>
56163 int pthread_spin_lock(pthread_spinlock_t *lock);
56164 int pthread_spin_trylock(pthread_spinlock_t *lock);
```

## 56165 DESCRIPTION

56166 The *pthread\_spin\_lock()* function shall lock the spin lock referenced by *lock*. The calling thread  
56167 shall acquire the lock if it is not held by another thread. Otherwise, the thread shall spin (that is,  
56168 shall not return from the *pthread\_spin\_lock()* call) until the lock becomes available. The results are  
56169 undefined if the calling thread holds the lock at the time the call is made. The  
56170 *pthread\_spin\_trylock()* function shall lock the spin lock referenced by *lock* if it is not held by any  
56171 thread. Otherwise, the function shall fail.

56172 The results are undefined if any of these functions is called with an uninitialized spin lock.

## 56173 RETURN VALUE

56174 Upon successful completion, these functions shall return zero; otherwise, an error number shall  
56175 be returned to indicate the error.

## 56176 ERRORS

56177 The *pthread\_spin\_lock()* function may fail if:

56178 [EDEADLK] A deadlock condition was detected.

56179 The *pthread\_spin\_trylock()* function shall fail if:

56180 [EBUSY] A thread currently holds the lock.

56181 These functions shall not return an error code of [EINTR].

## 56182 EXAMPLES

56183 None.

## 56184 APPLICATION USAGE

56185 Applications using this function may be subject to priority inversion, as discussed in XBD  
56186 [Section 3.291](#) (on page 80).

## 56187 RATIONALE

56188 If an implementation detects that the value specified by the *lock* argument to *pthread\_spin\_lock()*  
56189 or *pthread\_spin\_trylock()* does not refer to an initialized spin lock object, it is recommended that  
56190 the function should fail and report an [EINVAL] error.

56191 If an implementation detects that the value specified by the *lock* argument to *pthread\_spin\_lock()*  
56192 refers to a spin lock object for which the calling thread already holds the lock, it is recommended  
56193 that the function should fail and report an [EDEADLK] error.

## 56194 FUTURE DIRECTIONS

56195 None.

## 56196 SEE ALSO

56197 [pthread\\_spin\\_destroy\(\)](#), [pthread\\_spin\\_unlock\(\)](#)

56198 XBD [Section 3.291](#) (on page 80), [Section 4.12](#) (on page 111), [<pthread.h>](#)



**CHANGE HISTORY**

- 56199 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.
- 56200
- 56201 In the SYNOPSIS, the inclusion of **<sys/types.h>** is no longer required.
- 56202 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/107 is applied, updating the ERRORS
- 56203 section so that the [EDEADLK] error includes detection of a deadlock condition.
- 56204 **Issue 7**
- 56205 The *pthread\_spin\_lock()* and *pthread\_spin\_trylock()* functions are moved from the Spin Locks
- 56206 option to the Base.
- 56207 The [EINVAL] error for an uninitialized spin lock object is removed; this condition results in
- 56208 undefined behavior.
- 56209 The [EDEADLK] error for a spin lock object for which the calling thread already holds the lock is
- 56210 removed; this condition results in undefined behavior.

56211 **NAME**

56212 pthread\_spin\_unlock — unlock a spin lock object

56213 **SYNOPSIS**

56214 #include <pthread.h>

56215 int pthread\_spin\_unlock(pthread\_spinlock\_t \*lock);

56216 **DESCRIPTION**

56217 The *pthread\_spin\_unlock()* function shall release the spin lock referenced by *lock* which was  
56218 locked via the *pthread\_spin\_lock()* or *pthread\_spin\_trylock()* functions.

56219 The results are undefined if the lock is not held by the calling thread.

56220 If there are threads spinning on the lock when *pthread\_spin\_unlock()* is called, the lock becomes  
56221 available and an unspecified spinning thread shall acquire the lock.

56222 The results are undefined if this function is called with an uninitialized thread spin lock.

56223 **RETURN VALUE**

56224 Upon successful completion, the *pthread\_spin\_unlock()* function shall return zero; otherwise, an  
56225 error number shall be returned to indicate the error.

56226 **ERRORS**

56227 This function shall not return an error code of [EINTR].

56228 **EXAMPLES**

56229 None.

56230 **APPLICATION USAGE**

56231 None.

56232 **RATIONALE**

56233 If an implementation detects that the value specified by the *lock* argument to  
56234 *pthread\_spin\_unlock()* does not refer to an initialized spin lock object, it is recommended that the  
56235 function should fail and report an [EINVAL] error.

56236 If an implementation detects that the value specified by the *lock* argument to  
56237 *pthread\_spin\_unlock()* refers to a spin lock object for which the current thread does not hold the  
56238 lock, it is recommended that the function should fail and report an [EPERM] error.

56239 **FUTURE DIRECTIONS**

56240 None.

56241 **SEE ALSO**

56242 *pthread\_spin\_destroy()*, *pthread\_spin\_lock()*

56243 XBD Section 4.12 (on page 111), <pthread.h>

56244 **CHANGE HISTORY**

56245 First released in Issue 6. Derived from IEEE Std 1003.1j-2000.

56246 In the SYNOPSIS, the inclusion of <sys/types.h> is no longer required.

56247 **Issue 7**

56248 The *pthread\_spin\_unlock()* function is moved from the Spin Locks option to the Base.

56249 The [EINVAL] error for an uninitialized spin lock object is removed; this condition results in  
56250 undefined behavior.

56251  
56252

The [EPERM] error for a spin lock object for which the current thread does not hold the lock is removed; this condition results in undefined behavior.

56253 **NAME**

56254 pthread\_testcancel — set cancelability state

56255 **SYNOPSIS**

56256 #include &lt;pthread.h&gt;

56257 void pthread\_testcancel(void);

56258 **DESCRIPTION**56259 Refer to *pthread\_setcancelstate()*.

56260 **NAME**

56261 ptsname — get name of the slave pseudo-terminal device

56262 **SYNOPSIS**

```
56263 XSI      #include <stdlib.h>
56264         char *ptsname(int fildes);
```

56265 **DESCRIPTION**

56266 The *ptsname()* function shall return the name of the slave pseudo-terminal device associated  
 56267 with a master pseudo-terminal device. The *fildes* argument is a file descriptor that refers to the  
 56268 master device. The *ptsname()* function shall return a pointer to a string containing the pathname  
 56269 of the corresponding slave device.

56270 The *ptsname()* function need not be thread-safe.

56271 **RETURN VALUE**

56272 Upon successful completion, *ptsname()* shall return a pointer to a string which is the name of the  
 56273 pseudo-terminal slave device. Upon failure, *ptsname()* shall return a null pointer and may set  
 56274 *errno*. This could occur if *fildes* is an invalid file descriptor or if the slave device name does not  
 56275 exist in the file system.

56276 The application shall not modify the string returned. The returned pointer might be invalidated  
 56277 or the string content might be overwritten by a subsequent call to *ptsname()*. The returned  
 56278 pointer and the string content might also be invalidated if the calling thread is terminated. +

56279 **ERRORS**

56280 The *ptsname()* function may fail if:

|       |          |                                                                                     |  |
|-------|----------|-------------------------------------------------------------------------------------|--|
| 56281 | [EBADF]  | The <i>fildes</i> argument is not a valid file descriptor.                          |  |
| 56282 | [ENOTTY] | The file associated with the <i>fildes</i> argument is not a master pseudo-terminal |  |
| 56283 |          | device.                                                                             |  |

56284 **EXAMPLES**

56285 None.

56286 **APPLICATION USAGE**

56287 None.

56288 **RATIONALE**

56289 See the RATIONALE section for *posix\_openpt()*.

56290 **FUTURE DIRECTIONS**

56291 None.

56292 **SEE ALSO**

56293 *grantpt()*, *open()*, *posix\_openpt()*, *ttynam()*, *unlockpt()*

56294 XBD <stdlib.h>

56295 **CHANGE HISTORY**

56296 First released in Issue 4, Version 2.

56297 **Issue 5**

56298 Moved from X/OPEN UNIX extension to BASE.

56299 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

56300 **Issue 7**

56301 Austin Group Interpretation 1003.1-2001 #156 is applied.

56302 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0468 [75] and XSH/TC1-2008/0469  
56303 [96] are applied.

56304 **NAME**

56305       putc — put a byte on a stream

56306 **SYNOPSIS**

56307       #include &lt;stdio.h&gt;

56308       int putc(int *c*, FILE \**stream*);56309 **DESCRIPTION**

56310 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 56311 conflict between the requirements described here and the ISO C standard is unintentional. This  
 56312 volume of POSIX.1-2008 defers to the ISO C standard.

56313       The *putc()* function shall be equivalent to *fputc()*, except that if it is implemented as a macro it  
 56314 may evaluate *stream* more than once, so the argument should never be an expression with side-  
 56315 effects.

56316 **RETURN VALUE**56317       Refer to *fputc()*.56318 **ERRORS**56319       Refer to *fputc()*.56320 **EXAMPLES**

56321       None.

56322 **APPLICATION USAGE**

56323       Since it may be implemented as a macro, *putc()* may treat a *stream* argument with side-effects  
 56324 incorrectly. In particular, *putc(c,\*f++)* does not necessarily work correctly. Therefore, use of this  
 56325 function is not recommended in such situations; *fputc()* should be used instead.

56326 **RATIONALE**

56327       None.

56328 **FUTURE DIRECTIONS**

56329       None.

56330 **SEE ALSO**56331       Section 2.5 (on page 495), *fputc()*

56332       XBD &lt;stdio.h&gt;

56333 **CHANGE HISTORY**

56334       First released in Issue 1. Derived from Issue 1 of the SVID.

56335 **Issue 7**

56336       POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0470 [14] is applied.

# putc\_unlocked()

## 56337 NAME

56338       putc\_unlocked — stdio with explicit client locking

## 56339 SYNOPSIS

```
56340 CX       #include <stdio.h>  
56341       int putc_unlocked(int c, FILE *stream);
```

## 56342 DESCRIPTION

56343       Refer to *getc\_unlocked()*.



56344 **NAME**

56345 putchar — put a byte on a stdout stream

56346 **SYNOPSIS**

56347 #include &lt;stdio.h&gt;

56348 int putchar(int c);

56349 **DESCRIPTION**

56350 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
56351 conflict between the requirements described here and the ISO C standard is unintentional. This  
56352 volume of POSIX.1-2008 defers to the ISO C standard.

56353 The function call *putchar(c)* shall be equivalent to *putc(c,stdout)*.56354 **RETURN VALUE**56355 Refer to *fputc()*.56356 **ERRORS**56357 Refer to *fputc()*.56358 **EXAMPLES**

56359 None.

56360 **APPLICATION USAGE**

56361 None.

56362 **RATIONALE**

56363 None.

56364 **FUTURE DIRECTIONS**

56365 None.

56366 **SEE ALSO**56367 [Section 2.5](#) (on page 495), *putc()*56368 XBD [<stdio.h>](#)56369 **CHANGE HISTORY**

56370 First released in Issue 1. Derived from Issue 1 of the SVID.

56371 **Issue 7**

56372 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0471 [14] is applied.

56373 **NAME**

56374 putchar\_unlocked — stdio with explicit client locking

56375 **SYNOPSIS**

```
56376 CX #include <stdio.h>  
56377 int putchar_unlocked(int c);
```

56378 **DESCRIPTION**56379 Refer to *getc\_unlocked()*.

56380 **NAME**

56381 putenv — change or add a value to an environment

56382 **SYNOPSIS**

```
56383 XSI      #include <stdlib.h>
56384         int putenv(char *string);
```

56385 **DESCRIPTION**

56386 The *putenv()* function shall use the *string* argument to set environment variable values. The  
 56387 *string* argument should point to a string of the form "*name=value*". The *putenv()* function shall  
 56388 make the value of the environment variable *name* equal to *value* by altering an existing variable  
 56389 or creating a new one. In either case, the string pointed to by *string* shall become part of the  
 56390 environment, so altering the string shall change the environment.

56391 The *putenv()* function need not be thread-safe.

56392 **RETURN VALUE**

56393 Upon successful completion, *putenv()* shall return 0; otherwise, it shall return a non-zero value  
 56394 and set *errno* to indicate the error.

56395 **ERRORS**

56396 The *putenv()* function may fail if:

56397 [ENOMEM]        Insufficient memory was available.

56398 **EXAMPLES**56399 **Changing the Value of an Environment Variable**

56400 The following example changes the value of the *HOME* environment variable to the value  
 56401 */usr/home*.

```
56402 #include <stdlib.h>
56403 ...
56404 static char *var = "HOME=/usr/home";
56405 int ret;
56406
56407 ret = putenv(var);
```

56407 **APPLICATION USAGE**

56408 The *putenv()* function manipulates the environment pointed to by *environ*, and can be used in  
 56409 conjunction with *getenv()*.

56410 See *exec()* for restrictions on changing the environment in multi-threaded applications.

56411 This routine may use *malloc()* to enlarge the environment.

56412 A potential error is to call *putenv()* with an automatic variable as the argument, then return from  
 56413 the calling function while *string* is still part of the environment.

56414 Although the space used by *string* is no longer used once a new string which defines *name* is  
 56415 passed to *putenv()*, if any thread in the application has used *getenv()* to retrieve a pointer to this  
 56416 variable, it should not be freed by calling *free()*. If the changed environment variable is one  
 56417 known by the system (such as the locale environment variables) the application should never  
 56418 free the buffer used by earlier calls to *putenv()* for the same variable.

56419 The *setenv()* function is preferred over this function. One reason is that *putenv()* is optional and  
 56420 therefore less portable. Another is that using *putenv()* can slow down environment searches, as  
 56421 explained in the RATIONALE section for *getenv()*.

56422 **RATIONALE**56423 Refer to the RATIONALE section in *setenv()*.56424 **FUTURE DIRECTIONS**

56425 None.

56426 **SEE ALSO**56427 *exec*, *free()*, *getenv()*, *malloc()*, *setenv()*

56428 XBD &lt;stdlib.h&gt;

56429 **CHANGE HISTORY**

56430 First released in Issue 1. Derived from Issue 1 of the SVID.

56431 **Issue 5**56432 The type of the argument to this function is changed from **const char \*** to **char \***. This was  
56433 indicated as a FUTURE DIRECTION in previous issues.

56434 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

56435 **Issue 6**56436 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/48 is applied, clarifying wording in the  
56437 DESCRIPTION and adding a new paragraph into APPLICATION USAGE referring readers to  
56438 *exec*.56439 **Issue 7**

56440 Austin Group Interpretation 1003.1-2001 #156 is applied.

56441 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0472 [167], XSH/TC1-2008/0473 [167],  
56442 XSH/TC1-2008/0474 [273,438], and XSH/TC1-2008/0475 [273] are applied.

**NAME**

putmsg, putpmsg — send a message on a STREAM (STREAMS)

**SYNOPSIS**

```
#include <stropts.h>

int putmsg(int fildes, const struct strbuf *ctlptr,
           const struct strbuf *dataptr, int flags);
int putpmsg(int fildes, const struct strbuf *ctlptr,
            const struct strbuf *dataptr, int band, int flags);
```

**DESCRIPTION**

The *putmsg()* function shall create a message from a process buffer(s) and send the message to a STREAMS file. The message may contain either a data part, a control part, or both. The data and control parts are distinguished by placement in separate buffers, as described below. The semantics of each part are defined by the STREAMS module that receives the message.

The *putpmsg()* function is equivalent to *putmsg()*, except that the process can send messages in different priority bands. Except where noted, all requirements on *putmsg()* also pertain to *putpmsg()*.

The *fildes* argument specifies a file descriptor referencing an open STREAM. The *ctlptr* and *dataptr* arguments each point to a **strbuf** structure.

The *ctlptr* argument points to the structure describing the control part, if any, to be included in the message. The *buf* member in the **strbuf** structure points to the buffer where the control information resides, and the *len* member indicates the number of bytes to be sent. The *maxlen* member is not used by *putmsg()*. In a similar manner, the argument *dataptr* specifies the data, if any, to be included in the message. The *flags* argument indicates what type of message should be sent and is described further below.

To send the data part of a message, the application shall ensure that *dataptr* is not a null pointer and the *len* member of *dataptr* is 0 or greater. To send the control part of a message, the application shall ensure that the corresponding values are set for *ctlptr*. No data (control) part shall be sent if either *dataptr(ctlptr)* is a null pointer or the *len* member of *dataptr(ctlptr)* is set to -1.

For *putmsg()*, if a control part is specified and *flags* is set to RS\_HIPRI, a high priority message shall be sent. If no control part is specified, and *flags* is set to RS\_HIPRI, *putmsg()* shall fail and set *errno* to [EINVAL]. If *flags* is set to 0, a normal message (priority band equal to 0) shall be sent. If a control part and data part are not specified and *flags* is set to 0, no message shall be sent and 0 shall be returned.

For *putpmsg()*, the flags are different. The *flags* argument is a bitmask with the following mutually-exclusive flags defined: MSG\_HIPRI and MSG\_BAND. If *flags* is set to 0, *putpmsg()* shall fail and set *errno* to [EINVAL]. If a control part is specified and *flags* is set to MSG\_HIPRI and *band* is set to 0, a high-priority message shall be sent. If *flags* is set to MSG\_HIPRI and either no control part is specified or *band* is set to a non-zero value, *putpmsg()* shall fail and set *errno* to [EINVAL]. If *flags* is set to MSG\_BAND, then a message shall be sent in the priority band specified by *band*. If a control part and data part are not specified and *flags* is set to MSG\_BAND, no message shall be sent and 0 shall be returned.

The *putmsg()* function shall block if the STREAM write queue is full due to internal flow control conditions, with the following exceptions:

- For high-priority messages, *putmsg()* shall not block on this condition and continues processing the message.
- For other messages, *putmsg()* shall not block but shall fail when the write queue is full and *O\_NONBLOCK* is set.

The *putmsg()* function shall also block, unless prevented by lack of internal resources, while waiting for the availability of message blocks in the STREAM, regardless of priority or whether *O\_NONBLOCK* has been specified. No partial message shall be sent.

#### RETURN VALUE

Upon successful completion, *putmsg()* and *putpmsg()* shall return 0; otherwise, they shall return -1 and set *errno* to indicate the error.

#### ERRORS

The *putmsg()* and *putpmsg()* functions shall fail if:

- |                  |                                                                                                                                                                                                                                                                                                                                                                                                                |
|------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [EAGAIN]         | A non-priority message was specified, the <i>O_NONBLOCK</i> flag is set, and the STREAM write queue is full due to internal flow control conditions; or buffers could not be allocated for the message that was to be created.                                                                                                                                                                                 |
| [EBADF]          | <i>fildev</i> is not a valid file descriptor open for writing.                                                                                                                                                                                                                                                                                                                                                 |
| [EINTR]          | A signal was caught during <i>putmsg()</i> .                                                                                                                                                                                                                                                                                                                                                                   |
| [EINVAL]         | An undefined value is specified in <i>flags</i> , or <i>flags</i> is set to <i>RS_HIPRI</i> or <i>MSG_HIPRI</i> and no control part is supplied, or the STREAM or multiplexer referenced by <i>fildev</i> is linked (directly or indirectly) downstream from a multiplexer, or <i>flags</i> is set to <i>MSG_HIPRI</i> and <i>band</i> is non-zero (for <i>putpmsg()</i> only).                                |
| [ENOSR]          | Buffers could not be allocated for the message that was to be created due to insufficient STREAMS memory resources.                                                                                                                                                                                                                                                                                            |
| [ENOSTR]         | A STREAM is not associated with <i>fildev</i> .                                                                                                                                                                                                                                                                                                                                                                |
| [ENXIO]          | A hangup condition was generated downstream for the specified STREAM.                                                                                                                                                                                                                                                                                                                                          |
| [EPIPE] or [EIO] | The <i>fildev</i> argument refers to a STREAMS-based pipe and the other end of the pipe is closed. A SIGPIPE signal is generated for the calling thread.                                                                                                                                                                                                                                                       |
| [ERANGE]         | The size of the data part of the message does not fall within the range specified by the maximum and minimum packet sizes of the topmost STREAM module. This value is also returned if the control part of the message is larger than the maximum configured size of the control part of a message, or if the data part of a message is larger than the maximum configured size of the data part of a message. |

In addition, *putmsg()* and *putpmsg()* shall fail if the STREAM head had processed an asynchronous error before the call. In this case, the value of *errno* does not reflect the result of *putmsg()* or *putpmsg()*, but reflects the prior error.

**EXAMPLES****Sending a High-Priority Message**

The value of *fd* is assumed to refer to an open STREAMS file. This call to *putmsg()* does the following:

1. Creates a high-priority message with a control part and a data part, using the buffers pointed to by *ctrlbuf* and *databuf*, respectively.
2. Sends the message to the STREAMS file identified by *fd*.

```
#include <stropts.h>
#include <string.h>
...
int fd;
char *ctrlbuf = "This is the control part";
char *databuf = "This is the data part";
struct strbuf ctrl;
struct strbuf data;
int ret;

ctrl.buf = ctrlbuf;
ctrl.len = strlen(ctrlbuf);

data.buf = databuf;
data.len = strlen(databuf);

ret = putmsg(fd, &ctrl, &data, MSG_HIPRI);
```

**Using putpmsg()**

This example has the same effect as the previous example. In this example, however, the *putpmsg()* function creates and sends the message to the STREAMS file.

```
#include <stropts.h>
#include <string.h>
...
int fd;
char *ctrlbuf = "This is the control part";
char *databuf = "This is the data part";
struct strbuf ctrl;
struct strbuf data;
int ret;

ctrl.buf = ctrlbuf;
ctrl.len = strlen(ctrlbuf);

data.buf = databuf;
data.len = strlen(databuf);

ret = putpmsg(fd, &ctrl, &data, 0, MSG_HIPRI);
```

**APPLICATION USAGE**

None.

56564 **RATIONALE**

56565 None.

56566 **FUTURE DIRECTIONS**56567 The *putmsg()* and *putpmsg()* functions may be removed in a future version.56568 **SEE ALSO**56569 [Section 2.6](#) (on page 500), *getmsg()*, *poll()*, *read()*, *write()*56570 XBD [<stropts.h>](#)56571 **CHANGE HISTORY**

56572 First released in Issue 4, Version 2.

56573 **Issue 5**

56574 Moved from X/OPEN UNIX extension to BASE.

56575 The following text is removed from the DESCRIPTION: “The STREAM head guarantees that the  
56576 control part of a message generated by *putmsg()* is at least 64 bytes in length”.

56577 **Issue 6**

56578 This function is marked as part of the XSI STREAMS Option Group.

56579 The normative text is updated to avoid use of the term “must” for application requirements.

56580 **Issue 7**56581 The *putmsg()* and *putpmsg()* functions are marked obsolescent.



56582 **NAME**

56583 puts — put a string on standard output

56584 **SYNOPSIS**

56585 #include &lt;stdio.h&gt;

56586 int puts(const char \*s);

56587 **DESCRIPTION**

56588 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 56589 conflict between the requirements described here and the ISO C standard is unintentional. This  
 56590 volume of POSIX.1-2008 defers to the ISO C standard.

56591 The *puts()* function shall write the string pointed to by *s*, followed by a <newline>, to the  
 56592 standard output stream *stdout*. The terminating null byte shall not be written.

56593 CX The last data modification and last file status change timestamps of the file shall be marked for  
 56594 update between the successful execution of *puts()* and the next successful completion of a call to  
 56595 *fflush()* or *fclose()* on the same stream or a call to *exit()* or *abort()*.

56596 **RETURN VALUE**

56597 Upon successful completion, *puts()* shall return a non-negative number. Otherwise, it shall  
 56598 CX return EOF, shall set an error indicator for the stream, and *errno* shall be set to indicate the error.

56599 **ERRORS**56600 Refer to *fputc()*.56601 **EXAMPLES**56602 **Printing to Standard Output**

56603 The following example gets the current time, converts it to a string using *localtime()* and  
 56604 *asctime()*, and prints it to standard output using *puts()*. It then prints the number of minutes to  
 56605 an event for which it is waiting.

```
56606 #include <time.h>
56607 #include <stdio.h>
56608 ...
56609 time_t now;
56610 int minutes_to_event;
56611 ...
56612 time(&now);
56613 printf("The time is ");
56614 puts(asctime(localtime(&now)));
56615 printf("There are %d minutes to the event.\n",
56616        minutes_to_event);
56617 ...
```

56618 **APPLICATION USAGE**56619 The *puts()* function appends a <newline>, while *fputs()* does not.

56620 This volume of POSIX.1-2008 requires that successful completion simply return a non-negative  
 56621 integer. There are at least three known different implementation conventions for this  
 56622 requirement:

- 56623 • Return a constant value.

56624                   • Return the last character written.

56625                   • Return the number of bytes written. Note that this implementation convention cannot be

56626                   adhered to for strings longer than {INT\_MAX} bytes as the value would not be

56627                   representable in the return type of the function. For backwards compatibility,

56628                   implementations can return the number of bytes for strings of up to {INT\_MAX} bytes, and

56629                   return {INT\_MAX} for all longer strings.

56630 **RATIONALE**

56631                   None.

56632 **FUTURE DIRECTIONS**

56633                   None.

56634 **SEE ALSO**

56635                   [Section 2.5](#) (on page 495), [fopen\(\)](#), [fputs\(\)](#), [putc\(\)](#)

56636                   XBD [<stdio.h>](#)

56637 **CHANGE HISTORY**

56638                   First released in Issue 1. Derived from Issue 1 of the SVID.

56639 **Issue 6**

56640                   Extensions beyond the ISO C standard are marked.

56641 **Issue 7**

56642                   Changes are made related to support for finegrained timestamps.

56643                   POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0476 [174,412] and

56644                   XSH/TC1-2008/0477 [14] are applied.

56645 **NAME**

56646 pututxline — put an entry into the user accounting database

56647 **SYNOPSIS**

```
56648 XSI      #include <utmpx.h>  
56649      struct utmpx *pututxline(const struct utmpx *utmpx);
```

56650 **DESCRIPTION**56651 Refer to *endutxent()*.

56652 **NAME**

56653 putwc — put a wide character on a stream

56654 **SYNOPSIS**

56655 #include &lt;stdio.h&gt;

56656 #include &lt;wchar.h&gt;

56657 wint\_t putwc(wchar\_t wc, FILE \*stream);

56658 **DESCRIPTION**

56659 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
56660 conflict between the requirements described here and the ISO C standard is unintentional. This  
56661 volume of POSIX.1-2008 defers to the ISO C standard.

56662 The *putwc()* function shall be equivalent to *fputwc()*, except that if it is implemented as a macro  
56663 it may evaluate *stream* more than once, so the argument should never be an expression with  
56664 side-effects.

56665 **RETURN VALUE**56666 Refer to *fputwc()*.56667 **ERRORS**56668 Refer to *fputwc()*.56669 **EXAMPLES**

56670 None.

56671 **APPLICATION USAGE**

56672 Since it may be implemented as a macro, *putwc()* may treat a *stream* argument with side-effects  
56673 incorrectly. In particular, *putwc(wc,\*f++)* need not work correctly. Therefore, use of this function  
56674 is not recommended; *fputwc()* should be used instead.

56675 **RATIONALE**

56676 None.

56677 **FUTURE DIRECTIONS**

56678 None.

56679 **SEE ALSO**56680 [Section 2.5](#) (on page 495), *fputwc()*56681 XBD [<stdio.h>](#), [<wchar.h>](#)56682 **CHANGE HISTORY**

56683 First released as a World-wide Portability Interface in Issue 4.

56684 **Issue 5**

56685 Aligned with ISO/IEC 9899:1990/Amendment 1:1995 (E). Specifically, the type of argument *wc*  
56686 is changed from **wint\_t** to **wchar\_t**.

56687 The Optional Header (OH) marking is removed from **<stdio.h>**.56688 **Issue 7**

56689 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0478 [14] is applied.

56690 **NAME**

56691       putwchar — put a wide character on a stdout stream

56692 **SYNOPSIS**

56693       #include &lt;wchar.h&gt;

56694       wint\_t putwchar(wchar\_t wc);

56695 **DESCRIPTION**

56696 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 56697       conflict between the requirements described here and the ISO C standard is unintentional. This  
 56698       volume of POSIX.1-2008 defers to the ISO C standard.

56699       The function call *putwchar(wc)* shall be equivalent to *putwc(wc,stdout)*.56700 **RETURN VALUE**56701       Refer to *fputwc()*.56702 **ERRORS**56703       Refer to *fputwc()*.56704 **EXAMPLES**

56705       None.

56706 **APPLICATION USAGE**

56707       None.

56708 **RATIONALE**

56709       None.

56710 **FUTURE DIRECTIONS**

56711       None.

56712 **SEE ALSO**56713       Section 2.5 (on page 495), *fputwc()*, *putwc()*

56714       XBD &lt;wchar.h&gt;

56715 **CHANGE HISTORY**

56716       First released in Issue 4.

56717 **Issue 5**

56718       Aligned with ISO/IEC 9899:1990/Amendment 1:1995 (E). Specifically, the type of argument *wc*  
 56719       is changed from **wint\_t** to **wchar\_t**.

56720 **Issue 7**

56721       POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0479 [14] is applied.

56722 **NAME**

56723       pwrite — write on a file

56724 **SYNOPSIS**

56725       #include &lt;unistd.h&gt;

56726       ssize\_t pwrite(int *fildes*, const void \**buf*, size\_t *nbyte*,56727           off\_t *offset*);56728 **DESCRIPTION**56729       Refer to *write()*.

56730 **NAME**

56731 qsort — sort a table of data

56732 **SYNOPSIS**

56733 #include &lt;stdlib.h&gt;

```
56734 void qsort(void *base, size_t nel, size_t width,
56735           int (*compar)(const void *, const void *));
```

56736 **DESCRIPTION**

56737 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 56738 conflict between the requirements described here and the ISO C standard is unintentional. This  
 56739 volume of POSIX.1-2008 defers to the ISO C standard.

56740 The *qsort()* function shall sort an array of *nel* objects, the initial element of which is pointed to by  
 56741 *base*. The size of each object, in bytes, is specified by the *width* argument. If the *nel* argument has  
 56742 the value zero, the comparison function pointed to by *compar* shall not be called and no  
 56743 rearrangement shall take place.

56744 The application shall ensure that the comparison function pointed to by *compar* does not alter the  
 56745 contents of the array. The implementation may reorder elements of the array between calls to the  
 56746 comparison function, but shall not alter the contents of any individual element.

56747 When the same objects (consisting of *width* bytes, irrespective of their current positions in the  
 56748 array) are passed more than once to the comparison function, the results shall be consistent with  
 56749 one another. That is, they shall define a total ordering on the array.

56750 The contents of the array shall be sorted in ascending order according to a comparison function.  
 56751 The *compar* argument is a pointer to the comparison function, which is called with two  
 56752 arguments that point to the elements being compared. The application shall ensure that the  
 56753 function returns an integer less than, equal to, or greater than 0, if the first argument is  
 56754 considered respectively less than, equal to, or greater than the second. If two members compare  
 56755 as equal, their order in the sorted array is unspecified.

56756 **RETURN VALUE**56757 The *qsort()* function shall not return a value.56758 **ERRORS**

56759 No errors are defined.

56760 **EXAMPLES**

56761 None.

56762 **APPLICATION USAGE**

56763 The comparison function need not compare every byte, so arbitrary data may be contained in  
 56764 the elements in addition to the values being compared.

56765 **RATIONALE**

56766 The requirement that each argument (hereafter referred to as *p*) to the comparison function is a  
 56767 pointer to elements of the array implies that for every call, for each argument separately, all of  
 56768 the following expressions are non-zero:

```
56769 ((char *)p - (char *)base) % width == 0
56770 (char *)p >= (char *)base
56771 (char *)p < (char *)base + nel * width
```

56772 **FUTURE DIRECTIONS**

56773 None.

56774 **SEE ALSO**56775 *alphasort()*56776 XBD *<stdlib.h>*56777 **CHANGE HISTORY**

56778 First released in Issue 1. Derived from Issue 1 of the SVID.

56779 **Issue 6**

56780 The normative text is updated to avoid use of the term “must” for application requirements.

56781 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/49 is applied, adding the last sentence to  
56782 the first non-shaded paragraph in the DESCRIPTION, and the following two paragraphs. The  
56783 RATIONALE is also updated. These changes are for alignment with the ISO C standard.



56784 **NAME**

56785       raise — send a signal to the executing process

56786 **SYNOPSIS**

56787       #include &lt;signal.h&gt;

56788       int raise(int sig);

56789 **DESCRIPTION**

56790 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 56791 conflict between the requirements described here and the ISO C standard is unintentional. This  
 56792 volume of POSIX.1-2008 defers to the ISO C standard.

56793 CX       The *raise()* function shall send the signal *sig* to the executing thread or process. If a signal  
 56794 handler is called, the *raise()* function shall not return until after the signal handler does.

56795 CX       The effect of the *raise()* function shall be equivalent to calling:

56796 pthread\_kill(pthread\_self(), sig);

56797 **RETURN VALUE**

56798 CX       Upon successful completion, 0 shall be returned. Otherwise, a non-zero value shall be returned  
 56799 and *errno* shall be set to indicate the error.

56800 **ERRORS**

56801       The *raise()* function shall fail if:

56802 CX       [EINVAL]       The value of the *sig* argument is an invalid signal number.

56803 **EXAMPLES**

56804       None.

56805 **APPLICATION USAGE**

56806       None.

56807 **RATIONALE**

56808       The term “thread” is an extension to the ISO C standard.

56809 **FUTURE DIRECTIONS**

56810       None.

56811 **SEE ALSO**

56812       *kill()*, *sigaction()*

56813       XBD <signal.h>, <sys/types.h>

56814 **CHANGE HISTORY**

56815       First released in Issue 4. Derived from the ANSI C standard.

56816 **Issue 5**

56817       The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

56818 **Issue 6**

56819       Extensions beyond the ISO C standard are marked.

56820       The following new requirements on POSIX implementations derive from alignment with the  
 56821 Single UNIX Specification:

- 56822       • In the RETURN VALUE section, the requirement to set *errno* on error is added.

- 56823
- 56824
- 56825
- Issue 7

Functionality relating to the Threads option is moved to the Base.
- The [EINVAL] error condition is added.

56826 **NAME**

56827 rand, rand\_r, srand — pseudo-random number generator

56828 **SYNOPSIS**

```
56829 #include <stdlib.h>
56830 int rand(void);
56831 OB CX int rand_r(unsigned *seed);
56832 void srand(unsigned seed);
```

56833 **DESCRIPTION**

56834 CX For *rand()* and *srand()*: The functionality described on this reference page is aligned with the  
 56835 ISO C standard. Any conflict between the requirements described here and the ISO C standard is  
 56836 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

56837 The *rand()* function shall compute a sequence of pseudo-random integers in the range  
 56838 XSI [0,{RAND\_MAX}] with a period of at least  $2^{32}$ .

56839 CX The *rand()* function need not be thread-safe.

56840 OB CX The *rand\_r()* function shall compute a sequence of pseudo-random integers in the range  
 56841 [0,{RAND\_MAX}]. (The value of the {RAND\_MAX} macro shall be at least 32767.)

56842 If *rand\_r()* is called with the same initial value for the object pointed to by *seed* and that object is  
 56843 not modified between successive returns and calls to *rand\_r()*, the same sequence shall be  
 56844 generated.

56845 The *srand()* function uses the argument as a seed for a new sequence of pseudo-random  
 56846 numbers to be returned by subsequent calls to *rand()*. If *srand()* is then called with the same  
 56847 seed value, the sequence of pseudo-random numbers shall be repeated. If *rand()* is called before  
 56848 any calls to *srand()* are made, the same sequence shall be generated as when *srand()* is first  
 56849 called with a seed value of 1.

56850 The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls  
 56851 *rand()* or *srand()*.

56852 **RETURN VALUE**

56853 The *rand()* function shall return the next pseudo-random number in the sequence.

56854 OB CX The *rand\_r()* function shall return a pseudo-random integer.

56855 The *srand()* function shall not return a value.

56856 **ERRORS**

56857 No errors are defined.

56858 **EXAMPLES**56859 **Generating a Pseudo-Random Number Sequence**

56860 The following example demonstrates how to generate a sequence of pseudo-random numbers.

```
56861 #include <stdio.h>
56862 #include <stdlib.h>
56863 ...
56864     long count, i;
56865     char *keystr;
56866     int elementlen, len;
56867     char c;
56868     ...
```

```

56869      /* Initial random number generator. */
56870      srand(1);

56871      /* Create keys using only lowercase characters */
56872      len = 0;
56873      for (i=0; i<count; i++) {
56874          while (len < elementlen) {
56875              c = (char) (rand() % 128);
56876              if (islower(c))
56877                  keystr[len++] = c;
56878          }

56879          keystr[len] = '\0';
56880          printf("%s Element%0*ld\n", keystr, elementlen, i);
56881          len = 0;
56882      }

```

### Generating the Same Sequence on Different Machines

The following code defines a pair of functions that could be incorporated into applications wishing to ensure that the same sequence of numbers is generated across different machines.

```

56886      static unsigned long next = 1;
56887      int myrand(void) /* RAND_MAX assumed to be 32767. */
56888      {
56889          next = next * 1103515245 + 12345;
56890          return((unsigned)(next/65536) % 32768);
56891      }

56892      void mysrand(unsigned seed)
56893      {
56894          next = seed;
56895      }

```

### APPLICATION USAGE

The *drand48()* and *random()* functions provide much more elaborate pseudo-random number generators.

The limitations on the amount of state that can be carried between one function call and another mean the *rand\_r()* function can never be implemented in a way which satisfies all of the requirements on a pseudo-random number generator. These functions should be avoided whenever non-trivial requirements (including safety) have to be fulfilled.

### RATIONALE

The ISO C standard *rand()* and *srand()* functions allow per-process pseudo-random streams shared by all threads. Those two functions need not change, but there has to be mutual-exclusion that prevents interference between two threads concurrently accessing the random number generator.

With regard to *rand()*, there are two different behaviors that may be wanted in a multi-threaded program:

1. A single per-process sequence of pseudo-random numbers that is shared by all threads that call *rand()*

- 56912            2. A different sequence of pseudo-random numbers for each thread that calls *rand()*
- 56913            This is provided by the modified thread-safe function based on whether the seed value is global
- 56914            to the entire process or local to each thread.
- 56915            This does not address the known deficiencies of the *rand()* function implementations, which
- 56916            have been approached by maintaining more state. In effect, this specifies new thread-safe forms
- 56917            of a deficient function.
- 56918    **FUTURE DIRECTIONS**
- 56919            The *rand\_r()* function may be removed in a future version.
- 56920    **SEE ALSO**
- 56921            *drand48()*, *initstate()*
- 56922            XBD <stdlib.h>
- 56923    **CHANGE HISTORY**
- 56924            First released in Issue 1. Derived from Issue 1 of the SVID.
- 56925    **Issue 5**
- 56926            The *rand\_r()* function is included for alignment with the POSIX Threads Extension.
- 56927            A note indicating that the *rand()* function need not be reentrant is added to the DESCRIPTION.
- 56928    **Issue 6**
- 56929            Extensions beyond the ISO C standard are marked.
- 56930            The *rand\_r()* function is marked as part of the Thread-Safe Functions option.
- 56931    **Issue 7**
- 56932            Austin Group Interpretation 1003.1-2001 #156 is applied.
- 56933            The *rand\_r()* function is marked obsolescent.

56934 **NAME**

56935 random — generate pseudo-random number

56936 **SYNOPSIS**

```
56937 XSI      #include <stdlib.h>  
56938          long random(void);
```

56939 **DESCRIPTION**56940 Refer to *initstate()*.

**NAME**

pread, read — read from a file

**SYNOPSIS**

```
#include <unistd.h>
```

```
ssize_t pread(int fildes, void *buf, size_t nbyte, off_t offset);
```

```
ssize_t read(int fildes, void *buf, size_t nbyte);
```

**DESCRIPTION**

The *read()* function shall attempt to read *nbyte* bytes from the file associated with the open file descriptor, *fildes*, into the buffer pointed to by *buf*. The behavior of multiple concurrent reads on the same pipe, FIFO, or terminal device is unspecified.

Before any action described below is taken, and if *nbyte* is zero, the *read()* function may detect and return errors as described below. In the absence of errors, or if error detection is not performed, the *read()* function shall return zero and have no other results.

On files that support seeking (for example, a regular file), the *read()* shall start at a position in the file given by the file offset associated with *fildes*. The file offset shall be incremented by the number of bytes actually read.

Files that do not support seeking—for example, terminals—always read from the current position. The value of a file offset associated with such a file is undefined.

No data transfer shall occur past the current end-of-file. If the starting position is at or after the end-of-file, 0 shall be returned. If the file refers to a device special file, the result of subsequent *read()* requests is implementation-defined.

If the value of *nbyte* is greater than {SSIZE\_MAX}, the result is implementation-defined.

When attempting to read from an empty pipe or FIFO:

- If no process has the pipe open for writing, *read()* shall return 0 to indicate end-of-file.
- If some process has the pipe open for writing and O\_NONBLOCK is set, *read()* shall return -1 and set *errno* to [EAGAIN].
- If some process has the pipe open for writing and O\_NONBLOCK is clear, *read()* shall block the calling thread until some data is written or the pipe is closed by all processes that had the pipe open for writing.

When attempting to read a file (other than a pipe or FIFO) that supports non-blocking reads and has no data currently available:

- If O\_NONBLOCK is set, *read()* shall return -1 and set *errno* to [EAGAIN].
- If O\_NONBLOCK is clear, *read()* shall block the calling thread until some data becomes available.
- The use of the O\_NONBLOCK flag has no effect if there is some data available.

The *read()* function reads data previously written to a file. If any portion of a regular file prior to the end-of-file has not been written, *read()* shall return bytes with value 0. For example, *lseek()* allows the file offset to be set beyond the end of existing data in the file. If data is later written at this point, subsequent reads in the gap between the previous end of data and the newly written data shall return bytes with value 0 until data is written into the gap.

Upon successful completion, where *nbyte* is greater than 0, *read()* shall mark for update the last data access timestamp of the file, and shall return the number of bytes read. This number shall never be greater than *nbyte*. The value returned may be less than *nbyte* if the number of bytes

|       |        |                                                                                                                            |
|-------|--------|----------------------------------------------------------------------------------------------------------------------------|
| 56984 |        | left in the file is less than <i>nbyte</i> , if the <i>read()</i> request was interrupted by a signal, or if the file is a |
| 56985 |        | pipe or FIFO or special file and has fewer than <i>nbyte</i> bytes immediately available for reading.                      |
| 56986 |        | For example, a <i>read()</i> from a file associated with a terminal may return one typed line of data.                     |
| 56987 |        | If a <i>read()</i> is interrupted by a signal before it reads any data, it shall return $-1$ with <i>errno</i> set to      |
| 56988 |        | [EINTR].                                                                                                                   |
| 56989 |        | If a <i>read()</i> is interrupted by a signal after it has successfully read some data, it shall return the                |
| 56990 |        | number of bytes read.                                                                                                      |
| 56991 |        | For regular files, no data transfer shall occur past the offset maximum established in the open                            |
| 56992 |        | file description associated with <i>fildes</i> .                                                                           |
| 56993 |        | If <i>fildes</i> refers to a socket, <i>read()</i> shall be equivalent to <i>recv()</i> with no flags set.                 |
| 56994 | SIO    | If the O_DSYNC and O_RSYNC bits have been set, read I/O operations on the file descriptor                                  |
| 56995 |        | shall complete as defined by synchronized I/O data integrity completion. If the O_SYNC and                                 |
| 56996 |        | O_RSYNC bits have been set, read I/O operations on the file descriptor shall complete as                                   |
| 56997 |        | defined by synchronized I/O file integrity completion.                                                                     |
| 56998 | SHM    | If <i>fildes</i> refers to a shared memory object, the result of the <i>read()</i> function is unspecified.                |
| 56999 | TYM    | If <i>fildes</i> refers to a typed memory object, the result of the <i>read()</i> function is unspecified.                 |
| 57000 | OB XSR | A <i>read()</i> from a STREAMS file can read data in three different modes: <i>byte-stream</i> mode, <i>message-</i>       |
| 57001 |        | <i>nondiscard</i> mode, and <i>message-discard</i> mode. The default shall be byte-stream mode. This can be                |
| 57002 |        | changed using the I_SRDOPT <i>ioctl()</i> request, and can be tested with I_GRDOPT <i>ioctl()</i> . In byte-               |
| 57003 |        | stream mode, <i>read()</i> shall retrieve data from the STREAM until as many bytes as were requested                       |
| 57004 |        | are transferred, or until there is no more data to be retrieved. Byte-stream mode ignores                                  |
| 57005 |        | message boundaries.                                                                                                        |
| 57006 |        | In STREAMS message-nondiscard mode, <i>read()</i> shall retrieve data until as many bytes as were                          |
| 57007 |        | requested are transferred, or until a message boundary is reached. If <i>read()</i> does not retrieve all                  |
| 57008 |        | the data in a message, the remaining data shall be left on the STREAM, and can be retrieved by                             |
| 57009 |        | the next <i>read()</i> call. Message-discard mode also retrieves data until as many bytes as were                          |
| 57010 |        | requested are transferred, or a message boundary is reached. However, unread data remaining                                |
| 57011 |        | in a message after the <i>read()</i> returns shall be discarded, and shall not be available for a                          |
| 57012 |        | subsequent <i>read()</i> , <i>getmsg()</i> , or <i>getpmsg()</i> call.                                                     |
| 57013 |        | How <i>read()</i> handles zero-byte STREAMS messages is determined by the current read mode                                |
| 57014 |        | setting. In byte-stream mode, <i>read()</i> shall accept data until it has read <i>nbyte</i> bytes, or until there         |
| 57015 |        | is no more data to read, or until a zero-byte message block is encountered. The <i>read()</i> function                     |
| 57016 |        | shall then return the number of bytes read, and place the zero-byte message back on the                                    |
| 57017 |        | STREAM to be retrieved by the next <i>read()</i> , <i>getmsg()</i> , or <i>getpmsg()</i> . In message-nondiscard           |
| 57018 |        | mode or message-discard mode, a zero-byte message shall return 0 and the message shall be                                  |
| 57019 |        | removed from the STREAM. When a zero-byte message is read as the first message on a                                        |
| 57020 |        | STREAM, the message shall be removed from the STREAM and 0 shall be returned, regardless                                   |
| 57021 |        | of the read mode.                                                                                                          |
| 57022 |        | A <i>read()</i> from a STREAMS file shall return the data in the message at the front of the STREAM                        |
| 57023 |        | head read queue, regardless of the priority band of the message.                                                           |
| 57024 |        | By default, STREAMS are in control-normal mode, in which a <i>read()</i> from a STREAMS file can                           |
| 57025 |        | only process messages that contain a data part but do not contain a control part. The <i>read()</i> shall                  |
| 57026 |        | fail if a message containing a control part is encountered at the STREAM head. This default                                |
| 57027 |        | action can be changed by placing the STREAM in either control-data mode or control-discard                                 |
| 57028 |        | mode with the I_SRDOPT <i>ioctl()</i> command. In control-data mode, <i>read()</i> shall convert any                       |
| 57029 |        | control part to data and pass it to the application before passing any data part originally present                        |



57030 in the same message. In control-discard mode, *read()* shall discard message control parts but  
 57031 return to the process any data part in the message.

57032 In addition, *read()* shall fail if the STREAM head had processed an asynchronous error before  
 57033 the call. In this case, the value of *errno* shall not reflect the result of *read()*, but reflect the prior  
 57034 error. If a hangup occurs on the STREAM being read, *read()* shall continue to operate normally  
 57035 until the STREAM head read queue is empty. Thereafter, it shall return 0.

57036 The *pread()* function shall be equivalent to *read()*, except that it shall read from a given position  
 57037 in the file without changing the file offset. The first three arguments to *pread()* are the same as  
 57038 *read()* with the addition of a fourth argument *offset* for the desired position inside the file. An  
 57039 attempt to perform a *pread()* on a file that is incapable of seeking shall result in an error.

## 57040 RETURN VALUE

57041 Upon successful completion, these functions shall return a non-negative integer indicating the  
 57042 number of bytes actually read. Otherwise, the functions shall return -1 and set *errno* to indicate  
 57043 the error.

## 57044 ERRORS

57045 These functions shall fail if:

57046 [EAGAIN] The file is neither a pipe, nor a FIFO, nor a socket, the O\_NONBLOCK flag is  
 57047 set for the file descriptor, and the thread would be delayed in the read  
 57048 operation.

57049 [EBADF] The *fildev* argument is not a valid file descriptor open for reading.

57050 OB XSR [EBADMSG] The file is a STREAM file that is set to control-normal mode and the message  
 57051 waiting to be read includes a control part.

57052 [EINTR] The read operation was terminated due to the receipt of a signal, and no data  
 57053 was transferred.

57054 OB XSR [EINVAL] The STREAM or multiplexer referenced by *fildev* is linked (directly or  
 57055 indirectly) downstream from a multiplexer.

57056 [EIO] The process is a member of a background process group attempting to read  
 57057 from its controlling terminal, and either the calling thread is blocking  
 57058 SIGTTIN or the process is ignoring SIGTTIN or the process group of the  
 57059 process is orphaned. This error may also be generated for implementation-  
 57060 defined reasons.

57061 XSI [EISDIR] The *fildev* argument refers to a directory and the implementation does not  
 57062 allow the directory to be read using *read()* or *pread()*. The *readdir()* function  
 57063 should be used instead.

57064 [EOVERFLOW] The file is a regular file, *nbyte* is greater than 0, the starting position is before  
 57065 the end-of-file, and the starting position is greater than or equal to the offset  
 57066 maximum established in the open file description associated with *fildev*.

57067 The *pread()* function shall fail if:

57068 [EINVAL] The file is a regular file or block special file, and the *offset* argument is  
 57069 negative. The file offset shall remain unchanged.

57070 [ESPIPE] The file is incapable of seeking.

57071 The *read()* function shall fail if:

57072 [EAGAIN] The file is a pipe or FIFO, the O\_NONBLOCK flag is set for the file descriptor,  
57073 and the thread would be delayed in the read operation.

57074 [EAGAIN] or [EWOULDBLOCK]  
57075 The file is a socket, the O\_NONBLOCK flag is set for the file descriptor, and  
57076 the thread would be delayed in the read operation.

57077 [ECONNRESET] A read was attempted on a socket and the connection was forcibly closed by  
57078 its peer.

57079 [ENOTCONN] A read was attempted on a socket that is not connected.

57080 [ETIMEDOUT] A read was attempted on a socket and a transmission timeout occurred.

57081 These functions may fail if:

57082 [EIO] A physical I/O error has occurred.

57083 [ENOBUFS] Insufficient resources were available in the system to perform the operation.

57084 [ENOMEM] Insufficient memory was available to fulfill the request.

57085 [ENXIO] A request was made of a nonexistent device, or the request was outside the  
57086 capabilities of the device.

## 57087 EXAMPLES

### 57088 Reading Data into a Buffer

57089 The following example reads data from the file associated with the file descriptor *fd* into the  
57090 buffer pointed to by *buf*.

```
57091 #include <sys/types.h>
57092 #include <unistd.h>
57093 ...
57094 char buf[20];
57095 size_t nbytes;
57096 ssize_t bytes_read;
57097 int fd;
57098 ...
57099 nbytes = sizeof(buf);
57100 bytes_read = read(fd, buf, nbytes);
57101 ...
```

## 57102 APPLICATION USAGE

57103 None.

## 57104 RATIONALE

57105 This volume of POSIX.1-2008 does not specify the value of the file offset after an error is  
57106 returned; there are too many cases. For programming errors, such as [EBADF], the concept is  
57107 meaningless since no file is involved. For errors that are detected immediately, such as  
57108 [EAGAIN], clearly the offset should not change. After an interrupt or hardware error, however,  
57109 an updated value would be very useful and is the behavior of many implementations.

57110 Note that a *read()* of zero bytes does not modify the last data access timestamp. A *read()* that  
57111 requests more than zero bytes, but returns zero, is required to modify the last data access  
57112 timestamp.

Implementations are allowed, but not required, to perform error checking for *read()* requests of zero bytes.

## Input and Output

The use of I/O with large byte counts has always presented problems. Ideas such as *lread()* and *lwrite()* (using and returning **longs**) were considered at one time. The current solution is to use abstract types on the ISO C standard function to *read()* and *write()*. The abstract types can be declared so that existing functions work, but can also be declared so that larger types can be represented in future implementations. It is presumed that whatever constraints limit the maximum range of **size\_t** also limit portable I/O requests to the same range. This volume of POSIX.1-2008 also limits the range further by requiring that the byte count be limited so that a signed return value remains meaningful. Since the return type is also a (signed) abstract type, the byte count can be defined by the implementation to be larger than an **int** can hold.

The standard developers considered adding atomicity requirements to a pipe or FIFO, but recognized that due to the nature of pipes and FIFOs there could be no guarantee of atomicity of reads of {PIPE\_BUF} or any other size that would be an aid to applications portability.

This volume of POSIX.1-2008 requires that no action be taken for *read()* or *write()* when *nbyte* is zero. This is not intended to take precedence over detection of errors (such as invalid buffer pointers or file descriptors). This is consistent with the rest of this volume of POSIX.1-2008, but the phrasing here could be misread to require detection of the zero case before any other errors. A value of zero is to be considered a correct value, for which the semantics are a no-op.

I/O is intended to be atomic to ordinary files and pipes and FIFOs. Atomic means that all the bytes from a single operation that started out together end up together, without interleaving from other I/O operations. It is a known attribute of terminals that this is not honored, and terminals are explicitly (and implicitly permanently) excepted, making the behavior unspecified. The behavior for other device types is also left unspecified, but the wording is intended to imply that future standards might choose to specify atomicity (or not).

There were recommendations to add format parameters to *read()* and *write()* in order to handle networked transfers among heterogeneous file system and base hardware types. Such a facility may be required for support by the OSI presentation of layer services. However, it was determined that this should correspond with similar C-language facilities, and that is beyond the scope of this volume of POSIX.1-2008. The concept was suggested to the developers of the ISO C standard for their consideration as a possible area for future work.

In 4.3 BSD, a *read()* or *write()* that is interrupted by a signal before transferring any data does not by default return an [EINTR] error, but is restarted. In 4.2 BSD, 4.3 BSD, and the Eighth Edition, there is an additional function, *select()*, whose purpose is to pause until specified activity (data to read, space to write, and so on) is detected on specified file descriptors. It is common in applications written for those systems for *select()* to be used before *read()* in situations (such as keyboard input) where interruption of I/O due to a signal is desired.

The issue of which files or file types are interruptible is considered an implementation design issue. This is often affected primarily by hardware and reliability issues.

There are no references to actions taken following an “unrecoverable error”. It is considered beyond the scope of this volume of POSIX.1-2008 to describe what happens in the case of hardware errors.

Earlier versions of this standard allowed two very different behaviors with regard to the handling of interrupts. In order to minimize the resulting confusion, it was decided that POSIX.1-2008 should support only one of these behaviors. Historical practice on AT&T-derived

systems was to have *read()* and *write()* return `-1` and set *errno* to `[EINTR]` when interrupted after some, but not all, of the data requested had been transferred. However, the US Department of Commerce FIPS 151-1 and FIPS 151-2 require the historical BSD behavior, in which *read()* and *write()* return the number of bytes actually transferred before the interrupt. If `-1` is returned when any data is transferred, it is difficult to recover from the error on a seekable device and impossible on a non-seekable device. Most new implementations support this behavior. The behavior required by POSIX.1-2008 is to return the number of bytes transferred.

POSIX.1-2008 does not specify when an implementation that buffers *read()*s actually moves the data into the user-supplied buffer, so an implementation may choose to do this at the latest possible moment. Therefore, an interrupt arriving earlier may not cause *read()* to return a partial byte count, but rather to return `-1` and set *errno* to `[EINTR]`.

Consideration was also given to combining the two previous options, and setting *errno* to `[EINTR]` while returning a short count. However, not only is there no existing practice that implements this, it is also contradictory to the idea that when *errno* is set, the function responsible shall return `-1`.

This volume of POSIX.1-2008 intentionally does not specify any *pread()* errors related to pipes, FIFOs, and sockets other than `[ESPIPE]`.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*fcntl()*, *ioctl()*, *lseek()*, *open()*, *pipe()*, *readv()*

XBD Chapter 11 (on page 199), `<stropts.h>`, `<sys/uio.h>`, `<unistd.h>`

#### CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

##### Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

Large File Summit extensions are added.

The *pread()* function is added.

##### Issue 6

The DESCRIPTION and ERRORS sections are updated so that references to STREAMS are marked as part of the XSI STREAMS Option Group.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION now states that if *read()* is interrupted by a signal after it has successfully read some data, it returns the number of bytes read. In Issue 3, it was optional whether *read()* returned the number of bytes read, or whether it returned `-1` with *errno* set to `[EINTR]`. This is a FIPS requirement.
- In the DESCRIPTION, text is added to indicate that for regular files, no data transfer occurs past the offset maximum established in the open file description associated with *files*. This change is to support large files.
- The `[EOVERFLOW]` mandatory error condition is added.

- 57201           • The [ENXIO] optional error condition is added.
- 57202       Text referring to sockets is added to the DESCRIPTION.
- 57203       The following changes were made to align with the IEEE P1003.1a draft standard:
- 57204           • The effect of reading zero bytes is clarified.
- 57205       The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that
- 57206       *read()* results are unspecified for typed memory objects.
- 57207       New RATIONALE is added to explain the atomicity requirements for input and output
- 57208       operations.
- 57209       The following error conditions are added for operations on sockets: [EAGAIN],
- 57210       [ECONNRESET], [ENOTCONN], and [ETIMEDOUT].
- 57211       The [EIO] error is made optional.
- 57212       The following error conditions are added for operations on sockets: [ENOBUFFS] and
- 57213       [ENOMEM].
- 57214       The *readv()* function is split out into a separate reference page.
- 57215       IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/108 is applied, updating the [EAGAIN]
- 57216       error in the ERRORS section from “the process would be delayed” to “the thread would be
- 57217       delayed”.
- 57218       IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/109 is applied, making an editorial
- 57219       correction in the RATIONALE section.
- 57220   **Issue 7**
- 57221       The *pread()* function is moved from the XSI option to the Base.
- 57222       Functionality relating to the XSI STREAMS option is marked obsolescent.
- 57223       Changes are made related to support for finegrained timestamps.
- 57224       POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0480 [218], XSH/TC1-2008/0481 [79],
- 57225       XSH/TC1-2008/0482 [218], XSH/TC1-2008/0483 [218], XSH/TC1-2008/0484 [218], and
- 57226       XSH/TC1-2008/0485 [218,428] are applied.

## NAME

readdir, readdir\_r — read a directory

## SYNOPSIS

```
#include <dirent.h>

struct dirent *readdir(DIR *dirp);
int readdir_r(DIR *restrict dirp, struct dirent *restrict entry,
               struct dirent **restrict result);
```

## DESCRIPTION

The type **DIR**, which is defined in the **<dirent.h>** header, represents a *directory stream*, which is an ordered sequence of all the directory entries in a particular directory. Directory entries represent files; files may be removed from a directory or added to a directory asynchronously to the operation of *readdir()*.

The *readdir()* function shall return a pointer to a structure representing the directory entry at the current position in the directory stream specified by the argument *dirp*, and position the directory stream at the next entry. It shall return a null pointer upon reaching the end of the directory stream. The structure **dirent** defined in the **<dirent.h>** header describes a directory entry. The value of the structure's *d\_ino* member shall be set to the file serial number of the file named by the *d\_name* member. If the *d\_name* member names a symbolic link, the value of the *d\_ino* member shall be set to the file serial number of the symbolic link itself.

The *readdir()* function shall not return directory entries containing empty names. If entries for dot or dot-dot exist, one entry shall be returned for dot and one entry shall be returned for dot-dot; otherwise, they shall not be returned.

The application shall not modify the structure to which the return value of *readdir()* points, nor any storage areas pointed to by pointers within the structure. The returned pointer, and pointers within the structure, might be invalidated or the structure or the storage areas might be overwritten by a subsequent call to *readdir()* on the same directory stream. They shall not be affected by a call to *readdir()* on a different directory stream. The returned pointer, and pointers within the structure, might also be invalidated if the calling thread is terminated.

If a file is removed from or added to the directory after the most recent call to *opendir()* or *rewinddir()*, whether a subsequent call to *readdir()* returns an entry for that file is unspecified.

The *readdir()* function may buffer several directory entries per actual read operation; *readdir()* shall mark for update the last data access timestamp of the directory each time the directory is actually read.

After a call to *fork()*, either the parent or child (but not both) may continue processing the directory stream using *readdir()*, *rewinddir()*, or *seekdir()*. If both the parent and child processes use these functions, the result is undefined.

The *readdir()* function need not be thread-safe.

Applications wishing to check for error situations should set *errno* to 0 before calling *readdir()*. If *errno* is set to non-zero on return, an error occurred.

The *readdir\_r()* function shall initialize the **dirent** structure referenced by *entry* to represent the directory entry at the current position in the directory stream referred to by *dirp*, store a pointer to this structure at the location referenced by *result*, and position the directory stream at the next entry.

The storage pointed to by *entry* shall be large enough for a **dirent** with an array of **char** *d\_name* members containing at least {NAME\_MAX}+1 elements.



Upon successful return, the pointer returned at *\*result* shall have the same value as the argument *entry*. Upon reaching the end of the directory stream, this pointer shall have the value NULL.

The *readdir\_r()* function shall not return directory entries containing empty names.

If a file is removed from or added to the directory after the most recent call to *opendir()* or *rewinddir()*, whether a subsequent call to *readdir\_r()* returns an entry for that file is unspecified.

The *readdir\_r()* function may buffer several directory entries per actual read operation; *readdir\_r()* shall mark for update the last data access timestamp of the directory each time the directory is actually read.

## RETURN VALUE

Upon successful completion, *readdir()* shall return a pointer to an object of type **struct dirent**. When an error is encountered, a null pointer shall be returned and *errno* shall be set to indicate the error. When the end of the directory is encountered, a null pointer shall be returned and *errno* is not changed.

If successful, the *readdir\_r()* function shall return zero; otherwise, an error number shall be returned to indicate the error.

## ERRORS

These functions shall fail if:

[EOVERFLOW] One of the values in the structure to be returned cannot be represented correctly.

These functions may fail if:

[EBADF] The *dirp* argument does not refer to an open directory stream.

[ENOENT] The current position of the directory stream is invalid.

## EXAMPLES

The following sample program searches the current directory for each of the arguments supplied on the command line.

```
#include <dirent.h>
#include <errno.h>
#include <stdio.h>
#include <string.h>

static void lookup(const char *arg)
{
    DIR *dirp;
    struct dirent *dp;

    if ((dirp = opendir(".")) == NULL) {
        perror("couldn't open '.');
        return;
    }

    do {
        errno = 0;
        if ((dp = readdir(dirp)) != NULL) {
            if (strcmp(dp->d_name, arg) != 0)
                continue;

            (void) printf("found %s\n", arg);
            (void) closedir(dirp);
        }
    } while (1);
}
```

```

57316             return;
57317         }
57318     } while (dp != NULL);
57319     if (errno != 0)
57320         perror("error reading directory");
57321     else
57322         (void) printf("failed to find %s\n", arg);
57323     (void) closedir(dirp);
57324     return;
57325 }
57326 int main(int argc, char *argv[])
57327 {
57328     int i;
57329     for (i = 1; i < argc; i++)
57330         lookup(argv[i]);
57331     return (0);
57332 }

```

### APPLICATION USAGE

The `readdir()` function should be used in conjunction with `opendir()`, `closedir()`, and `rewinddir()` to examine the contents of the directory.

The `readdir_r()` function is thread-safe and shall return values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call.

### RATIONALE

The returned value of `readdir()` merely *represents* a directory entry. No equivalence should be inferred.

Historical implementations of `readdir()` obtain multiple directory entries on a single read operation, which permits subsequent `readdir()` operations to operate from the buffered information. Any wording that required each successful `readdir()` operation to mark the directory last data access timestamp for update would disallow such historical performance-oriented implementations.

When returning a directory entry for the root of a mounted file system, some historical implementations of `readdir()` returned the file serial number of the underlying mount point, rather than of the root of the mounted file system. This behavior is considered to be a bug, since the underlying file serial number has no significance to applications.

Since `readdir()` returns NULL when it detects an error and when the end of the directory is encountered, an application that needs to tell the difference must set `errno` to zero before the call and check it if NULL is returned. Since the function must not change `errno` in the second case and must set it to a non-zero value in the first case, a zero `errno` after a call returning NULL indicates end-of-directory; otherwise, an error.

Routines to deal with this problem more directly were proposed:

```

57356 int derror (dirp)
57357 DIR *dirp;
57358
57358 void clearerr (dirp)
57359 DIR *dirp;

```

The first would indicate whether an error had occurred, and the second would clear the error



indication. The simpler method involving *errno* was adopted instead by requiring that *readdir()* not change *errno* when end-of-directory is encountered.

An error or signal indicating that a directory has changed while open was considered but rejected.

The thread-safe version of the directory reading function returns values in a user-supplied buffer instead of possibly using a static data area that may be overwritten by each call. Either the {NAME\_MAX} compile-time constant or the corresponding *pathconf()* option can be used to determine the maximum sizes of returned pathnames.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*closedir()*, *dirfd()*, *exec*, *fdopendir()*, *fstatat()*, *rewinddir()*, *symlink()*

XBD <dirent.h>, <sys/types.h>

## CHANGE HISTORY

First released in Issue 2.

### Issue 5

Large File Summit extensions are added.

The *readdir\_r()* function is included for alignment with the POSIX Threads Extension.

A note indicating that the *readdir()* function need not be reentrant is added to the DESCRIPTION.

### Issue 6

The *readdir\_r()* function is marked as part of the Thread-Safe Functions option.

The Open Group Corrigendum U026/7 is applied, correcting the prototype for *readdir\_r()*.

The Open Group Corrigendum U026/8 is applied, clarifying the wording of the successful return for the *readdir\_r()* function.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- A statement is added to the DESCRIPTION indicating the disposition of certain fields in **struct dirent** when an entry refers to a symbolic link.
- The [EOVERFLOW] mandatory error condition is added. This change is to support large files.
- The [ENOENT] optional error condition is added.

The APPLICATION USAGE section is updated to include a note on the thread-safe function and its avoidance of possibly using a static data area.

The **restrict** keyword is added to the *readdir\_r()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/50 is applied, replacing the EXAMPLES section with a new example.

57402 **Issue 7**

57403 Austin Group Interpretation 1003.1-2001 #059 is applied, updating the ERRORS section.

57404 Austin Group Interpretation 1003.1-2001 #156 is applied.

57405 The *readdir\_r()* function is moved from the Thread-Safe Functions option to the Base.

57406 Changes are made related to support for finegrained timestamps.

57407 The value of the *d\_ino* member is no longer unspecified for symbolic links.

57408 SD5-XSH-ERN-193 is applied.

57409 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0486 [75] is applied.

57410 **NAME**

57411 readlink, readlinkat — read the contents of a symbolic link

57412 **SYNOPSIS**

57413 #include &lt;unistd.h&gt;

57414 ssize\_t readlink(const char \*restrict path, char \*restrict buf,  
57415 size\_t bufsiz);

57416 OH #include &lt;fcntl.h&gt;

57417

+  
+57418 ssize\_t readlinkat(int fd, const char \*restrict path,  
57419 char \*restrict buf, size\_t bufsiz);57420 **DESCRIPTION**57421 The *readlink()* function shall place the contents of the symbolic link referred to by *path* in the  
57422 buffer *buf* which has size *bufsiz*. If the number of bytes in the symbolic link is less than *bufsiz*,  
57423 the contents of the remainder of *buf* are unspecified. If the *buf* argument is not large enough to  
57424 contain the link content, the first *bufsiz* bytes shall be placed in *buf*.57425 If the value of *bufsiz* is greater than {SSIZE\_MAX}, the result is implementation-defined.57426 Upon successful completion, *readlink()* shall mark for update the last data access timestamp of  
57427 the symbolic link.57428 The *readlinkat()* function shall be equivalent to the *readlink()* function except in the case where  
57429 *path* specifies a relative path. In this case the symbolic link whose content is read is relative to the  
57430 directory associated with the file descriptor *fd* instead of the current working directory. If the  
57431 access mode of the open file description associated with the file descriptor is not O\_SEARCH, |  
57432 the function shall check whether directory searches are permitted using the current permissions |  
57433 of the directory underlying the file descriptor. If the access mode is O\_SEARCH, the function |  
57434 shall not perform the check.57435 If *readlinkat()* is passed the special value AT\_FDCWD in the *fd* parameter, the current working  
57436 directory shall be used and the behavior shall be identical to a call to *readlink()*.57437 **RETURN VALUE**57438 Upon successful completion, these functions shall return the count of bytes placed in the buffer.  
57439 Otherwise, these functions shall return a value of -1, leave the buffer unchanged, and set *errno* to  
57440 indicate the error.57441 **ERRORS**

57442 These functions shall fail if:

57443 [EACCES] Search permission is denied for a component of the path prefix of *path*.57444 [EINVAL] The *path* argument names a file that is not a symbolic link.

57445 [EIO] An I/O error occurred while reading from the file system.

57446 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*  
57447 argument.

57448 [ENAMETOOLONG]

57449 The length of a component of a pathname is longer than {NAME\_MAX}.

57450 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.

57451 [ENOTDIR] A component of the path prefix names an existing file that is neither a  
 57452 directory nor a symbolic link to a directory, or the *path* argument contains at  
 57453 least one non-`<slash>` character and ends with one or more trailing `<slash>`  
 57454 characters and the last pathname component names an existing file that is  
 57455 neither a directory nor a symbolic link to a directory.

57456 The *readlinkat*() function shall fail if:

57457 [EACCES] The access mode of the open file description associated with *fd* is not  
 57458 O\_SEARCH and the permissions of the directory underlying *fd* do not permit  
 57459 directory searches.

57460 [EBADF] The *path* argument does not specify an absolute path and the *fd* argument is  
 57461 neither AT\_FDCWD nor a valid file descriptor open for reading or searching.

57462 [ENOTDIR] The *path* argument is not an absolute path and *fd* is a file descriptor associated  
 57463 with a non-directory file.

57464 These functions may fail if:

57465 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
 57466 resolution of the *path* argument.

57467 [ENAMETOOLONG]  
 57468 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
 57469 symbolic link produced an intermediate result with a length that exceeds  
 57470 {PATH\_MAX}.

## 57471 EXAMPLES

### 57472 Reading the Name of a Symbolic Link

57473 The following example shows how to read the name of a symbolic link named `/modules/pass1`.

```
57474 #include <unistd.h>
57475 char buf[1024];
57476 ssize_t len;
57477 ...
57478 if ((len = readlink("/modules/pass1", buf, sizeof(buf)-1)) != -1)
57479     buf[len] = '\0';
```

## 57480 APPLICATION USAGE

57481 Conforming applications should not assume that the returned contents of the symbolic link are  
 57482 null-terminated.

## 57483 RATIONALE

57484 The type associated with *bufsiz* is a **size\_t** in order to be consistent with both the ISO C standard  
 57485 and the definition of *read()*. The behavior specified for *readlink()* when *bufsiz* is zero represents  
 57486 historical practice. For this case, the standard developers considered a change whereby *readlink()*  
 57487 would return the number of non-null bytes contained in the symbolic link with the buffer *buf*  
 57488 remaining unchanged; however, since the **stat** structure member *st\_size* value can be used to  
 57489 determine the size of buffer necessary to contain the contents of the symbolic link as returned by  
 57490 *readlink()*, this proposal was rejected, and the historical practice retained.

57491 The purpose of the *readlinkat()* function is to read the content of symbolic links in directories  
 57492 other than the current working directory without exposure to race conditions. Any part of the  
 57493 path of a file could be changed in parallel to a call to *readlink()*, resulting in unspecified behavior.  
 57494 By opening a file descriptor for the target directory and using the *readlinkat()* function it can be

57495 guaranteed that the symbolic link read is located relative to the desired directory.

## 57496 FUTURE DIRECTIONS

57497 None.

## 57498 SEE ALSO

57499 *fstatat()*, *symlink()*

57500 XBD *<fcntl.h>*, *<unistd.h>*

## 57501 CHANGE HISTORY

57502 First released in Issue 4, Version 2.

### 57503 Issue 5

57504 Moved from X/OPEN UNIX extension to BASE.

### 57505 Issue 6

57506 The return type is changed to **ssize\_t**, to align with the IEEE P1003.1a draft standard.

57507 The following new requirements on POSIX implementations derive from alignment with the  
57508 Single UNIX Specification:

- 57509 • This function is made mandatory.
- 57510 • In this function it is possible for the return value to exceed the range of the type **ssize\_t**  
57511 (since **size\_t** has a larger range of positive values than **ssize\_t**). A sentence restricting the  
57512 size of the **size\_t** object is added to the description to resolve this conflict.

57513 The following changes are made for alignment with the ISO POSIX-1: 1996 standard:

- 57514 • The FUTURE DIRECTIONS section is changed to None.

57515 The following changes were made to align with the IEEE P1003.1a draft standard:

- 57516 • The [ELOOP] optional error condition is added.

57517 The **restrict** keyword is added to the *readlink()* prototype for alignment with the  
57518 ISO/IEC 9899: 1999 standard.

### 57519 Issue 7

57520 Austin Group Interpretation 1003.1-2001 #143 is applied.

57521 SD5-XSH-ERN-189 is applied, updating the ERRORS section.

57522 The *readlinkat()* function is added from The Open Group Technical Standard, 2006, Extended  
57523 API Set Part 2.

57524 The [EACCES] error is removed from the “may fail” error conditions.

57525 Changes are made to allow a directory to be opened for searching.

57526 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a  
57527 pathname exists but is not a directory or a symbolic link to a directory.

57528 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0487 [120], XSH/TC1-2008/0488 [461],  
57529 XSH/TC1-2008/0489 [143], XSH/TC1-2008/0490 [324], XSH/TC1-2008/0491 [278],  
57530 XSH/TC1-2008/0492 [278], XSH/TC1-2008/0493 [455], and XSH/TC1-2008/0494 [151,231] are  
57531 applied.

**NAME**

readv — read a vector

**SYNOPSIS**

```
XSI    #include <sys/uio.h>
      ssize_t readv(int fildes, const struct iovec *iov, int iovcnt);
```

**DESCRIPTION**

The *readv()* function shall be equivalent to *read()*, except as described below. The *readv()* function shall place the input data into the *iovcnt* buffers specified by the members of the *iov* array: *iov*[0], *iov*[1], ..., *iov*[*iovcnt*−1]. The *iovcnt* argument is valid if greater than 0 and less than or equal to {IOV\_MAX}.

Each *iovec* entry specifies the base address and length of an area in memory where data should be placed. The *readv()* function shall always fill an area completely before proceeding to the next.

Upon successful completion, *readv()* shall mark for update the last data access timestamp of the file.

**RETURN VALUE**

Refer to *read()*.

**ERRORS**

Refer to *read()*.

In addition, the *readv()* function shall fail if:

[EINVAL]        The sum of the *iov\_len* values in the *iov* array overflowed an *ssize\_t*.

The *readv()* function may fail if:

[EINVAL]        The *iovcnt* argument was less than or equal to 0, or greater than {IOV\_MAX}.

**EXAMPLES****Reading Data into an Array**

The following example reads data from the file associated with the file descriptor *fd* into the buffers specified by members of the *iov* array.

```
#include <sys/types.h>
#include <sys/uio.h>
#include <unistd.h>
...
ssize_t bytes_read;
int fd;
char buf0[20];
char buf1[30];
char buf2[40];
int iovcnt;
struct iovec iov[3];

iov[0].iov_base = buf0;
iov[0].iov_len = sizeof(buf0);
iov[1].iov_base = buf1;
iov[1].iov_len = sizeof(buf1);
iov[2].iov_base = buf2;
```

```

57575         iov[2].iov_len = sizeof(buf2);
57576         ...
57577         iovcnt = sizeof(iov) / sizeof(struct iovec);
57578         bytes_read = readv(fd, iov, iovcnt);
57579         ...

```

#### 57580 APPLICATION USAGE

57581 None.

#### 57582 RATIONALE

57583 Refer to *read()*.

#### 57584 FUTURE DIRECTIONS

57585 None.

#### 57586 SEE ALSO

57587 *read()*, *writev()*

57588 XBD <sys/uio.h>

#### 57589 CHANGE HISTORY

57590 First released in Issue 4, Version 2.

#### 57591 Issue 6

57592 Split out from the *read()* reference page.

#### 57593 Issue 7

57594 Changes are made related to support for finegrained timestamps.

57595 **NAME**

57596       realloc — memory reallocator

57597 **SYNOPSIS**

57598       #include &lt;stdlib.h&gt;

57599       void \*realloc(void \*ptr, size\_t size);

57600 **DESCRIPTION**

57601 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 57602 conflict between the requirements described here and the ISO C standard is unintentional. This  
 57603 volume of POSIX.1-2008 defers to the ISO C standard.

57604       The *realloc()* function shall deallocate the old object pointed to by *ptr* and return a pointer to a  
 57605 new object that has the size specified by *size*. The contents of the new object shall be the same as  
 57606 that of the old object prior to deallocation, up to the lesser of the new and old sizes. Any bytes in  
 57607 the new object beyond the size of the old object have indeterminate values. If the size of the  
 57608 space requested is zero, the behavior shall be implementation-defined: either a null pointer is  
 57609 returned, or the behavior shall be as if the size were some non-zero value, except that the  
 57610 behavior is undefined if the returned pointer is used to access an object. If the space cannot be  
 57611 allocated, the object shall remain unchanged.

57612       If *ptr* is a null pointer, *realloc()* shall be equivalent to *malloc()* for the specified size.

57613       If *ptr* does not match a pointer returned earlier by *calloc()*, *malloc()*, or *realloc()* or if the space has  
 57614 previously been deallocated by a call to *free()* or *realloc()*, the behavior is undefined.

57615       The order and contiguity of storage allocated by successive calls to *realloc()* is unspecified. The  
 57616 pointer returned if the allocation succeeds shall be suitably aligned so that it may be assigned to  
 57617 a pointer to any type of object and then used to access such an object in the space allocated (until  
 57618 the space is explicitly freed or reallocated). Each such allocation shall yield a pointer to an object  
 57619 disjoint from any other object. The pointer returned shall point to the start (lowest byte address)  
 57620 of the allocated space. If the space cannot be allocated, a null pointer shall be returned.

57621 **RETURN VALUE**

57622       Upon successful completion, *realloc()* shall return a pointer to the (possibly moved) allocated  
 57623 space. If *size* is 0, either:

57624 CX       • A null pointer shall be returned and, if *ptr* is not a null pointer, *errno* shall be set to an  
 57625 implementation-defined value.

57626       • A pointer to the allocated space shall be returned, and the memory object pointed to by *ptr*  
 57627 shall be freed. The application shall ensure that the pointer is not used to access an object.

57628 CX       If there is not enough available memory, *realloc()* shall return a null pointer and set *errno* to  
 57629 CX       [ENOMEM]. If *realloc()* returns a null pointer and *errno* has been set to [ENOMEM], the  
 57630 memory referenced by *ptr* shall not be changed.

57631 **ERRORS**

57632       The *realloc()* function shall fail if:

57633 CX       [ENOMEM]       Insufficient memory is available.



**EXAMPLES**

None.

**APPLICATION USAGE**

The description of `realloc()` has been modified from previous versions of this standard to align with the ISO/IEC 9899:1999 standard. Previous versions explicitly permitted a call to `realloc(p, 0)` to free the space pointed to by `p` and return a null pointer. While this behavior could be interpreted as permitted by this version of the standard, the C language committee have indicated that this interpretation is incorrect. Applications should assume that if `realloc()` returns a null pointer, the space pointed to by `p` has not been freed. Since this could lead to double-frees, implementations should also set `errno` if a null pointer actually indicates a failure, and applications should only free the space if `errno` was changed.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

This standard defers to the ISO C standard. While that standard currently has language that might permit `realloc(p, 0)`, where `p` is not a null pointer, to free `p` while still returning a null pointer, the committee responsible for that standard is considering clarifying the language to explicitly prohibit that alternative.

**SEE ALSO**`calloc()`, `free()`, `malloc()`

XBD &lt;stdlib.h&gt;

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 6**

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE section, if there is not enough available memory, the setting of `errno` to [ENOMEM] is added.
- The [ENOMEM] error condition is added.

**Issue 7**

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0495 [400], XSH/TC1-2008/0496 [400], XSH/TC1-2008/0497 [400], and XSH/TC1-2008/0498 [400] are applied.

**NAME**

realpath — resolve a pathname

**SYNOPSIS**

```
XSI    #include <stdlib.h>

char *realpath(const char *restrict file_name,
               char *restrict resolved_name);
```

**DESCRIPTION**

The *realpath()* function shall derive, from the pathname pointed to by *file\_name*, an absolute pathname that resolves to the same directory entry, whose resolution does not involve '.', '..', or symbolic links. If *resolved\_name* is a null pointer, the generated pathname shall be stored as a null-terminated string in a buffer allocated as if by a call to *malloc()*. Otherwise, if {PATH\_MAX} is defined as a constant in the <limits.h> header, then the generated pathname shall be stored as a null-terminated string, up to a maximum of {PATH\_MAX} bytes, in the buffer pointed to by *resolved\_name*.

If *resolved\_name* is not a null pointer and {PATH\_MAX} is not defined as a constant in the <limits.h> header, the behavior is undefined.

**RETURN VALUE**

Upon successful completion, *realpath()* shall return a pointer to the buffer containing the resolved name. Otherwise, *realpath()* shall return a null pointer and set *errno* to indicate the error.

If the *resolved\_name* argument is a null pointer, the pointer returned by *realpath()* can be passed to *free()*.

If the *resolved\_name* argument is not a null pointer and the *realpath()* function fails, the contents of the buffer pointed to by *resolved\_name* are undefined.

**ERRORS**

The *realpath()* function shall fail if:

- |                |                                                                                                                                                                                                                                                                                                                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [EACCES]       | Search permission was denied for a component of the path prefix of <i>file_name</i> .                                                                                                                                                                                                                                                                                                                      |
| [EINVAL]       | The <i>file_name</i> argument is a null pointer.                                                                                                                                                                                                                                                                                                                                                           |
| [EIO]          | An error occurred while reading from the file system.                                                                                                                                                                                                                                                                                                                                                      |
| [ELOOP]        | A loop exists in symbolic links encountered during resolution of the <i>file_name</i> argument.                                                                                                                                                                                                                                                                                                            |
| [ENAMETOOLONG] | The length of a component of a pathname is longer than {NAME_MAX}.                                                                                                                                                                                                                                                                                                                                         |
| [ENOENT]       | A component of <i>file_name</i> does not name an existing file or <i>file_name</i> points to an empty string.                                                                                                                                                                                                                                                                                              |
| [ENOTDIR]      | A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>file_name</i> argument contains at least one non- <i>&lt;slash&gt;</i> character and ends with one or more trailing <i>&lt;slash&gt;</i> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory. |

57706 The *realpath()* function may fail if:

- 57707 [EACCES] The *file\_name* argument does not begin with a <slash> and none of the
- 57708 symbolic links (if any) processed during pathname resolution of *file\_name* had
- 57709 contents that began with a <slash>, and either search permission was denied
- 57710 for the current directory or read or search permission was denied for a
- 57711 directory above the current directory in the file hierarchy.
- 57712 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during
- 57713 resolution of the *file\_name* argument.
- 57714 [ENAMETOOLONG]
- 57715 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a
- 57716 symbolic link produced an intermediate result with a length that exceeds
- 57717 {PATH\_MAX}.
- 57718 [ENOMEM] Insufficient storage space is available.

## 57719 EXAMPLES

### 57720 Generating an Absolute Pathname

57721 The following example generates an absolute pathname for the file identified by the *symlinkpath*

57722 argument. The generated pathname is stored in the buffer pointed to by *actualpath*.

```

57723 #include <stdlib.h>
57724 ...
57725 char *symlinkpath = "/tmp/symlink/file";
57726 char *actualpath;

57727 actualpath = realpath(symlinkpath, NULL);
57728 if (actualpath != NULL)
57729 {
57730     ... use actualpath ...
57731     free(actualpath);
57732 }
57733 else
57734 {
57735     ... handle error ...
57736 }
```

## 57737 APPLICATION USAGE

57738 For functions that allocate memory as if by *malloc()*, the application should release such memory

57739 when it is no longer required by a call to *free()*. For *realpath()*, this is the return value.

## 57740 RATIONALE

57741 Since *realpath()* has no *length* argument, if {PATH\_MAX} is not defined as a constant in

57742 <limits.h>, applications have no way of determining how large a buffer they need to allocate for

57743 it to be safe to pass to *realpath()*. A {PATH\_MAX} value obtained from a prior *pathconf()* call is

57744 out-of-date by the time *realpath()* is called. Hence the only reliable way to use *realpath()* when

57745 {PATH\_MAX} is not defined in <limits.h> is to pass a null pointer for *resolved\_name* so that

57746 *realpath()* will allocate a buffer of the necessary size.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*fpathconf()*, *free()*, *getcwd()*, *sysconf()*

XBD **<limits.h>**, **<stdlib.h>**

**CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

**Issue 6**

The **restrict** keyword is added to the *realpath()* prototype for alignment with the ISO/IEC 9899:1999 standard.

The wording of the mandatory [ELOOP] error condition is updated, and a second optional [ELOOP] error condition is added.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/51 is applied, adding new text to the DESCRIPTION for the case when *resolved\_name* is a null pointer, changing the [EINVAL] error text, adding text to the RATIONALE, and adding text to FUTURE DIRECTIONS.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/110 is applied, updating the ERRORS section to refer to the *file\_name* argument, rather than a nonexistent *path* argument.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #143 is applied.

This function is updated for passing a null pointer to *realpath()* for the *resolved\_name* argument.

The APPLICATION USAGE section is updated to clarify that memory is allocated as if by *malloc()*.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0499 [353], XSH/TC1-2008/0500 [324], and XSH/TC1-2008/0501 [353] are applied.

**NAME**

recv — receive a message from a connected socket

**SYNOPSIS**

```
#include <sys/socket.h>

ssize_t recv(int socket, void *buffer, size_t length, int flags);
```

**DESCRIPTION**

The *recv()* function shall receive a message from a connection-mode or connectionless-mode socket. It is normally used with connected sockets because it does not permit the application to retrieve the source address of received data.

The *recv()* function takes the following arguments:

|               |                                                                                                                                                                                                                                                                                                                                      |
|---------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>socket</i> | Specifies the socket file descriptor.                                                                                                                                                                                                                                                                                                |
| <i>buffer</i> | Points to a buffer where the message should be stored.                                                                                                                                                                                                                                                                               |
| <i>length</i> | Specifies the length in bytes of the buffer pointed to by the <i>buffer</i> argument.                                                                                                                                                                                                                                                |
| <i>flags</i>  | Specifies the type of message reception. Values of this argument are formed by logically OR'ing zero or more of the following values:                                                                                                                                                                                                |
| MSG_PEEK      | Peeks at an incoming message. The data is treated as unread and the next <i>recv()</i> or similar function shall still return this data.                                                                                                                                                                                             |
| MSG_OOB       | Requests out-of-band data. The significance and semantics of out-of-band data are protocol-specific.                                                                                                                                                                                                                                 |
| MSG_WAITALL   | On SOCK_STREAM sockets this requests that the function block until the full amount of data can be returned. The function may return the smaller amount of data if the socket is a message-based socket, if a signal is caught, if the connection is terminated, if MSG_PEEK was specified, or if an error is pending for the socket. |

The *recv()* function shall return the length of the message written to the buffer pointed to by the *buffer* argument. For message-based sockets, such as SOCK\_DGRAM and SOCK\_SEQPACKET, the entire message shall be read in a single operation. If a message is too long to fit in the supplied buffer, and MSG\_PEEK is not set in the *flags* argument, the excess bytes shall be discarded. For stream-based sockets, such as SOCK\_STREAM, message boundaries shall be ignored. In this case, data shall be returned to the user as soon as it becomes available, and no data shall be discarded.

If the MSG\_WAITALL flag is not set, data shall be returned only up to the end of the first message.

If no messages are available at the socket and O\_NONBLOCK is not set on the socket's file descriptor, *recv()* shall block until a message arrives. If no messages are available at the socket and O\_NONBLOCK is set on the socket's file descriptor, *recv()* shall fail and set *errno* to [EAGAIN] or [EWOULDBLOCK].

**RETURN VALUE**

Upon successful completion, *recv()* shall return the length of the message in bytes. If no messages are available to be received and the peer has performed an orderly shutdown, *recv()* shall return 0. Otherwise, -1 shall be returned and *errno* set to indicate the error.

**ERRORS**

The *recv()* function shall fail if:

[EAGAIN] or [EWOULDBLOCK]

The socket's file descriptor is marked O\_NONBLOCK and no data is waiting to be received; or MSG\_OOB is set and no out-of-band data is available and either the socket's file descriptor is marked O\_NONBLOCK or the socket does not support blocking to await out-of-band data.

[EBADF] The *socket* argument is not a valid file descriptor.

[ECONNRESET] A connection was forcibly closed by a peer.

[EINTR] The *recv()* function was interrupted by a signal that was caught, before any data was available.

[EINVAL] The MSG\_OOB flag is set and no out-of-band data is available.

[ENOTCONN] A receive is attempted on a connection-mode socket that is not connected.

[ENOTSOCK] The *socket* argument does not refer to a socket.

[EOPNOTSUPP] The specified flags are not supported for this socket type or protocol.

[ETIMEDOUT] The connection timed out during connection establishment, or due to a transmission timeout on active connection.

The *recv()* function may fail if:

[EIO] An I/O error occurred while reading from or writing to the file system.

[ENOBUFS] Insufficient resources were available in the system to perform the operation.

[ENOMEM] Insufficient memory was available to fulfill the request.

**EXAMPLES**

None.

**APPLICATION USAGE**

The *recv()* function is equivalent to *recvfrom()* with null pointer *address* and *address\_len* arguments, and to *read()* if the *socket* argument refers to a socket and the *flags* argument is 0.

The *select()* and *poll()* functions can be used to determine when data is available to be received.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*poll()*, *pselect()*, *read()*, *recvmsg()*, *recvfrom()*, *send()*, *sendmsg()*, *sendto()*, *shutdown()*, *socket()*, *write()*

XBD <sys/socket.h>

**CHANGE HISTORY**

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

57852 **Issue 7**  
57853

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0502 [462] is applied.

**NAME**

recvfrom — receive a message from a socket

**SYNOPSIS**

```
#include <sys/socket.h>

ssize_t recvfrom(int socket, void *restrict buffer, size_t length,
                 int flags, struct sockaddr *restrict address,
                 socklen_t *restrict address_len);
```

**DESCRIPTION**

The *recvfrom*() function shall receive a message from a connection-mode or connectionless-mode socket. It is normally used with connectionless-mode sockets because it permits the application to retrieve the source address of received data.

The *recvfrom*() function takes the following arguments:

|                    |                                                                                                                                                                                                                                                                                                                                      |
|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>socket</i>      | Specifies the socket file descriptor.                                                                                                                                                                                                                                                                                                |
| <i>buffer</i>      | Points to the buffer where the message should be stored.                                                                                                                                                                                                                                                                             |
| <i>length</i>      | Specifies the length in bytes of the buffer pointed to by the <i>buffer</i> argument.                                                                                                                                                                                                                                                |
| <i>flags</i>       | Specifies the type of message reception. Values of this argument are formed by logically OR'ing zero or more of the following values:                                                                                                                                                                                                |
| MSG_PEEK           | Peeks at an incoming message. The data is treated as unread and the next <i>recvfrom</i> () or similar function shall still return this data.                                                                                                                                                                                        |
| MSG_OOB            | Requests out-of-band data. The significance and semantics of out-of-band data are protocol-specific.                                                                                                                                                                                                                                 |
| MSG_WAITALL        | On SOCK_STREAM sockets this requests that the function block until the full amount of data can be returned. The function may return the smaller amount of data if the socket is a message-based socket, if a signal is caught, if the connection is terminated, if MSG_PEEK was specified, or if an error is pending for the socket. |
| <i>address</i>     | A null pointer, or points to a <b>sockaddr</b> structure in which the sending address is to be stored. The length and format of the address depend on the address family of the socket.                                                                                                                                              |
| <i>address_len</i> | Either a null pointer, if <i>address</i> is a null pointer, or a pointer to a <b>socklen_t</b> object which on input specifies the length of the supplied <b>sockaddr</b> structure, and on output specifies the length of the stored address.                                                                                       |

The *recvfrom*() function shall return the length of the message written to the buffer pointed to by the *buffer* argument. For message-based sockets, such as **SOCK\_RAW**, **SOCK\_DGRAM**, and **SOCK\_SEQPACKET**, the entire message shall be read in a single operation. If a message is too long to fit in the supplied buffer, and MSG\_PEEK is not set in the *flags* argument, the excess bytes shall be discarded. For stream-based sockets, such as **SOCK\_STREAM**, message boundaries shall be ignored. In this case, data shall be returned to the user as soon as it becomes available, and no data shall be discarded.

If the MSG\_WAITALL flag is not set, data shall be returned only up to the end of the first message.

Not all protocols provide the source address for messages. If the *address* argument is not a null



57898 pointer and the protocol provides the source address of messages, the source address of the  
 57899 received message shall be stored in the **sockaddr** structure pointed to by the *address* argument,  
 57900 and the length of this address shall be stored in the object pointed to by the *address\_len*  
 57901 argument.

57902 If the actual length of the address is greater than the length of the supplied **sockaddr** structure,  
 57903 the stored address shall be truncated.

57904 If the *address* argument is not a null pointer and the protocol does not provide the source address  
 57905 of messages, the value stored in the object pointed to by *address* is unspecified.

57906 If no messages are available at the socket and O\_NONBLOCK is not set on the socket's file  
 57907 descriptor, *recvfrom()* shall block until a message arrives. If no messages are available at the  
 57908 socket and O\_NONBLOCK is set on the socket's file descriptor, *recvfrom()* shall fail and set *errno*  
 57909 to [EAGAIN] or [EWOULDBLOCK].

#### 57910 RETURN VALUE

57911 Upon successful completion, *recvfrom()* shall return the length of the message in bytes. If no  
 57912 messages are available to be received and the peer has performed an orderly shutdown,  
 57913 *recvfrom()* shall return 0. Otherwise, the function shall return -1 and set *errno* to indicate the  
 57914 error.

#### 57915 ERRORS

57916 The *recvfrom()* function shall fail if:

57917 [EAGAIN] or [EWOULDBLOCK]

57918 The socket's file descriptor is marked O\_NONBLOCK and no data is waiting  
 57919 to be received; or MSG\_OOB is set and no out-of-band data is available and  
 57920 either the socket's file descriptor is marked O\_NONBLOCK or the socket does  
 57921 not support blocking to await out-of-band data.

57922 [EBADF] The *socket* argument is not a valid file descriptor.

57923 [ECONNRESET] A connection was forcibly closed by a peer.

57924 [EINTR] A signal interrupted *recvfrom()* before any data was available.

57925 [EINVAL] The MSG\_OOB flag is set and no out-of-band data is available.

57926 [ENOTCONN] A receive is attempted on a connection-mode socket that is not connected.

57927 [ENOTSOCK] The *socket* argument does not refer to a socket.

57928 [EOPNOTSUPP] The specified flags are not supported for this socket type.

57929 [ETIMEDOUT] The connection timed out during connection establishment, or due to a  
 57930 transmission timeout on active connection.

57931 The *recvfrom()* function may fail if:

57932 [EIO] An I/O error occurred while reading from or writing to the file system.

57933 [ENOBUFS] Insufficient resources were available in the system to perform the operation.

57934 [ENOMEM] Insufficient memory was available to fulfill the request.

57935 **EXAMPLES**

57936 None.

57937 **APPLICATION USAGE**57938 The *select()* and *poll()* functions can be used to determine when data is available to be received.57939 **RATIONALE**

57940 None.

57941 **FUTURE DIRECTIONS**

57942 None.

57943 **SEE ALSO**57944 *poll()*, *pselect()*, *read()*, *recv()*, *recvmsg()*, *send()*, *sendmsg()*, *sendto()*, *shutdown()*, *socket()*, *write()*

57945 XBD &lt;sys/socket.h&gt;

57946 **CHANGE HISTORY**

57947 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

57948 **Issue 7**

57949 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0503 [464] is applied.

## 57950 NAME

57951 recvmsg — receive a message from a socket

## 57952 SYNOPSIS

57953 #include &lt;sys/socket.h&gt;

57954 ssize\_t recvmsg(int socket, struct msghdr \*message, int flags);

## 57955 DESCRIPTION

57956 The *recvmsg()* function shall receive a message from a connection-mode or connectionless-mode  
 57957 socket. It is normally used with connectionless-mode sockets because it permits the application  
 57958 to retrieve the source address of received data.

57959 The *recvmsg()* function takes the following arguments:

|       |                |                                                                                                                                                                                                                                                                                                                                      |
|-------|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 57960 | <i>socket</i>  | Specifies the socket file descriptor.                                                                                                                                                                                                                                                                                                |
| 57961 | <i>message</i> | Points to a <b>msghdr</b> structure, containing both the buffer to store the source address and the buffers for the incoming message. The length and format of the address depend on the address family of the socket. The <i>msg_flags</i> member is ignored on input, but may contain meaningful values on output.                 |
| 57962 |                |                                                                                                                                                                                                                                                                                                                                      |
| 57963 |                |                                                                                                                                                                                                                                                                                                                                      |
| 57964 |                |                                                                                                                                                                                                                                                                                                                                      |
| 57965 | <i>flags</i>   | Specifies the type of message reception. Values of this argument are formed by logically OR'ing zero or more of the following values:                                                                                                                                                                                                |
| 57966 |                |                                                                                                                                                                                                                                                                                                                                      |
| 57967 | MSG_OOB        | Requests out-of-band data. The significance and semantics of out-of-band data are protocol-specific.                                                                                                                                                                                                                                 |
| 57968 |                |                                                                                                                                                                                                                                                                                                                                      |
| 57969 | MSG_PEEK       | Peeks at the incoming message.                                                                                                                                                                                                                                                                                                       |
| 57970 | MSG_WAITALL    | On SOCK_STREAM sockets this requests that the function block until the full amount of data can be returned. The function may return the smaller amount of data if the socket is a message-based socket, if a signal is caught, if the connection is terminated, if MSG_PEEK was specified, or if an error is pending for the socket. |
| 57971 |                |                                                                                                                                                                                                                                                                                                                                      |
| 57972 |                |                                                                                                                                                                                                                                                                                                                                      |
| 57973 |                |                                                                                                                                                                                                                                                                                                                                      |
| 57974 |                |                                                                                                                                                                                                                                                                                                                                      |
| 57975 |                |                                                                                                                                                                                                                                                                                                                                      |

57976 The *recvmsg()* function shall receive messages from unconnected or connected sockets and shall  
 57977 return the length of the message.

57978 The *recvmsg()* function shall return the total length of the message. For message-based sockets,  
 57979 such as SOCK\_DGRAM and SOCK\_SEQPACKET, the entire message shall be read in a single  
 57980 operation. If a message is too long to fit in the supplied buffers, and MSG\_PEEK is not set in the  
 57981 *flags* argument, the excess bytes shall be discarded, and MSG\_TRUNC shall be set in the  
 57982 *msg\_flags* member of the **msghdr** structure. For stream-based sockets, such as SOCK\_STREAM,  
 57983 message boundaries shall be ignored. In this case, data shall be returned to the user as soon as it  
 57984 becomes available, and no data shall be discarded.

57985 If the MSG\_WAITALL flag is not set, data shall be returned only up to the end of the first  
 57986 message.

57987 If no messages are available at the socket and O\_NONBLOCK is not set on the socket's file  
 57988 descriptor, *recvmsg()* shall block until a message arrives. If no messages are available at the  
 57989 socket and O\_NONBLOCK is set on the socket's file descriptor, the *recvmsg()* function shall fail  
 57990 and set *errno* to [EAGAIN] or [EWOULDBLOCK].

57991 In the **msghdr** structure, the *msg\_name* member may be a null pointer if the source address is not  
 57992 required. Otherwise, if the socket is unconnected, the *msg\_name* member points to a **sockaddr**  
 57993 structure in which the source address is to be stored, and the *msg\_namelen* member on input  
 57994 specifies the length of the supplied **sockaddr** structure and on output specifies the length of the

stored address. If the actual length of the address is greater than the length of the supplied **sockaddr** structure, the stored address shall be truncated. If the socket is connected, the *msg\_name* and *msg\_namelen* members shall be ignored. The *msg\_iov* and *msg\_iovlen* fields are used to specify where the received data shall be stored. The *msg\_iov* member points to an array of **iovec** structures; the *msg\_iovlen* member shall be set to the dimension of this array. In each **iovec** structure, the *iov\_base* field specifies a storage area and the *iov\_len* field gives its size in bytes. Each storage area indicated by *msg\_iov* is filled with received data in turn until all of the received data is stored or all of the areas have been filled.

Upon successful completion, the *msg\_flags* member of the message header shall be the bitwise-inclusive OR of all of the following flags that indicate conditions detected for the received message:

|            |                                                            |
|------------|------------------------------------------------------------|
| MSG_EOR    | End-of-record was received (if supported by the protocol). |
| MSG_OOB    | Out-of-band data was received.                             |
| MSG_TRUNC  | Normal data was truncated.                                 |
| MSG_CTRUNC | Control data was truncated.                                |

## RETURN VALUE

Upon successful completion, *recvmsg()* shall return the length of the message in bytes. If no messages are available to be received and the peer has performed an orderly shutdown, *recvmsg()* shall return 0. Otherwise, -1 shall be returned and *errno* set to indicate the error.

## ERRORS

The *recvmsg()* function shall fail if:

[EAGAIN] or [EWOULDBLOCK]

The socket's file descriptor is marked O\_NONBLOCK and no data is waiting to be received; or MSG\_OOB is set and no out-of-band data is available and either the socket's file descriptor is marked O\_NONBLOCK or the socket does not support blocking to await out-of-band data.

[EBADF] The *socket* argument is not a valid open file descriptor.

[ECONNRESET] A connection was forcibly closed by a peer.

[EINTR] This function was interrupted by a signal before any data was available.

[EINVAL] The sum of the *iov\_len* values overflows a **ssize\_t**, or the MSG\_OOB flag is set and no out-of-band data is available.

[EMSGSIZE] The *msg\_iovlen* member of the **msghdr** structure pointed to by *message* is less than or equal to 0, or is greater than {IOV\_MAX}.

[ENOTCONN] A receive is attempted on a connection-mode socket that is not connected.

[ENOTSOCK] The *socket* argument does not refer to a socket.

[EOPNOTSUPP] The specified flags are not supported for this socket type.

[ETIMEDOUT] The connection timed out during connection establishment, or due to a transmission timeout on active connection.

The *recvmsg()* function may fail if:

[EIO] An I/O error occurred while reading from or writing to the file system.

58035 [ENOBUFS] Insufficient resources were available in the system to perform the operation.

58036 [ENOMEM] Insufficient memory was available to fulfill the request.

#### 58037 EXAMPLES

58038 None.

#### 58039 APPLICATION USAGE

58040 The *select()* and *poll()* functions can be used to determine when data is available to be received.

#### 58041 RATIONALE

58042 None.

#### 58043 FUTURE DIRECTIONS

58044 None.

#### 58045 SEE ALSO

58046 *poll()*, *pselect()*, *recv()*, *recvfrom()*, *send()*, *sendmsg()*, *sendto()*, *shutdown()*, *socket()*

58047 XBD <sys/socket.h>

#### 58048 CHANGE HISTORY

58049 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

#### 58050 Issue 7

58051 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0504 [464] is applied.

**NAME**

regcomp, regerror, regex, regfree — regular expression matching

**SYNOPSIS**

```
#include <regex.h>

int regcomp(regex_t *restrict preg, const char *restrict pattern,
            int cflags);
size_t regerror(int errcode, const regex_t *restrict preg,
                char *restrict errbuf, size_t errbuf_size);
int regex(const regex_t *restrict preg, const char *restrict string,
           size_t nmatch, regmatch_t pmatch[restrict], int eflags);
void regfree(regex_t *preg);
```

**DESCRIPTION**

These functions interpret *basic* and *extended* regular expressions as described in XBD [Chapter 9](#) (on page 181).

The **regex\_t** structure is defined in **<regex.h>** and contains at least the following member:

| Member Type   | Member Name    | Description                             |
|---------------|----------------|-----------------------------------------|
| <b>size_t</b> | <i>re_nsub</i> | Number of parenthesized subexpressions. |

The **regmatch\_t** structure is defined in **<regex.h>** and contains at least the following members:

| Member Type     | Member Name  | Description                                                                                |
|-----------------|--------------|--------------------------------------------------------------------------------------------|
| <b>regoff_t</b> | <i>rm_so</i> | Byte offset from start of <i>string</i> to start of substring.                             |
| <b>regoff_t</b> | <i>rm_eo</i> | Byte offset from start of <i>string</i> of the first character after the end of substring. |

The *regcomp()* function shall compile the regular expression contained in the string pointed to by the *pattern* argument and place the results in the structure pointed to by *preg*. The *cflags* argument is the bitwise-inclusive OR of zero or more of the following flags, which are defined in the **<regex.h>** header:

REG\_EXTENDED     Use Extended Regular Expressions.

REG\_ICASE        Ignore case in match (see XBD [Chapter 9](#), on page 181).

REG\_NOSUB        Report only success/fail in *regex()*.

REG\_NEWLINE      Change the handling of <newline> characters, as described in the text.

The default regular expression type for *pattern* is a Basic Regular Expression. The application can specify Extended Regular Expressions using the REG\_EXTENDED *cflags* flag.

If the REG\_NOSUB flag was not set in *cflags*, then *regcomp()* shall set *re\_nsub* to the number of parenthesized subexpressions (delimited by "*\( \)*" in basic regular expressions or "*( )*" in extended regular expressions) found in *pattern*.

The *regex()* function compares the null-terminated string specified by *string* with the compiled regular expression *preg* initialized by a previous call to *regcomp()*. If it finds a match, *regex()* shall return 0; otherwise, it shall return non-zero indicating either no match or an error. The *eflags* argument is the bitwise-inclusive OR of zero or more of the following flags, which are defined in the **<regex.h>** header:

58094 REG\_NOTBOL The first character of the string pointed to by *string* is not the beginning of  
 58095 the line. Therefore, the <circumflex> character ('^'), when taken as a  
 58096 special character, shall not match the beginning of *string*.

58097 REG\_NOTEOL The last character of the string pointed to by *string* is not the end of the  
 58098 line. Therefore, the <dollar-sign> ('\$'), when taken as a special character,  
 58099 shall not match the end of *string*.

58100 If *nmatch* is 0 or REG\_NOSUB was set in the *cflags* argument to *regcomp()*, then *regexexec()* shall  
 58101 ignore the *pmatch* argument. Otherwise, the application shall ensure that the *pmatch* argument  
 58102 points to an array with at least *nmatch* elements, and *regexexec()* shall fill in the elements of that  
 58103 array with offsets of the substrings of *string* that correspond to the parenthesized subexpressions  
 58104 of *pattern*: *pmatch[i].rm\_so* shall be the byte offset of the beginning and *pmatch[i].rm\_eo* shall be  
 58105 one greater than the byte offset of the end of substring *i*. (Subexpression *i* begins at the *i*th  
 58106 matched open parenthesis, counting from 1.) Offsets in *pmatch*[0] identify the substring that  
 58107 corresponds to the entire regular expression. Unused elements of *pmatch* up to *pmatch*[*nmatch*-1]  
 58108 shall be filled with -1. If there are more than *nmatch* subexpressions in *pattern* (*pattern* itself  
 58109 counts as a subexpression), then *regexexec()* shall still do the match, but shall record only the first  
 58110 *nmatch* substrings.

58111 When matching a basic or extended regular expression, any given parenthesized subexpression  
 58112 of *pattern* might participate in the match of several different substrings of *string*, or it might not  
 58113 match any substring even though the pattern as a whole did match. The following rules shall be  
 58114 used to determine which substrings to report in *pmatch* when matching regular expressions:

- 58115 1. If subexpression *i* in a regular expression is not contained within another subexpression,  
 58116 and it participated in the match several times, then the byte offsets in *pmatch*[*i*] shall  
 58117 delimit the last such match.
- 58118 2. If subexpression *i* is not contained within another subexpression, and it did not  
 58119 participate in an otherwise successful match, the byte offsets in *pmatch*[*i*] shall be -1. A  
 58120 subexpression does not participate in the match when:  
 58121 ' \* ' or "\{\}" appears immediately after the subexpression in a basic regular  
 58122 expression, or ' \* ', ' ? ', or " { } " appears immediately after the subexpression in  
 58123 an extended regular expression, and the subexpression did not match (matched 0  
 58124 times)

58125 or:

58126 ' | ' is used in an extended regular expression to select this subexpression or  
 58127 another, and the other subexpression matched.

- 58128 3. If subexpression *i* is contained within another subexpression *j*, and *i* is not contained  
 58129 within any other subexpression that is contained within *j*, and a match of subexpression *j*  
 58130 is reported in *pmatch*[*j*], then the match or non-match of subexpression *i* reported in  
 58131 *pmatch*[*i*] shall be as described in 1. and 2. above, but within the substring reported in  
 58132 *pmatch*[*j*] rather than the whole string. The offsets in *pmatch*[*i*] are still relative to the start  
 58133 of *string*.
- 58134 4. If subexpression *i* is contained in subexpression *j*, and the byte offsets in *pmatch*[*j*] are -1,  
 58135 then the pointers in *pmatch*[*i*] shall also be -1.
- 58136 5. If subexpression *i* matched a zero-length string, then both byte offsets in *pmatch*[*i*] shall be  
 58137 the byte offset of the character or null terminator immediately following the zero-length  
 58138 string.



58139 If, when *regexec()* is called, the locale is different from when the regular expression was  
58140 compiled, the result is undefined.

58141 If REG\_NEWLINE is not set in *cflags*, then a <newline> in *pattern* or *string* shall be treated as an  
58142 ordinary character. If REG\_NEWLINE is set, then <newline> shall be treated as an ordinary  
58143 character except as follows:

- 58144 1. A <newline> in *string* shall not be matched by a <period> outside a bracket expression or  
58145 by any form of a non-matching list (see XBD Chapter 9, on page 181).
- 58146 2. A <circumflex> ( ' ^ ' ) in *pattern*, when used to specify expression anchoring (see XBD  
58147 Section 9.3.8, on page 188), shall match the zero-length string immediately after a  
58148 <newline> in *string*, regardless of the setting of REG\_NOTBOL.
- 58149 3. A <dollar-sign> ( ' \$ ' ) in *pattern*, when used to specify expression anchoring, shall match  
58150 the zero-length string immediately before a <newline> in *string*, regardless of the setting  
58151 of REG\_NOTEOL.

58152 The *regfree()* function frees any memory allocated by *regcomp()* associated with *preg*.

58153 The following constants are defined as the minimum set of error return values, although other  
58154 errors listed as implementation extensions in <regex.h> are possible:

|       |              |                                                                      |
|-------|--------------|----------------------------------------------------------------------|
| 58155 | REG_BADBR    | Content of "\{\}" invalid: not a number, number too large, more than |
| 58156 |              | two numbers, first larger than second.                               |
| 58157 | REG_BADPAT   | Invalid regular expression.                                          |
| 58158 | REG_BADRPT   | '?', '*', or '+' not preceded by valid regular expression.           |
| 58159 | REG_EBRACE   | "\{\}" imbalance.                                                    |
| 58160 | REG_EBRACK   | "[" imbalance.                                                       |
| 58161 | REG_ECOLLATE | Invalid collating element referenced.                                |
| 58162 | REG_ECTYPE   | Invalid character class type referenced.                             |
| 58163 | REG_EESCAPE  | Trailing <backslash> character in pattern.                           |
| 58164 | REG_EPAREN   | "\(" or "(" imbalance.                                               |
| 58165 | REG_ERANGE   | Invalid endpoint in range expression.                                |
| 58166 | REG_ESPACE   | Out of memory.                                                       |
| 58167 | REG_ESUBREG  | Number in "\digit" invalid or in error.                              |
| 58168 | REG_NOMATCH  | <i>regexec()</i> failed to match.                                    |

58169 If more than one error occurs in processing a function call, any one of the possible constants may  
58170 be returned, as the order of detection is unspecified.

58171 The *regerror()* function provides a mapping from error codes returned by *regcomp()* and  
58172 *regexec()* to unspecified printable strings. It generates a string corresponding to the value of the  
58173 *errcode* argument, which the application shall ensure is the last non-zero value returned by  
58174 *regcomp()* or *regexec()* with the given value of *preg*. If *errcode* is not such a value, the content of  
58175 the generated string is unspecified.

58176 If *preg* is a null pointer, but *errcode* is a value returned by a previous call to *regexec()* or *regcomp()*,  
58177 the *regerror()* still generates an error string corresponding to the value of *errcode*, but it might not  
58178 be as detailed under some implementations.



If the *errbuf\_size* argument is not 0, *regerror()* shall place the generated string into the buffer of size *errbuf\_size* bytes pointed to by *errbuf*. If the string (including the terminating null) cannot fit in the buffer, *regerror()* shall truncate the string and null-terminate the result.

If *errbuf\_size* is 0, *regerror()* shall ignore the *errbuf* argument, and return the size of the buffer needed to hold the generated string.

If the *preg* argument to *regexexec()* or *regfree()* is not a compiled regular expression returned by *regcomp()*, the result is undefined. A *preg* is no longer treated as a compiled regular expression after it is given to *regfree()*.

## RETURN VALUE

Upon successful completion, the *regcomp()* function shall return 0. Otherwise, it shall return an integer value indicating an error as described in <regex.h>, and the content of *preg* is undefined. If a code is returned, the interpretation shall be as given in <regex.h>.

If *regcomp()* detects an invalid RE, it may return REG\_BADPAT, or it may return one of the error codes that more precisely describes the error.

Upon successful completion, the *regexexec()* function shall return 0. Otherwise, it shall return REG\_NOMATCH to indicate no match.

Upon successful completion, the *regerror()* function shall return the number of bytes needed to hold the entire generated string, including the null termination. If the return value is greater than *errbuf\_size*, the string returned in the buffer pointed to by *errbuf* has been truncated.

The *regfree()* function shall not return a value.

## ERRORS

No errors are defined.

## EXAMPLES

```
#include <regex.h>

/*
 * Match string against the extended regular expression in
 * pattern, treating errors as no match.
 *
 * Return 1 for match, 0 for no match.
 */

int
match(const char *string, char *pattern)
{
    int    status;
    regex_t re;

    if (regcomp(&re, pattern, REG_EXTENDED|REG_NOSUB) != 0) {
        return(0);      /* Report error. */
    }
    status = regexexec(&re, string, (size_t) 0, NULL, 0);
    regfree(&re);
    if (status != 0) {
        return(0);      /* Report error. */
    }
    return(1);
}
```

The following demonstrates how the REG\_NOTBOL flag could be used with *regexexec()* to find all substrings in a line that match a pattern supplied by a user. (For simplicity of the example, very little error checking is done.)

```
(void) regcomp (&re, pattern, 0);
/* This call to regexexec() finds the first match on the line. */
error = regexexec (&re, &buffer[0], 1, &pm, 0);
while (error == 0) { /* While matches found. */
    /* Substring found between pm.rm_so and pm.rm_eo. */
    /* This call to regexexec() finds the next match. */
    error = regexexec (&re, buffer + pm.rm_eo, 1, &pm, REG_NOTBOL);
}
```

## APPLICATION USAGE

An application could use:

```
regerror(code, preg, (char *)NULL, (size_t)0)
```

to find out how big a buffer is needed for the generated string, *malloc()* a buffer to hold the string, and then call *regerror()* again to get the string. Alternatively, it could allocate a fixed, static buffer that is big enough to hold most strings, and then use *malloc()* to allocate a larger buffer if it finds that this is too small.

To match a pattern as described in XCU [Section 2.13](#) (on page 2372), use the *fnmatch()* function.

## RATIONALE

The *regexexec()* function must fill in all *nmatch* elements of *pmatch*, where *nmatch* and *pmatch* are supplied by the application, even if some elements of *pmatch* do not correspond to subexpressions in *pattern*. The application developer should note that there is probably no reason for using a value of *nmatch* that is larger than *preg->re\_nsub*+1.

The REG\_NEWLINE flag supports a use of RE matching that is needed in some applications like text editors. In such applications, the user supplies an RE asking the application to find a line that matches the given expression. An anchor in such an RE anchors at the beginning or end of any line. Such an application can pass a sequence of <newline>-separated lines to *regexexec()* as a single long string and specify REG\_NEWLINE to *regcomp()* to get the desired behavior. The application must ensure that there are no explicit <newline> characters in *pattern* if it wants to ensure that any match occurs entirely within a single line.

The REG\_NEWLINE flag affects the behavior of *regexexec()*, but it is in the *cflags* parameter to *regcomp()* to allow flexibility of implementation. Some implementations will want to generate the same compiled RE in *regcomp()* regardless of the setting of REG\_NEWLINE and have *regexexec()* handle anchors differently based on the setting of the flag. Other implementations will generate different compiled REs based on the REG\_NEWLINE.

The REG\_ICASE flag supports the operations taken by the *grep -i* option and the historical implementations of *ex* and *vi*. Including this flag will make it easier for application code to be written that does the same thing as these utilities.

The substrings reported in *pmatch[]* are defined using offsets from the start of the string rather than pointers. This allows type-safe access to both constant and non-constant strings.

The type **regoff\_t** is used for the elements of *pmatch[]* to ensure that the application can represent large arrays in memory (important for an application conforming to the Shell and Utilities volume of POSIX.1-2008).

The 1992 edition of this standard required **regoff\_t** to be at least as wide as **off\_t**, to facilitate future extensions in which the string to be searched is taken from a file. However, these future

extensions have not appeared. The requirement rules out popular implementations with 32-bit **regoff\_t** and 64-bit **off\_t**, so it has been removed.

The standard developers rejected the inclusion of a *regsub()* function that would be used to do substitutions for a matched RE. While such a routine would be useful to some applications, its utility would be much more limited than the matching function described here. Both RE parsing and substitution are possible to implement without support other than that required by the ISO C standard, but matching is much more complex than substituting. The only difficult part of substitution, given the information supplied by *regexec()*, is finding the next character in a string when there can be multi-byte characters. That is a much larger issue, and one that needs a more general solution.

The *errno* variable has not been used for error returns to avoid filling the *errno* name space for this feature.

The interface is defined so that the matched substrings *rm\_sp* and *rm\_ep* are in a separate **regmatch\_t** structure instead of in **regex\_t**. This allows a single compiled RE to be used simultaneously in several contexts; in *main()* and a signal handler, perhaps, or in multiple threads of lightweight processes. (The *preg* argument to *regexec()* is declared with type **const**, so the implementation is not permitted to use the structure to store intermediate results.) It also allows an application to request an arbitrary number of substrings from an RE. The number of subexpressions in the RE is reported in *re\_nsub* in *preg*. With this change to *regexec()*, consideration was given to dropping the REG\_NOSUB flag since the user can now specify this with a zero *nmatch* argument to *regexec()*. However, keeping REG\_NOSUB allows an implementation to use a different (perhaps more efficient) algorithm if it knows in *regcomp()* that no subexpressions need be reported. The implementation is only required to fill in *pmatch* if *nmatch* is not zero and if REG\_NOSUB is not specified. Note that the **size\_t** type, as defined in the ISO C standard, is unsigned, so the description of *regexec()* does not need to address negative values of *nmatch*.

REG\_NOTBOL was added to allow an application to do repeated searches for the same pattern in a line. If the pattern contains a <circumflex> character that should match the beginning of a line, then the pattern should only match when matched against the beginning of the line. Without the REG\_NOTBOL flag, the application could rewrite the expression for subsequent matches, but in the general case this would require parsing the expression. The need for REG\_NOTEOL is not as clear; it was added for symmetry.

The addition of the *regerror()* function addresses the historical need for conforming application programs to have access to error information more than “Function failed to compile/match your RE for unknown reasons”.

This interface provides for two different methods of dealing with error conditions. The specific error codes (REG\_EBRACE, for example), defined in <regex.h>, allow an application to recover from an error if it is so able. Many applications, especially those that use patterns supplied by a user, will not try to deal with specific error cases, but will just use *regerror()* to obtain a human-readable error message to present to the user.

The *regerror()* function uses a scheme similar to *confstr()* to deal with the problem of allocating memory to hold the generated string. The scheme used by *strerror()* in the ISO C standard was considered unacceptable since it creates difficulties for multi-threaded applications.

The *preg* argument is provided to *regerror()* to allow an implementation to generate a more descriptive message than would be possible with *errcode* alone. An implementation might, for example, save the character offset of the offending character of the pattern in a field of *preg*, and then include that in the generated message string. The implementation may also ignore *preg*.

A REG\_FILENAME flag was considered, but omitted. This flag caused *regexexec()* to match patterns as described in XCU Section 2.13 (on page 2372) instead of REs. This service is now provided by the *fnmatch()* function.

Notice that there is a difference in philosophy between the ISO POSIX-2:1993 standard and POSIX.1-2008 in how to handle a “bad” regular expression. The ISO POSIX-2:1993 standard says that many bad constructs “produce undefined results”, or that “the interpretation is undefined”. POSIX.1-2008, however, says that the interpretation of such REs is unspecified. The term “undefined” means that the action by the application is an error, of similar severity to passing a bad pointer to a function.

The *regcomp()* and *regexexec()* functions are required to accept any null-terminated string as the *pattern* argument. If the meaning of the string is “undefined”, the behavior of the function is “unspecified”. POSIX.1-2008 does not specify how the functions will interpret the pattern; they might return error codes, or they might do pattern matching in some completely unexpected way, but they should not do something like abort the process.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*fnmatch()*, *glob()*

XBD Chapter 9 (on page 181), *<regex.h>*, *<sys/types.h>*

XCUI Section 2.13 (on page 2372)

#### CHANGE HISTORY

First released in Issue 4. Derived from the ISO POSIX-2 standard.

##### Issue 5

Moved from POSIX2 C-language Binding to BASE.

##### Issue 6

In the SYNOPSIS, the optional include of the *<sys/types.h>* header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include *<sys/types.h>* has been removed. Although *<sys/types.h>* was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.

The normative text is updated to avoid use of the term “must” for application requirements.

The REG\_ENOSYS constant is removed.

The **restrict** keyword is added to the *regcomp()*, *regerror()*, and *regexexec()* prototypes for alignment with the ISO/IEC 9899:1999 standard.

##### Issue 7

Austin Group Interpretation 1003.1-2001 #134 is applied, clarifying that if more than one error occurs in processing a function call, any one of the possible constants may be returned.

SD5-XBD-ERN-60 is applied.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0505 [305] is applied.

58357 **NAME**

58358 remainder, remainderf, remainderl — remainder function

58359 **SYNOPSIS**

```
58360 #include <math.h>
58361 double remainder(double x, double y);
58362 float remainderf(float x, float y);
58363 long double remainderl(long double x, long double y);
```

58364 **DESCRIPTION**

58365 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 58366 conflict between the requirements described here and the ISO C standard is unintentional. This  
 58367 volume of POSIX.1-2008 defers to the ISO C standard.

58368 These functions shall return the floating-point remainder  $r = x - ny$  when  $y$  is non-zero. The value  
 58369  $n$  is the integral value nearest the exact value  $x/y$ . When  $|n - x/y| = 1/2$ , the value  $n$  is chosen to  
 58370 be even.

58371 The behavior of `remainder()` shall be independent of the rounding mode.

58372 **RETURN VALUE**

58373 Upon successful completion, these functions shall return the floating-point remainder  $r = x - ny$   
 58374 when  $y$  is non-zero.

58375 On systems that do not support the IEC 60559 Floating-Point option, if  $y$  is zero, it is  
 58376 implementation-defined whether a domain error occurs or zero is returned.

58377 MX If  $x$  or  $y$  is NaN, a NaN shall be returned.

58378 If  $x$  is infinite or  $y$  is 0 and the other is non-NaN, a domain error shall occur, and a NaN shall be  
 58379 returned.

58380 **ERRORS**

58381 These functions shall fail if:

58382 MX **Domain Error** The  $x$  argument is  $\pm\text{Inf}$ , or the  $y$  argument is  $\pm 0$  and the other argument is non-  
 58383 NaN.

58384 If the integer expression `(math_errhandling & MATH_ERRNO)` is non-zero,  
 58385 then `errno` shall be set to [EDOM]. If the integer expression `(math_errhandling`  
 58386 `& MATH_ERREXCEPT)` is non-zero, then the invalid floating-point exception  
 58387 shall be raised.

58388 These functions may fail if:

58389 **Domain Error** The  $y$  argument is zero.

58390 If the integer expression `(math_errhandling & MATH_ERRNO)` is non-zero,  
 58391 then `errno` shall be set to [EDOM]. If the integer expression `(math_errhandling`  
 58392 `& MATH_ERREXCEPT)` is non-zero, then the invalid floating-point exception  
 58393 shall be raised.

**EXAMPLES**

None.

**APPLICATION USAGE**

On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* & MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*abs()*, *div()*, *feclearexcept()*, *fetestexcept()*, *ldiv()*

XBD Section 4.20 (on page 117), <math.h>

**CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

**Issue 6**

The *remainder()* function is no longer marked as an extension.

The *remainderf()* and *remainderl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are revised to align with the ISO/IEC 9899:1999 standard.

IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are marked.

**Issue 7**

ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #55 (SD5-XSH-ERN-82) is applied.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0506 [320] is applied.

58421 **NAME**

58422       remove — remove a file

58423 **SYNOPSIS**

58424       #include &lt;stdio.h&gt;

58425       int remove(const char \*path);

58426 **DESCRIPTION**

58427 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 58428       conflict between the requirements described here and the ISO C standard is unintentional. This  
 58429       volume of POSIX.1-2008 defers to the ISO C standard.

58430       The *remove()* function shall cause the file named by the pathname pointed to by *path* to be no  
 58431       longer accessible by that name. A subsequent attempt to open that file using that name shall fail,  
 58432       unless it is created anew.

58433 CX       If *path* does not name a directory, *remove(path)* shall be equivalent to *unlink(path)*.

58434       If *path* names a directory, *remove(path)* shall be equivalent to *rmdir(path)*.

58435 **RETURN VALUE**58436 CX       Refer to *rmdir()* or *unlink()*.58437 **ERRORS**58438 CX       Refer to *rmdir()* or *unlink()*.58439 **EXAMPLES**58440       **Removing Access to a File**58441       The following example shows how to remove access to a file named */home/cnd/old\_mods*.

58442       #include &lt;stdio.h&gt;

58443       int status;

58444       ...

58445       status = remove("/home/cnd/old\_mods");

58446 **APPLICATION USAGE**

58447       None.

58448 **RATIONALE**

58449       None.

58450 **FUTURE DIRECTIONS**

58451       None.

58452 **SEE ALSO**58453       *rmdir()*, *unlink()*

58454       XBD &lt;stdio.h&gt;

58455 **CHANGE HISTORY**

58456       First released in Issue 3. Included for alignment with the POSIX.1-1988 standard and the ISO C  
 58457       standard.

58458 **Issue 6**

58459       Extensions beyond the ISO C standard are marked.



58460  
58461

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

58462  
58463  
58464

- The DESCRIPTION, RETURN VALUE, and ERRORS sections are updated so that if *path* is not a directory, *remove()* is equivalent to *unlink()*, and if it is a directory, it is equivalent to *rmdir()*.



58465 **NAME**

58466 remque — remove an element from a queue

58467 **SYNOPSIS**

```
58468 XSI      #include <search.h>
58469          void remque(void *element);
```

58470 **DESCRIPTION**58471 Refer to *insque()*.

58472 **NAME**

58473 remquo, remquof, remquol — remainder functions

58474 **SYNOPSIS**

```
58475 #include <math.h>
58476 double remquo(double x, double y, int *quo);
58477 float remquof(float x, float y, int *quo);
58478 long double remquol(long double x, long double y, int *quo);
```

58479 **DESCRIPTION**

58480 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 58481 conflict between the requirements described here and the ISO C standard is unintentional. This  
 58482 volume of POSIX.1-2008 defers to the ISO C standard.

58483 The *remquo()*, *remquof()*, and *remquol()* functions shall compute the same remainder as the  
 58484 *remainder()*, *remainderf()*, and *remainderl()* functions, respectively. In the object pointed to by *quo*,  
 58485 they store a value whose sign is the sign of  $x/y$  and whose magnitude is congruent modulo  $2^n$  to  
 58486 the magnitude of the integral quotient of  $x/y$ , where  $n$  is an implementation-defined integer  
 58487 greater than or equal to 3. If  $y$  is zero, the value stored in the object pointed to by *quo* is  
 58488 unspecified.

58489 An application wishing to check for error situations should set *errno* to zero and call  
 58490 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 58491 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 58492 zero, an error has occurred.

58493 **RETURN VALUE**58494 These functions shall return  $x \text{ REM } y$ .

58495 On systems that do not support the IEC 60559 Floating-Point option, if  $y$  is zero, it is  
 58496 implementation-defined whether a domain error occurs or zero is returned.

58497 MX If  $x$  or  $y$  is NaN, a NaN shall be returned.

58498 If  $x$  is  $\pm\text{Inf}$  or  $y$  is zero and the other argument is non-NaN, a domain error shall occur, and a  
 58499 NaN shall be returned.

58500 **ERRORS**

58501 These functions shall fail if:

58502 MX **Domain Error** The  $x$  argument is  $\pm\text{Inf}$ , or the  $y$  argument is  $\pm 0$  and the other argument is non-  
 58503 NaN.

58504 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 58505 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 58506 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 58507 shall be raised.

58508 These functions may fail if:

58509 **Domain Error** The  $y$  argument is zero.

58510 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 58511 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 58512 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 58513 shall be raised.

58514 **EXAMPLES**

58515 None.

58516 **APPLICATION USAGE**

58517 On error, the expressions `(math_errhandling & MATH_ERRNO)` and `(math_errhandling &`  
58518 `MATH_ERREXCEPT)` are independent of each other, but at least one of them must be non-zero.

58519 **RATIONALE**

58520 These functions are intended for implementing argument reductions which can exploit a few  
58521 low-order bits of the quotient. Note that  $x$  may be so large in magnitude relative to  $y$  that an  
58522 exact representation of the quotient is not practical.

58523 **FUTURE DIRECTIONS**

58524 None.

58525 **SEE ALSO**58526 [`feclearexcept\(\)`](#), [`fetestexcept\(\)`](#), [`remainder\(\)`](#)58527 XBD [Section 4.20](#) (on page 117), [`<math.h>`](#)58528 **CHANGE HISTORY**

58529 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

58530 **Issue 7**

58531 ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #56 (SD5-XSH-ERN-83) is applied.

58532 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0507 [320] is applied.

## 58533 NAME

58534 rename, renameat — rename file |

## 58535 SYNOPSIS

58536 #include &lt;stdio.h&gt;

58537 int rename(const char \*old, const char \*new);

58538 OH CX #include &lt;fcntl.h&gt; +

58539 +

58540 CX int renameat(int oldfd, const char \*old, int newfd,  
58541 const char \*new);

## 58542 DESCRIPTION

58543 CX For *rename()*: The functionality described on this reference page is aligned with the ISO C  
 58544 standard. Any conflict between the requirements described here and the ISO C standard is  
 58545 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

58546 The *rename()* function shall change the name of a file. The *old* argument points to the pathname  
 58547 CX of the file to be renamed. The *new* argument points to the new pathname of the file. If the *new*  
 58548 argument does not resolve to an existing directory entry for a file of type directory and the *new*  
 58549 argument contains at least one non-*<slash>* character and ends with one or more trailing  
 58550 *<slash>* characters after all symbolic links have been processed, *rename()* shall fail.

58551 If either the *old* or *new* argument names a symbolic link, *rename()* shall operate on the symbolic  
 58552 link itself, and shall not resolve the last component of the argument. If the *old* argument and the  
 58553 *new* argument resolve to either the same existing directory entry or different directory entries for  
 58554 the same existing file, *rename()* shall return successfully and perform no other action.

58555 If the *old* argument points to the pathname of a file that is not a directory, the *new* argument shall  
 58556 not point to the pathname of a directory. If the link named by the *new* argument exists, it shall be  
 58557 removed and *old* renamed to *new*. In this case, a link named *new* shall remain visible to other  
 58558 threads throughout the renaming operation and refer either to the file referred to by *new* or *old*  
 58559 before the operation began. Write access permission is required for both the directory containing  
 58560 *old* and the directory containing *new*.

58561 If the *old* argument points to the pathname of a directory, the *new* argument shall not point to the  
 58562 pathname of a file that is not a directory. If the directory named by the *new* argument exists, it  
 58563 shall be removed and *old* renamed to *new*. In this case, a link named *new* shall exist throughout  
 58564 the renaming operation and shall refer either to the directory referred to by *new* or *old* before the  
 58565 operation began. If *new* names an existing directory, it shall be required to be an empty directory.

58566 If either *pathname* argument refers to a path whose final component is either dot or dot-dot,  
 58567 *rename()* shall fail.

58568 If the *old* argument points to a pathname of a symbolic link, the symbolic link shall be renamed.  
 58569 If the *new* argument points to a pathname of a symbolic link, the symbolic link shall be removed.

58570 The *old* pathname shall not name an ancestor directory of the *new* pathname. Write access  
 58571 permission is required for the directory containing *old* and the directory containing *new*. If the  
 58572 *old* argument points to the pathname of a directory, write access permission may be required for  
 58573 the directory named by *old*, and, if it exists, the directory named by *new*.

58574 If the link named by the *new* argument exists and the file's link count becomes 0 when it is  
 58575 removed and no process has the file open, the space occupied by the file shall be freed and the  
 58576 file shall no longer be accessible. If one or more processes have the file open when the last link is  
 58577 removed, the link shall be removed before *rename()* returns, but the removal of the file contents

58578 shall be postponed until all references to the file are closed.

58579 Upon successful completion, *rename()* shall mark for update the last data modification and last  
58580 file status change timestamps of the parent directory of each file.

58581 If the *rename()* function fails for any reason other than [EIO], any file named by *new* shall be  
58582 unaffected.

58583 The *renameat()* function shall be equivalent to the *rename()* function except in the case where  
58584 either *old* or *new* specifies a relative path. If *old* is a relative path, the file to be renamed is located  
58585 relative to the directory associated with the file descriptor *oldfd* instead of the current working  
58586 directory. If *new* is a relative path, the same happens only relative to the directory associated  
58587 with *newfd*. If the access mode of the open file description associated with the file descriptor is  
58588 not O\_SEARCH, the function shall check whether directory searches are permitted using the  
58589 current permissions of the directory underlying the file descriptor. If the access mode is  
58590 O\_SEARCH, the function shall not perform the check.

58591 If *renameat()* is passed the special value AT\_FDCWD in the *oldfd* or *newfd* parameter, the current  
58592 working directory shall be used in the determination of the file for the respective *path* parameter.

## 58593 RETURN VALUE

58594 CX Upon successful completion, the *rename()* function shall return 0. Otherwise, it shall return -1,  
58595 *errno* shall be set to indicate the error, and neither the file named by *old* nor the file named by  
58596 *new* shall be changed or created.

58597 CX Upon successful completion, the *renameat()* function shall return 0. Otherwise, it shall return -1  
58598 and set *errno* to indicate the error.

## 58599 ERRORS

58600 CX The *rename()* and *renameat()* functions shall fail if:

58601 CX [EACCES] A component of either path prefix denies search permission; or one of the  
58602 directories containing *old* or *new* denies write permissions; or, write  
58603 permission is required and is denied for a directory pointed to by the *old* or  
58604 *new* arguments.

58605 CX [EBUSY] The directory named by *old* or *new* is currently in use by the system or another  
58606 process, and the implementation considers this an error.

58607 CX [EEXIST] or [ENOTEMPTY]  
58608 The link named by *new* is a directory that is not an empty directory.

58609 CX [EINVAL] The *old* pathname names an ancestor directory of the *new* pathname, or either  
58610 pathname argument contains a final component that is dot or dot-dot.

58611 CX [EIO] A physical I/O error has occurred.

58612 CX [EISDIR] The *new* argument points to a directory and the *old* argument points to a file  
58613 that is not a directory.

58614 CX [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*  
58615 argument.

58616 CX [EMLINK] The file named by *old* is a directory, and the link count of the parent directory  
58617 of *new* would exceed {LINK\_MAX}.

58618 CX [ENAMETOOLONG]  
58619 The length of a component of a pathname is longer than {NAME\_MAX}.

|       |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|-------|--------|------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 58620 | CX     | [ENOENT]                                                         | The link named by <i>old</i> does not name an existing file, a component of the path prefix of <i>new</i> does not exist, or either <i>old</i> or <i>new</i> points to an empty string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 58621 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58622 | CX     | [ENOSPC]                                                         | The directory that would contain <i>new</i> cannot be extended.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58623 | CX     | [ENOTDIR]                                                        | A component of either path prefix names an existing file that is neither a directory nor a symbolic link to a directory; or the <i>old</i> argument names a directory and the <i>new</i> argument names a non-directory file; or the <i>old</i> argument contains at least one non- <code>&lt;slash&gt;</code> character and ends with one or more trailing <code>&lt;slash&gt;</code> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory; or the <i>old</i> argument names an existing non-directory file and the <i>new</i> argument names a nonexistent file, contains at least one non- <code>&lt;slash&gt;</code> character, and ends with one or more trailing <code>&lt;slash&gt;</code> characters; or the <i>new</i> argument names an existing non-directory file, contains at least one non- <code>&lt;slash&gt;</code> character, and ends with one or more trailing <code>&lt;slash&gt;</code> characters. |
| 58624 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58625 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58626 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58627 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58628 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58629 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58630 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58631 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58632 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58633 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58634 | XSI    | [EPERM] or [EACCES]                                              | The <code>S_ISVTX</code> flag is set on the directory containing the file referred to by <i>old</i> and the process does not satisfy the criteria specified in XBD Section 4.3 (on page 108) with respect to <i>old</i> ; or <i>new</i> refers to an existing file, the <code>S_ISVTX</code> flag is set on the directory containing this file, and the process does not satisfy the criteria specified in XBD Section 4.3 with respect to this file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 58635 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58636 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58637 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58638 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58639 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58640 | CX     | [EROFS]                                                          | The requested operation requires writing in a directory on a read-only file system.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 58641 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58642 | CX     | [EXDEV]                                                          | The links named by <i>new</i> and <i>old</i> are on different file systems and the implementation does not support links between file systems.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 58643 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58644 | CX     | In addition, the <i>renameat()</i> function shall fail if:       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58645 |        | [EACCES]                                                         | The access mode of the open file description associated with <i>olddfd</i> or <i>newfd</i> is not <code>O_SEARCH</code> and the permissions of the directory underlying <i>olddfd</i> or <i>newfd</i> , respectively, do not permit directory searches.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 58646 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58647 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58648 |        | [EBADF]                                                          | The <i>old</i> argument does not specify an absolute path and the <i>olddfd</i> argument is neither <code>AT_FDCWD</code> nor a valid file descriptor open for reading or searching, or the <i>new</i> argument does not specify an absolute path and the <i>newfd</i> argument is neither <code>AT_FDCWD</code> nor a valid file descriptor open for reading or searching.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 58649 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58650 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58651 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58652 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58653 |        | [ENOTDIR]                                                        | The <i>old</i> or <i>new</i> argument is not an absolute path and <i>olddfd</i> or <i>newfd</i> , respectively, is a file descriptor associated with a non-directory file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 58654 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58655 | CX     | The <i>rename()</i> and <i>renameat()</i> functions may fail if: |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58656 | OB XSR | [EBUSY]                                                          | The file named by the <i>old</i> or <i>new</i> arguments is a named STREAM.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 58657 | CX     | [ELOOP]                                                          | More than <code>{SYMLOOP_MAX}</code> symbolic links were encountered during resolution of the <i>path</i> argument.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 58658 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58659 | CX     | [ENAMETOOLONG]                                                   | The length of a pathname exceeds <code>{PATH_MAX}</code> , or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds <code>{PATH_MAX}</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 58660 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58661 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 58662 |        |                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

58663 CX [ETXTBSY] The file named by *new* exists and is the last directory entry to a pure procedure  
 58664 (shared text) file that is being executed.

## 58665 EXAMPLES

### 58666 Renaming a File

58667 The following example shows how to rename a file named `/home/cnd/mod1` to  
 58668 `/home/cnd/mod2`.

```
58669 #include <stdio.h>
58670
58671 int status;
58672 ...
58673 status = rename("/home/cnd/mod1", "/home/cnd/mod2");
```

## 58673 APPLICATION USAGE

58674 Some implementations mark for update the last file status change timestamp of renamed files  
 58675 and some do not. Applications which make use of the last file status change timestamp may  
 58676 behave differently with respect to renamed files unless they are designed to allow for either  
 58677 behavior.

## 58678 RATIONALE

58679 This *rename()* function is equivalent for regular files to that defined by the ISO C standard. Its  
 58680 inclusion here expands that definition to include actions on directories and specifies behavior  
 58681 when the *new* parameter names a file that already exists. That specification requires that the  
 58682 action of the function be atomic.

58683 One of the reasons for introducing this function was to have a means of renaming directories  
 58684 while permitting implementations to prohibit the use of *link()* and *unlink()* with directories,  
 58685 thus constraining links to directories to those made by *mkdir()*.

58686 The specification that if *old* and *new* refer to the same file is intended to guarantee that:

```
58687 rename("x", "x");
```

58688 does not remove the file.

58689 Renaming dot or dot-dot is prohibited in order to prevent cyclical file system paths.

58690 See also the descriptions of [ENOTEMPTY] and [ENAMETOOLONG] in *rmdir()* and [EBUSY] in  
 58691 *unlink()*. For a discussion of [EXDEV], see *link()*.

58692 The purpose of the *renameat()* function is to rename files in directories other than the current  
 58693 working directory without exposure to race conditions. Any part of the path of a file could be  
 58694 changed in parallel to a call to *rename()*, resulting in unspecified behavior. By opening file  
 58695 descriptors for the source and target directories and using the *renameat()* function it can be  
 58696 guaranteed that that renamed file is located correctly and the resulting file is in the desired  
 58697 directory.

## 58698 FUTURE DIRECTIONS

58699 None.

## 58700 SEE ALSO

58701 *link()*, *rmdir()*, *symlink()*, *unlink()*

58702 XBD Section 4.3 (on page 108), `<fcntl.h>`, `<stdio.h>`



**CHANGE HISTORY**

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

**Issue 5**

The [EBUSY] error is added to the optional part of the ERRORS section.

**Issue 6**

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The [EIO] mandatory error condition is added.
- The [ELOOP] mandatory error condition is added.
- A second [ENAMETOOLONG] is added as an optional error condition.
- The [ETXTBSY] optional error condition is added.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Details are added regarding the treatment of symbolic links.
- The [ELOOP] optional error condition is added.

The normative text is updated to avoid use of the term “must” for application requirements.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #016 is applied, changing the definition of the [ENOTDIR] error.

Austin Group Interpretation 1003.1-2001 #076 is applied, clarifying the behavior if the final component of a path is either dot or dot-dot, and adding the associated [EINVAL] error case.

Austin Group Interpretation 1003.1-2001 #143 is applied.

Austin Group Interpretation 1003.1-2001 #145 is applied, clarifying that the [ENOENT] error condition also applies to the case in which a component of *new* does not exist.

Austin Group Interpretations 1003.1-2001 #174 and #181 are applied.

The *renameat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Changes are made related to support for finegrained timestamps.

Changes are made to allow a directory to be opened for searching.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0508 [324], XSH/TC1-2008/0509 [147], XSH/TC1-2008/0510 [379], XSH/TC1-2008/0511 [278], and XSH/TC1-2008/0512 [278] are applied.



58735 **NAME**

58736       rewind — reset the file position indicator in a stream

58737 **SYNOPSIS**

58738       #include &lt;stdio.h&gt;

58739       void rewind(FILE \*stream);

58740 **DESCRIPTION**

58741 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 58742       conflict between the requirements described here and the ISO C standard is unintentional. This  
 58743       volume of POSIX.1-2008 defers to the ISO C standard.

58744       The call:

58745       rewind(stream)

58746       shall be equivalent to:

58747       (void) fseek(stream, 0L, SEEK\_SET)

58748       except that *rewind()* shall also clear the error indicator.

58749 CX       Since *rewind()* does not return a value, an application wishing to detect errors should clear *errno*,  
 58750       then call *rewind()*, and if *errno* is non-zero, assume an error has occurred.

58751 **RETURN VALUE**58752       The *rewind()* function shall not return a value.58753 **ERRORS**58754 CX       Refer to *fseek()* with the exception of [EINVAL] which does not apply.58755 **EXAMPLES**

58756       None.

58757 **APPLICATION USAGE**

58758       None.

58759 **RATIONALE**

58760       None.

58761 **FUTURE DIRECTIONS**

58762       None.

58763 **SEE ALSO**58764       Section 2.5 (on page 495), *fseek()*

58765       XBD &lt;stdio.h&gt;

58766 **CHANGE HISTORY**

58767       First released in Issue 1. Derived from Issue 1 of the SVID.

58768 **Issue 6**

58769       Extensions beyond the ISO C standard are marked.

58770 **Issue 7**

58771       POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0513 [14] is applied.

58772 **NAME**

58773       rewinddir — reset the position of a directory stream to the beginning of a directory

58774 **SYNOPSIS**

58775       #include <dirent.h>

58776       void rewinddir(DIR \*dirp);

58777 **DESCRIPTION**

58778       The *rewinddir()* function shall reset the position of the directory stream to which *dirp* refers to the  
58779       beginning of the directory. It shall also cause the directory stream to refer to the current state of  
58780       the corresponding directory, as a call to *opendir()* would have done. If *dirp* does not refer to a  
58781       directory stream, the effect is undefined.

58782       After a call to the *fork()* function, either the parent or child (but not both) may continue  
58783       XSI       processing the directory stream using *readdir()*, *rewinddir()*, or *seekdir()*. If both the parent and  
58784       child processes use these functions, the result is undefined.

58785 **RETURN VALUE**

58786       The *rewinddir()* function shall not return a value.

58787 **ERRORS**

58788       No errors are defined.

58789 **EXAMPLES**

58790       None.

58791 **APPLICATION USAGE**

58792       The *rewinddir()* function should be used in conjunction with *opendir()*, *readdir()*, and *closedir()* to  
58793       examine the contents of the directory. This method is recommended for portability.

58794 **RATIONALE**

58795       None.

58796 **FUTURE DIRECTIONS**

58797       None.

58798 **SEE ALSO**

58799       *closedir()*, *fdopendir()*, *readdir()*

58800       XBD <dirent.h>, <sys/types.h>

58801 **CHANGE HISTORY**

58802       First released in Issue 2.

58803 **Issue 6**

58804       In the SYNOPSIS, the optional include of the <sys/types.h> header is removed.

58805       The following new requirements on POSIX implementations derive from alignment with the  
58806       Single UNIX Specification:

- 58807       • The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was  
58808       required for conforming implementations of previous POSIX specifications, it was not  
58809       required for UNIX applications.

58810 **NAME**

58811 rint, rintf, rintl — round-to-nearest integral value

58812 **SYNOPSIS**

```
58813 #include <math.h>
58814 double rint(double x);
58815 float rintf(float x);
58816 long double rintl(long double x);
```

58817 **DESCRIPTION**

58818 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 58819 conflict between the requirements described here and the ISO C standard is unintentional. This  
 58820 volume of POSIX.1-2008 defers to the ISO C standard.

58821 These functions shall return the integral value (represented as a **double**) nearest  $x$  in the  
 58822 direction of the current rounding mode. The current rounding mode is implementation-defined.

58823 If the current rounding mode rounds toward negative infinity, then *rint()* shall be equivalent to  
 58824 *floor()*. If the current rounding mode rounds toward positive infinity, then *rint()* shall be  
 58825 equivalent to *ceil()*. If the current rounding mode rounds towards zero, then *rint()* shall be  
 58826 MX equivalent to *trunc()*. If the current rounding mode rounds towards nearest, then *rint()* differs  
 58827 from *round()* in that halfway cases are rounded to even rather than away from zero.

58828 These functions differ from the *nearbyint()*, *nearbyintf()*, and *nearbyintl()* functions only in that  
 58829 they may raise the inexact floating-point exception if the result differs in value from the  
 58830 argument.

58831 An application wishing to check for error situations should set *errno* to zero and call  
 58832 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 58833 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 58834 zero, an error has occurred.

58835 **RETURN VALUE**

58836 Upon successful completion, these functions shall return the integer (represented as a double  
 58837 MX precision number) nearest  $x$  in the direction of the current rounding mode. The result shall have  
 58838 the same sign as  $x$ .

58839 MX If  $x$  is NaN, a NaN shall be returned.

58840 If  $x$  is  $\pm 0$  or  $\pm \text{Inf}$ ,  $x$  shall be returned.

58841 **ERRORS**

58842 No errors are defined.

58843 **EXAMPLES**

58844 None.

58845 **APPLICATION USAGE**

58846 The integral value returned by these functions need not be expressible as an **intmax\_t**. The  
 58847 return value should be tested before assigning it to an integer type to avoid the undefined  
 58848 results of an integer overflow.

58849 **RATIONALE**

58850 None.

58851 **FUTURE DIRECTIONS**

58852 None.

58853 **SEE ALSO**58854 *abs()*, *ceil()*, *feclearexcept()*, *fetestexcept()*, *floor()*, *isnan()*, *nearbyint()*58855 XBD [Section 4.20](#) (on page 117), **<math.h>**58856 **CHANGE HISTORY**

58857 First released in Issue 4, Version 2.

58858 **Issue 5**

58859 Moved from X/OPEN UNIX extension to BASE.

58860 **Issue 6**

58861 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 58862 • The *rintf()* and *rintl()* functions are added.
- 58863 • The *rint()* function is no longer marked as an extension.
- 58864 • The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are
- 58865 revised to align with the ISO/IEC 9899:1999 standard.
- 58866 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard
- 58867 are marked.

58868 **Issue 7**

58869 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0514 [346], XSH/TC1-2008/0515 [346],  
58870 XSH/TC1-2008/0516 [346], XSH/TC1-2008/0517 [346], and XSH/TC1-2008/0518 [346] are  
58871 applied.

58872 **NAME**

58873       rmdir — remove a directory

58874 **SYNOPSIS**

58875       #include &lt;unistd.h&gt;

58876       int rmdir(const char \*path);

58877 **DESCRIPTION**58878       The *rmdir()* function shall remove a directory whose name is given by *path*. The directory shall  
58879       be removed only if it is an empty directory.58880       If the directory is the root directory or the current working directory of any process, it is  
58881       unspecified whether the function succeeds, or whether it shall fail and set *errno* to [EBUSY].58882       If *path* names a symbolic link, then *rmdir()* shall fail and set *errno* to [ENOTDIR].58883       If the *path* argument refers to a path whose final component is either dot or dot-dot, *rmdir()* shall  
58884       fail.58885       If the directory's link count becomes 0 and no process has the directory open, the space occupied  
58886       by the directory shall be freed and the directory shall no longer be accessible. If one or more  
58887       processes have the directory open when the last link is removed, the dot and dot-dot entries, if  
58888       present, shall be removed before *rmdir()* returns and no new entries may be created in the  
58889       directory, but the directory shall not be removed until all references to the directory are closed.58890       If the directory is not an empty directory, *rmdir()* shall fail and set *errno* to [EEXIST] or  
58891       [ENOTEMPTY].58892       Upon successful completion, *rmdir()* shall mark for update the last data modification and last  
58893       file status change timestamps of the parent directory.58894 **RETURN VALUE**58895       Upon successful completion, the function *rmdir()* shall return 0. Otherwise, -1 shall be returned,  
58896       and *errno* set to indicate the error. If -1 is returned, the named directory shall not be changed.58897 **ERRORS**58898       The *rmdir()* function shall fail if:58899       [EACCES]       Search permission is denied on a component of the path prefix, or write  
58900       permission is denied on the parent directory of the directory to be removed.58901       [EBUSY]       The directory to be removed is currently in use by the system or some process  
58902       and the implementation considers this to be an error.58903       [EEXIST] or [ENOTEMPTY]       The *path* argument names a directory that is not an empty directory, or there  
58904       are hard links to the directory other than dot or a single entry in dot-dot.  
5890558906       [EINVAL]       The *path* argument contains a last component that is dot.

58907       [EIO]       A physical I/O error has occurred.

58908       [ELOOP]       A loop exists in symbolic links encountered during resolution of the *path*  
58909       argument.58910       [ENAMETOOLONG]       The length of a component of a pathname is longer than {NAME\_MAX}.  
5891158912       [ENOENT]       A component of *path* does not name an existing file, or the *path* argument  
58913       names a nonexistent directory or points to an empty string.

58914 [ENOTDIR] A component of *path* names an existing file that is neither a directory nor a  
 58915 symbolic link to a directory.

58916 XSI [EPERM] or [EACCES]  
 58917 The S\_ISVTX flag is set on the directory containing the file referred to by the  
 58918 *path* argument and the process does not satisfy the criteria specified in XBD  
 58919 Section 4.3 (on page 108).

58920 [EROFS] The directory entry to be removed resides on a read-only file system.

58921 The *rmdir()* function may fail if:

58922 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
 58923 resolution of the *path* argument.

58924 [ENAMETOOLONG]  
 58925 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
 58926 symbolic link produced an intermediate result with a length that exceeds  
 58927 {PATH\_MAX}.

## 58928 EXAMPLES

### 58929 Removing a Directory

58930 The following example shows how to remove a directory named `/home/cnd/mod1`.

```
58931 #include <unistd.h>
58932 int status;
58933 ...
58934 status = rmdir("/home/cnd/mod1");
```

## 58935 APPLICATION USAGE

58936 None.

## 58937 RATIONALE

58938 The *rmdir()* and *rename()* functions originated in 4.2 BSD, and they used [ENOTEMPTY] for the  
 58939 condition when the directory to be removed does not exist or *new* already exists. When the 1984  
 58940 /usr/group standard was published, it contained [EEXIST] instead. When these functions were  
 58941 adopted into System V, the 1984 /usr/group standard was used as a reference. Therefore,  
 58942 several existing applications and implementations support/use both forms, and no agreement  
 58943 could be reached on either value. All implementations are required to supply both [EEXIST] and  
 58944 [ENOTEMPTY] in `<errno.h>` with distinct values, so that applications can use both values in C-  
 58945 language **case** statements.

58946 The meaning of deleting *pathname/dot* is unclear, because the name of the file (directory) in the  
 58947 parent directory to be removed is not clear, particularly in the presence of multiple links to a  
 58948 directory.

58949 The POSIX.1-1990 standard was silent with regard to the behavior of *rmdir()* when there are  
 58950 multiple hard links to the directory being removed. The requirement to set *errno* to [EEXIST] or  
 58951 [ENOTEMPTY] clarifies the behavior in this case.

58952 If the current working directory of the process is being removed, that should be an allowed  
 58953 error.

58954 Virtually all existing implementations detect [ENOTEMPTY] or the case of dot-dot. The text in  
 58955 Section 2.3 (on page 481) about returning any one of the possible errors permits that behavior to  
 58956 continue. The [ELOOP] error may be returned if more than {SYMLOOP\_MAX} symbolic links

58957 are encountered during resolution of the *path* argument.

#### 58958 FUTURE DIRECTIONS

58959 None.

#### 58960 SEE ALSO

58961 [Section 2.3](#) (on page 481), *mkdir()*, *remove()*, *rename()*, *unlink()*

58962 [XBD Section 4.3](#) (on page 108), [<unistd.h>](#)

#### 58963 CHANGE HISTORY

58964 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

#### 58965 Issue 6

58966 The following new requirements on POSIX implementations derive from alignment with the  
58967 Single UNIX Specification:

- 58968 • The DESCRIPTION is updated to indicate the results of naming a symbolic link in *path*.
- 58969 • The [EIO] mandatory error condition is added.
- 58970 • The [ELOOP] mandatory error condition is added.
- 58971 • A second [ENAMETOOLONG] is added as an optional error condition.

58972 The following changes were made to align with the IEEE P1003.1a draft standard:

- 58973 • The [ELOOP] optional error condition is added.

#### 58974 Issue 7

58975 Austin Group Interpretation 1003.1-2001 #143 is applied.

58976 Austin Group Interpretation 1003.1-2001 #181 is applied, updating the requirements for  
58977 operations when the S\_ISVTX bit is set.

58978 Changes are made related to support for finegrained timestamps.

58979 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0519 [324] is applied.

58980 **NAME**

58981 round, roundf, roundl — round to the nearest integer value in a floating-point format

58982 **SYNOPSIS**

58983 #include <math.h>  
58984 double round(double x);  
58985 float roundf(float x);  
58986 long double roundl(long double x);

58987 **DESCRIPTION**

58988 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
58989 conflict between the requirements described here and the ISO C standard is unintentional. This  
58990 volume of POSIX.1-2008 defers to the ISO C standard.

58991 These functions shall round their argument to the nearest integer value in floating-point format,  
58992 rounding halfway cases away from zero, regardless of the current rounding direction.

58993 **RETURN VALUE**

58994 MX Upon successful completion, these functions shall return the rounded integer value. The result  
58995 shall have the same sign as *x*.

58996 MX If *x* is NaN, a NaN shall be returned.

58997 If *x* is  $\pm 0$  or  $\pm \text{Inf}$ , *x* shall be returned.

58998 **ERRORS**

58999 No errors are defined.

59000 **EXAMPLES**

59001 None.

59002 **APPLICATION USAGE**

59003 The integral value returned by these functions need not be expressible as an **intmax\_t**. The  
59004 return value should be tested before assigning it to an integer type to avoid the undefined  
59005 results of an integer overflow.

59006 These functions may raise the inexact floating-point exception if the result differs in value from  
59007 the argument.

59008 **RATIONALE**

59009 None.

59010 **FUTURE DIRECTIONS**

59011 None.

59012 **SEE ALSO**

59013 *feclearexcept()*, *fetestexcept()*

59014 XBD Section 4.20 (on page 117), **<math.h>**

59015 **CHANGE HISTORY**

59016 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

59017 **Issue 7**

59018 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0520 [346] is applied.



59019 **NAME**

59020 scalbln, scalblnf, scalblnl, scalbn, scalbnf, scalbnl — compute exponent using FLT\_RADIX

59021 **SYNOPSIS**

```
59022 #include <math.h>
59023 double scalbln(double x, long n);
59024 float scalblnf(float x, long n);
59025 long double scalblnl(long double x, long n);
59026 double scalbn(double x, int n);
59027 float scalbnf(float x, int n);
59028 long double scalbnl(long double x, int n);
```

59029 **DESCRIPTION**

59030 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 59031 conflict between the requirements described here and the ISO C standard is unintentional. This  
 59032 volume of POSIX.1-2008 defers to the ISO C standard.

59033 These functions shall compute  $x * \text{FLT\_RADIX}^n$  efficiently, not normally by computing  
 59034  $\text{FLT\_RADIX}^n$  explicitly.

59035 An application wishing to check for error situations should set *errno* to zero and call  
 59036 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 59037 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 59038 zero, an error has occurred.

59039 **RETURN VALUE**

59040 Upon successful completion, these functions shall return  $x * \text{FLT\_RADIX}^n$ .

59041 If the result would cause overflow, a range error shall occur and these functions shall return  
 59042  $\pm\text{HUGE\_VAL}$ ,  $\pm\text{HUGE\_VALF}$ , and  $\pm\text{HUGE\_VALL}$  (according to the sign of *x*) as appropriate for  
 59043 the return type of the function.

59044 MXX If the correct value would cause underflow, and is not representable, a range error may occur,  
 59045 MXX and *scalbln*(), *scalblnf*(), *scalblnl*(), *scalbn*(), *scalbnf*(), and *scalbnl*() shall return 0.0, or (if IEC  
 59046 60559 Floating-Point is not supported) an implementation-defined value no greater in  
 59047 magnitude than DBL\_MIN, FLT\_MIN, LDBL\_MIN, DBL\_MIN, FLT\_MIN, and LDBL\_MIN,  
 59048 respectively.

59049 MX If *x* is NaN, a NaN shall be returned.

59050 If *x* is  $\pm 0$  or  $\pm\text{Inf}$ , *x* shall be returned.

59051 If *n* is 0, *x* shall be returned.

59052 MXX If the correct value would cause underflow, and is representable, a range error may occur and  
 59053 the correct value shall be returned.

59054 **ERRORS**

59055 These functions shall fail if:

59056 Range Error The result overflows.

59057 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 59058 then *errno* shall be set to [ERANGE]. If the integer expression  
 59059 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the overflow  
 59060 floating-point exception shall be raised.

59061 These functions may fail if:

59062 Range Error The result underflows.

59063 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
59064 then *errno* shall be set to [ERANGE]. If the integer expression  
59065 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
59066 floating-point exception shall be raised.

#### 59067 EXAMPLES

59068 None.

#### 59069 APPLICATION USAGE

59070 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
59071 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

#### 59072 RATIONALE

59073 These functions are named so as to avoid conflicting with the historical definition of the *scalb()*  
59074 function from the Single UNIX Specification. The difference is that the *scalb()* function has a  
59075 second argument of **double** instead of **int**. The *scalb()* function is not part of the ISO C standard.  
59076 The three functions whose second type is **long** are provided because the factor required to scale  
59077 from the smallest positive floating-point value to the largest finite one, on many  
59078 implementations, is too large to represent in the minimum-width **int** format.

#### 59079 FUTURE DIRECTIONS

59080 None.

#### 59081 SEE ALSO

59082 *feclearexcept()*, *fetestexcept()*

59083 XBD Section 4.20 (on page 117), **<math.h>**

#### 59084 CHANGE HISTORY

59085 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

#### 59086 Issue 7

59087 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0521 [68] and XSH/TC1-2008/0522  
59088 [68] are applied.

59089 **NAME**

59090       scandir — scan a directory

59091 **SYNOPSIS**

59092       #include &lt;dirent.h&gt;

```
59093       int scandir(const char *dir, struct dirent ***namelist,  
59094                   int (*sel)(const struct dirent *),  
59095                   int (*compar)(const struct dirent **, const struct dirent **));
```

59096 **DESCRIPTION**59097       Refer to *alphasort()*.

59098 **NAME**

59099       scanf — convert formatted input

59100 **SYNOPSIS**

59101       #include &lt;stdio.h&gt;

59102       int scanf(const char \*restrict *format*, ...);59103 **DESCRIPTION**59104       Refer to *fscanf()*.

**59105 NAME**

59106 sched\_get\_priority\_max, sched\_get\_priority\_min — get priority limits (**REALTIME**)

**59107 SYNOPSIS**

```
59108 PS|TPS #include <sched.h>
59109 int sched_get_priority_max(int policy);
59110 int sched_get_priority_min(int policy);
```

**59111 DESCRIPTION**

59112 The *sched\_get\_priority\_max()* and *sched\_get\_priority\_min()* functions shall return the appropriate  
59113 maximum or minimum, respectively, for the scheduling policy specified by *policy*.

59114 The value of *policy* shall be one of the scheduling policy values defined in **<sched.h>**.

**59115 RETURN VALUE**

59116 If successful, the *sched\_get\_priority\_max()* and *sched\_get\_priority\_min()* functions shall return the  
59117 appropriate maximum or minimum values, respectively. If unsuccessful, they shall return a  
59118 value of  $-1$  and set *errno* to indicate the error.

**59119 ERRORS**

59120 The *sched\_get\_priority\_max()* and *sched\_get\_priority\_min()* functions shall fail if:

59121 [EINVAL] The value of the *policy* parameter does not represent a defined scheduling  
59122 policy.

**59123 EXAMPLES**

59124 None.

**59125 APPLICATION USAGE**

59126 None.

**59127 RATIONALE**

59128 None.

**59129 FUTURE DIRECTIONS**

59130 None.

**59131 SEE ALSO**

59132 *sched\_getparam()*, *sched\_setparam()*, *sched\_getscheduler()*, *sched\_rr\_get\_interval()*,  
59133 *sched\_setscheduler()*

59134 XBD **<sched.h>**

**59135 CHANGE HISTORY**

59136 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**59137 Issue 6**

59138 These functions are marked as part of the Process Scheduling option.

59139 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
59140 implementation does not support the Process Scheduling option.

59141 The [ESRCH] error condition has been removed since these functions do not take a *pid*  
59142 argument.

59143 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/52 is applied, changing the PS margin  
59144 code in the SYNOPSIS to PS|TPS.

59145 **NAME**59146 sched\_getparam — get scheduling parameters (**REALTIME**)59147 **SYNOPSIS**

```
59148 PS      #include <sched.h>
59149      int sched_getparam(pid_t pid, struct sched_param *param);
```

59150 **DESCRIPTION**

59151 The *sched\_getparam()* function shall return the scheduling parameters of a process specified by  
 59152 *pid* in the **sched\_param** structure pointed to by *param*.

59153 If a process specified by *pid* exists, and if the calling process has permission, the scheduling  
 59154 parameters for the process whose process ID is equal to *pid* shall be returned.

59155 If *pid* is zero, the scheduling parameters for the calling process shall be returned. The behavior of  
 59156 the *sched\_getparam()* function is unspecified if the value of *pid* is negative.

59157 **RETURN VALUE**

59158 Upon successful completion, the *sched\_getparam()* function shall return zero. If the call to  
 59159 *sched\_getparam()* is unsuccessful, the function shall return a value of  $-1$  and set *errno* to indicate  
 59160 the error.

59161 **ERRORS**

59162 The *sched\_getparam()* function shall fail if:

59163 [EPERM] The requesting process does not have permission to obtain the scheduling  
 59164 parameters of the specified process.

59165 [ESRCH] No process can be found corresponding to that specified by *pid*.

59166 **EXAMPLES**

59167 None.

59168 **APPLICATION USAGE**

59169 None.

59170 **RATIONALE**

59171 None.

59172 **FUTURE DIRECTIONS**

59173 None.

59174 **SEE ALSO**

59175 *sched\_getscheduler()*, *sched\_setparam()*, *sched\_setscheduler()*

59176 XBD **<sched.h>**

59177 **CHANGE HISTORY**

59178 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

59179 **Issue 6**

59180 The *sched\_getparam()* function is marked as part of the Process Scheduling option.

59181 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
 59182 implementation does not support the Process Scheduling option.

59183 **NAME**59184 sched\_getscheduler — get scheduling policy (**REALTIME**)59185 **SYNOPSIS**

```
59186 PS #include <sched.h>
59187 int sched_getscheduler(pid_t pid);
```

59188 **DESCRIPTION**

59189 The *sched\_getscheduler()* function shall return the scheduling policy of the process specified by  
59190 *pid*. If the value of *pid* is negative, the behavior of the *sched\_getscheduler()* function is  
59191 unspecified.

59192 The values that can be returned by *sched\_getscheduler()* are defined in the **<sched.h>** header.

59193 If a process specified by *pid* exists, and if the calling process has permission, the scheduling  
59194 policy shall be returned for the process whose process ID is equal to *pid*.

59195 If *pid* is zero, the scheduling policy shall be returned for the calling process.

59196 **RETURN VALUE**

59197 Upon successful completion, the *sched\_getscheduler()* function shall return the scheduling policy  
59198 of the specified process. If unsuccessful, the function shall return -1 and set *errno* to indicate the  
59199 error.

59200 **ERRORS**

59201 The *sched\_getscheduler()* function shall fail if:

59202 [EPERM] The requesting process does not have permission to determine the scheduling  
59203 policy of the specified process.

59204 [ESRCH] No process can be found corresponding to that specified by *pid*.

59205 **EXAMPLES**

59206 None.

59207 **APPLICATION USAGE**

59208 None.

59209 **RATIONALE**

59210 None.

59211 **FUTURE DIRECTIONS**

59212 None.

59213 **SEE ALSO**

59214 [sched\\_getparam\(\)](#), [sched\\_setparam\(\)](#), [sched\\_setscheduler\(\)](#)

59215 XBD **<sched.h>**

59216 **CHANGE HISTORY**

59217 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

59218 **Issue 6**

59219 The *sched\_getscheduler()* function is marked as part of the Process Scheduling option.

59220 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
59221 implementation does not support the Process Scheduling option.

59222 **NAME**59223 sched\_rr\_get\_interval — get execution time limits (**REALTIME**)59224 **SYNOPSIS**

```
59225 PS|TPS #include <sched.h>
59226 int sched_rr_get_interval(pid_t pid, struct timespec *interval);
```

59227 **DESCRIPTION**

59228 The *sched\_rr\_get\_interval()* function shall update the **timespec** structure referenced by the  
 59229 *interval* argument to contain the current execution time limit (that is, time quantum) for the  
 59230 process specified by *pid*. If *pid* is zero, the current execution time limit for the calling process  
 59231 shall be returned.

59232 **RETURN VALUE**

59233 If successful, the *sched\_rr\_get\_interval()* function shall return zero. Otherwise, it shall return a  
 59234 value of -1 and set *errno* to indicate the error.

59235 **ERRORS**

59236 The *sched\_rr\_get\_interval()* function shall fail if:

59237 [ESRCH] No process can be found corresponding to that specified by *pid*.

59238 **EXAMPLES**

59239 None.

59240 **APPLICATION USAGE**

59241 None.

59242 **RATIONALE**

59243 None.

59244 **FUTURE DIRECTIONS**

59245 None.

59246 **SEE ALSO**

59247 *sched\_getparam()*, *sched\_get\_priority\_max()*, *sched\_getscheduler()*, *sched\_setparam()*,  
 59248 *sched\_setscheduler()*

59249 XBD **<sched.h>**

59250 **CHANGE HISTORY**

59251 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

59252 **Issue 6**

59253 The *sched\_rr\_get\_interval()* function is marked as part of the Process Scheduling option.

59254 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
 59255 implementation does not support the Process Scheduling option.

59256 IEEE Std 1003.1-2001/Cor 1-2002, XSH/TC1/D6/53 is applied, changing the PS margin code in  
 59257 the SYNOPSIS to PS|TPS.



**NAME**

sched\_setparam — set scheduling parameters (**REALTIME**)

**SYNOPSIS**

```
#include <sched.h>

int sched_setparam(pid_t pid, const struct sched_param *param);
```

**DESCRIPTION**

The *sched\_setparam()* function shall set the scheduling parameters of the process specified by *pid* to the values specified by the **sched\_param** structure pointed to by *param*. The value of the *sched\_priority* member in the **sched\_param** structure shall be any integer within the inclusive priority range for the current scheduling policy of the process specified by *pid*. Higher numerical values for the priority represent higher priorities. If the value of *pid* is negative, the behavior of the *sched\_setparam()* function is unspecified.

If a process specified by *pid* exists, and if the calling process has permission, the scheduling parameters shall be set for the process whose process ID is equal to *pid*.

If *pid* is zero, the scheduling parameters shall be set for the calling process.

The conditions under which one process has permission to change the scheduling parameters of another process are implementation-defined.

Implementations may require the requesting process to have appropriate privileges to set its own scheduling parameters or those of another process.

See [Scheduling Policies](#) (on page 506) for a description on how this function affects the scheduling of the threads within the target process.

If the current scheduling policy for the target process is not **SCHED\_FIFO**, **SCHED\_RR**, or **SCHED\_SPORADIC**, the result is implementation-defined; this case includes the **SCHED\_OTHER** policy.

The specified *sched\_ss\_repl\_period* shall be greater than or equal to the specified *sched\_ss\_init\_budget* for the function to succeed; if it is not, then the function shall fail.

The value of *sched\_ss\_max\_repl* shall be within the inclusive range [1, {SS\_REPL\_MAX}] for the function to succeed; if not, the function shall fail. It is unspecified whether the *sched\_ss\_repl\_period* and *sched\_ss\_init\_budget* values are stored as provided by this function or are rounded to align with the resolution of the clock being used.

This function is not atomic with respect to other threads in the process. Threads may continue to execute while this function call is in the process of changing the scheduling policy for the underlying kernel-scheduled entities used by the process contention scope threads.

**RETURN VALUE**

If successful, the *sched\_setparam()* function shall return zero.

If the call to *sched\_setparam()* is unsuccessful, the priority shall remain unchanged, and the function shall return a value of **-1** and set *errno* to indicate the error.

**ERRORS**

The *sched\_setparam()* function shall fail if:

[EINVAL] One or more of the requested scheduling parameters is outside the range defined for the scheduling policy of the specified *pid*.

59299 [EPERM] The requesting process does not have permission to set the scheduling  
 59300 parameters for the specified process, or does not have appropriate privileges  
 59301 to invoke *sched\_setparam()*.

59302 [ESRCH] No process can be found corresponding to that specified by *pid*.

#### 59303 EXAMPLES

59304 None.

#### 59305 APPLICATION USAGE

59306 None.

#### 59307 RATIONALE

59308 None.

#### 59309 FUTURE DIRECTIONS

59310 None.

#### 59311 SEE ALSO

59312 [Scheduling Policies](#) (on page 506), *sched\_getparam()*, *sched\_getscheduler()*, *sched\_setscheduler()*

59313 XBD [<sched.h>](#)

#### 59314 CHANGE HISTORY

59315 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

#### 59316 Issue 6

59317 The *sched\_setparam()* function is marked as part of the Process Scheduling option.

59318 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
 59319 implementation does not support the Process Scheduling option.

59320 The following new requirements on POSIX implementations derive from alignment with the  
 59321 Single UNIX Specification:

59322 • In the DESCRIPTION, the effect of this function on a thread's scheduling parameters is  
 59323 added.

59324 • Sections describing two-level scheduling and atomicity of the function are added.

59325 The SCHED\_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

59326 IEEE PASC Interpretation 1003.1 #100 is applied.

#### 59327 Issue 7

59328 Austin Group Interpretation 1003.1-2001 #061 is applied, updating the DESCRIPTION.

59329 Austin Group Interpretation 1003.1-2001 #119 is applied, clarifying the accuracy requirements  
 59330 for the *sched\_ss\_repl\_period* and *sched\_ss\_init\_budget* values.

59331 **NAME**59332 sched\_setscheduler — set scheduling policy and parameters (**REALTIME**)59333 **SYNOPSIS**

```
59334 PS      #include <sched.h>
59335
59336      int sched_setscheduler(pid_t pid, int policy,
          const struct sched_param *param);
```

59337 **DESCRIPTION**

59338 The *sched\_setscheduler()* function shall set the scheduling policy and scheduling parameters of  
 59339 the process specified by *pid* to *policy* and the parameters specified in the **sched\_param** structure  
 59340 pointed to by *param*, respectively. The value of the *sched\_priority* member in the **sched\_param**  
 59341 structure shall be any integer within the inclusive priority range for the scheduling policy  
 59342 specified by *policy*. If the value of *pid* is negative, the behavior of the *sched\_setscheduler()*  
 59343 function is unspecified.

59344 The possible values for the *policy* parameter are defined in the **<sched.h>** header.

59345 If a process specified by *pid* exists, and if the calling process has permission, the scheduling  
 59346 policy and scheduling parameters shall be set for the process whose process ID is equal to *pid*.

59347 If *pid* is zero, the scheduling policy and scheduling parameters shall be set for the calling  
 59348 process.

59349 The conditions under which one process has appropriate privileges to change the scheduling  
 59350 parameters of another process are implementation-defined.

59351 Implementations may require that the requesting process have permission to set its own  
 59352 scheduling parameters or those of another process. Additionally, implementation-defined  
 59353 restrictions may apply as to the appropriate privileges required to set the scheduling policy of  
 59354 the process, or the scheduling policy of another process, to a particular value.

59355 The *sched\_setscheduler()* function shall be considered successful if it succeeds in setting the  
 59356 scheduling policy and scheduling parameters of the process specified by *pid* to the values  
 59357 specified by *policy* and the structure pointed to by *param*, respectively.

59358 See [Scheduling Policies](#) (on page 506) for a description on how this function affects the  
 59359 scheduling of the threads within the target process.

59360 SS If the current scheduling policy for the target process is not SCHED\_FIFO, SCHED\_RR, or  
 59361 SCHED\_SPORADIC, the result is implementation-defined; this case includes the  
 59362 SCHED\_OTHER policy.

59363 SS The specified *sched\_ss\_repl\_period* shall be greater than or equal to the specified  
 59364 *sched\_ss\_init\_budget* for the function to succeed; if it is not, then the function shall fail.

59365 The value of *sched\_ss\_max\_repl* shall be within the inclusive range [1,{SS\_REPL\_MAX}] for the  
 59366 function to succeed; if not, the function shall fail. It is unspecified whether the  
 59367 *sched\_ss\_repl\_period* and *sched\_ss\_init\_budget* values are stored as provided by this function or are  
 59368 rounded to align with the resolution of the clock being used.

59369 This function is not atomic with respect to other threads in the process. Threads may continue to  
 59370 execute while this function call is in the process of changing the scheduling policy and  
 59371 associated scheduling parameters for the underlying kernel-scheduled entities used by the  
 59372 process contention scope threads.

**RETURN VALUE**

Upon successful completion, the function shall return the former scheduling policy of the specified process. If the *sched\_setscheduler()* function fails to complete successfully, the policy and scheduling parameters shall remain unchanged, and the function shall return a value of `-1` and set *errno* to indicate the error.

**ERRORS**

The *sched\_setscheduler()* function shall fail if:

[EINVAL] The value of the *policy* parameter is invalid, or one or more of the parameters contained in *param* is outside the valid range for the specified scheduling policy.

[EPERM] The requesting process does not have permission to set either or both of the scheduling parameters or the scheduling policy of the specified process.

[ESRCH] No process can be found corresponding to that specified by *pid*.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Scheduling Policies](#) (on page 506), *sched\_getparam()*, *sched\_getscheduler()*, *sched\_setparam()*

XBD [<sched.h>](#)

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**Issue 6**

The *sched\_setscheduler()* function is marked as part of the Process Scheduling option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Process Scheduling option.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the effect of this function on a thread's scheduling parameters is added.
- Sections describing two-level scheduling and atomicity of the function are added.

The SCHED\_SPORADIC scheduling policy is added for alignment with IEEE Std 1003.1d-1999.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #061 is applied, updating the DESCRIPTION.

Austin Group Interpretation 1003.1-2001 #119 is applied, clarifying the accuracy requirements for the *sched\_ss\_repl\_period* and *sched\_ss\_init\_budget* values.

59413 **NAME**

59414 sched\_yield — yield the processor

59415 **SYNOPSIS**

59416 #include &lt;sched.h&gt;

59417 int sched\_yield(void);

59418 **DESCRIPTION**59419 The *sched\_yield()* function shall force the running thread to relinquish the processor until it again  
59420 becomes the head of its thread list. It takes no arguments.59421 **RETURN VALUE**59422 The *sched\_yield()* function shall return 0 if it completes successfully; otherwise, it shall return a  
59423 value of -1 and set *errno* to indicate the error.59424 **ERRORS**

59425 No errors are defined.

59426 **EXAMPLES**

59427 None.

59428 **APPLICATION USAGE**59429 The conceptual model for scheduling semantics in POSIX.1-2008 defines a set of thread lists. This  
59430 set of thread lists is always present regardless of the scheduling options supported by the  
59431 system. On a system where the Process Scheduling option is not supported, portable  
59432 applications should not make any assumptions regarding whether threads from other processes  
59433 will be on the same thread list.59434 **RATIONALE**

59435 None.

59436 **FUTURE DIRECTIONS**

59437 None.

59438 **SEE ALSO**59439 XBD [<sched.h>](#)59440 **CHANGE HISTORY**59441 First released in Issue 5. Included for alignment with the POSIX Realtime Extension and the  
59442 POSIX Threads Extension.59443 **Issue 6**59444 The *sched\_yield()* function is now marked as part of the Process Scheduling and Threads options.59445 **Issue 7**

59446 SD5-XSH-ERN-120 is applied, adding APPLICATION USAGE.

59447 The *sched\_yield()* function is moved to the Base.

59448 **NAME**

59449 seed48 — seed a uniformly distributed pseudo-random non-negative long integer generator

59450 **SYNOPSIS**59451 XSI `#include <stdlib.h>`59452 `unsigned short *seed48(unsigned short seed16v[3]);`59453 **DESCRIPTION**59454 Refer to *drand48()*.

**NAME**

seekdir — set the position of a directory stream

**SYNOPSIS**

```
XSI    #include <dirent.h>
void seekdir(DIR *dirp, long loc);
```

**DESCRIPTION**

The *seekdir()* function shall set the position of the next *readdir()* operation on the directory stream specified by *dirp* to the position specified by *loc*. The value of *loc* should have been returned from an earlier call to *telldir()* using the same directory stream. The new position reverts to the one associated with the directory stream when *telldir()* was performed.

If the value of *loc* was not obtained from an earlier call to *telldir()*, or if a call to *rewinddir()* occurred between the call to *telldir()* and the call to *seekdir()*, the results of subsequent calls to *readdir()* are unspecified.

**RETURN VALUE**

The *seekdir()* function shall not return a value.

**ERRORS**

No errors are defined.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

The original standard developers perceived that there were restrictions on the use of the *seekdir()* and *telldir()* functions related to implementation details, and for that reason these functions need not be supported on all POSIX-conforming systems. They are required on implementations supporting the XSI option.

One of the perceived problems of implementation is that returning to a given point in a directory is quite difficult to describe formally, in spite of its intuitive appeal, when systems that use B-trees, hashing functions, or other similar mechanisms to order their directories are considered. The definition of *seekdir()* and *telldir()* does not specify whether, when using these interfaces, a given directory entry will be seen at all, or more than once.

On systems not supporting these functions, their capability can sometimes be accomplished by saving a filename found by *readdir()* and later using *rewinddir()* and a loop on *readdir()* to relocate the position from which the filename was saved.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*fdopendir()*, *readdir()*, *telldir()*

XBD *<dirent.h>*, *<sys/types.h>*

**CHANGE HISTORY**

First released in Issue 2.

59496 **Issue 6**

59497 In the SYNOPSIS, the inclusion of `<sys/types.h>` is no longer required.

59498 **Issue 7**

59499 SD5-XSH-ERN-200 is applied, updating the DESCRIPTION to note that the value of *loc* should  
59500 have been returned from an earlier call to *telldir()* using the same directory stream.



59501 **NAME**

59502       select — synchronous I/O multiplexing

59503 **SYNOPSIS**

59504       #include &lt;sys/select.h&gt;

```
59505       int select(int nfds, fd_set *restrict readfds,  
59506                fd_set *restrict writefds, fd_set *restrict errorfds,  
59507                struct timeval *restrict timeout);
```

59508 **DESCRIPTION**59509       Refer to *pselect()*.

59510 **NAME**

59511 sem\_close — close a named semaphore

59512 **SYNOPSIS**

59513 #include &lt;semaphore.h&gt;

59514 int sem\_close(sem\_t \*sem);

59515 **DESCRIPTION**

59516 The *sem\_close()* function shall indicate that the calling process is finished using the named  
59517 semaphore indicated by *sem*. The effects of calling *sem\_close()* for an unnamed semaphore (one  
59518 created by *sem\_init()*) are undefined. The *sem\_close()* function shall deallocate (that is, make  
59519 available for reuse by a subsequent *sem\_open()* by this process) any system resources allocated  
59520 by the system for use by this process for this semaphore. The effect of subsequent use of the  
59521 semaphore indicated by *sem* by this process is undefined. If any threads in the calling process are  
59522 currently blocked on the semaphore, the behavior is undefined. If the semaphore has not been  
59523 removed with a successful call to *sem\_unlink()*, then *sem\_close()* has no effect on the state of the  
59524 semaphore. If the *sem\_unlink()* function has been successfully invoked for *name* after the most  
59525 recent call to *sem\_open()* with O\_CREAT for this semaphore, then when all processes that have  
59526 opened the semaphore close it, the semaphore is no longer accessible.

59527 **RETURN VALUE**

59528 Upon successful completion, a value of zero shall be returned. Otherwise, a value of -1 shall be  
59529 returned and *errno* set to indicate the error.

59530 **ERRORS**59531 The *sem\_close()* function may fail if:59532 [EINVAL] The *sem* argument is not a valid semaphore descriptor.59533 **EXAMPLES**

59534 None.

59535 **APPLICATION USAGE**

59536 None.

59537 **RATIONALE**

59538 None.

59539 **FUTURE DIRECTIONS**

59540 None.

59541 **SEE ALSO**59542 *semctl()*, *semget()*, *semop()*, *sem\_init()*, *sem\_open()*, *sem\_unlink()*

59543 XBD &lt;semaphore.h&gt;

59544 **CHANGE HISTORY**

59545 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

59546 **Issue 6**59547 The *sem\_close()* function is marked as part of the Semaphores option.

59548 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
59549 implementation does not support the Semaphores option.

59550 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/113 is applied, updating the ERRORS  
59551 section so that the [EINVAL] error becomes optional.

59552 **Issue 7**

59553

The *sem\_close()* function is moved from the Semaphores option to the Base.

59554

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0523 [37] is applied.

59555 **NAME**

59556 sem\_destroy — destroy an unnamed semaphore

59557 **SYNOPSIS**

59558 #include <semaphore.h>

59559 int sem\_destroy(sem\_t \*sem);

59560 **DESCRIPTION**

59561 The *sem\_destroy()* function shall destroy the unnamed semaphore indicated by *sem*. Only a  
59562 semaphore that was created using *sem\_init()* may be destroyed using *sem\_destroy()*; the effect of  
59563 calling *sem\_destroy()* with a named semaphore is undefined. The effect of subsequent use of the  
59564 semaphore *sem* is undefined until *sem* is reinitialized by another call to *sem\_init()*.

59565 It is safe to destroy an initialized semaphore upon which no threads are currently blocked. The  
59566 effect of destroying a semaphore upon which other threads are currently blocked is undefined.

59567 **RETURN VALUE**

59568 Upon successful completion, a value of zero shall be returned. Otherwise, a value of -1 shall be  
59569 returned and *errno* set to indicate the error.

59570 **ERRORS**

59571 The *sem\_destroy()* function may fail if:

59572 [EINVAL] The *sem* argument is not a valid semaphore.

59573 [EBUSY] There are currently processes blocked on the semaphore.

59574 **EXAMPLES**

59575 None.

59576 **APPLICATION USAGE**

59577 None.

59578 **RATIONALE**

59579 None.

59580 **FUTURE DIRECTIONS**

59581 None.

59582 **SEE ALSO**

59583 *semctl()*, *semget()*, *semop()*, *sem\_init()*, *sem\_open()*

59584 XBD <semaphore.h>

59585 **CHANGE HISTORY**

59586 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

59587 **Issue 6**

59588 The *sem\_destroy()* function is marked as part of the Semaphores option.

59589 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
59590 implementation does not support the Semaphores option.

59591 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/114 is applied, updating the ERRORS  
59592 section so that the [EINVAL] error becomes optional.

59593 **Issue 7**

59594 The *sem\_destroy()* function is moved from the Semaphores option to the Base.

59595 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0524 [37] is applied.

**59596 NAME**

59597 sem\_getvalue — get the value of a semaphore

**59598 SYNOPSIS**

59599 #include <semaphore.h>

59600 int sem\_getvalue(sem\_t \*restrict sem, int \*restrict sval);

**59601 DESCRIPTION**

59602 The *sem\_getvalue()* function shall update the location referenced by the *sval* argument to have  
59603 the value of the semaphore referenced by *sem* without affecting the state of the semaphore. The  
59604 updated value represents an actual semaphore value that occurred at some unspecified time  
59605 during the call, but it need not be the actual value of the semaphore when it is returned to the  
59606 calling process.

59607 If *sem* is locked, then the object to which *sval* points shall either be set to zero or to a negative  
59608 number whose absolute value represents the number of processes waiting for the semaphore at  
59609 some unspecified time during the call.

**59610 RETURN VALUE**

59611 Upon successful completion, the *sem\_getvalue()* function shall return a value of zero. Otherwise,  
59612 it shall return a value of -1 and set *errno* to indicate the error.

**59613 ERRORS**

59614 The *sem\_getvalue()* function may fail if:

59615 [EINVAL] The *sem* argument does not refer to a valid semaphore.

**59616 EXAMPLES**

59617 None.

**59618 APPLICATION USAGE**

59619 None.

**59620 RATIONALE**

59621 None.

**59622 FUTURE DIRECTIONS**

59623 None.

**59624 SEE ALSO**

59625 *semctl()*, *semget()*, *semop()*, *sem\_post()*, *sem\_timedwait()*, *sem\_trywait()*

59626 XBD <semaphore.h>

**59627 CHANGE HISTORY**

59628 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**59629 Issue 6**

59630 The *sem\_getvalue()* function is marked as part of the Semaphores option.

59631 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
59632 implementation does not support the Semaphores option.

59633 The *sem\_timedwait()* function is added to the SEE ALSO section for alignment with IEEE Std  
59634 1003.1d-1999.

59635 The **restrict** keyword is added to the *sem\_getvalue()* prototype for alignment with the  
59636 ISO/IEC 9899:1999 standard.

59637 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/54 is applied.

- 59638
- 59639
- 59640
- 59641
- 59642
- IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/115 is applied, updating the ERRORS section so that the [EINVAL] error becomes optional.

**Issue 7**

The *sem\_getvalue()* function is moved from the Semaphores option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0525 [37] is applied.

59643 **NAME**

59644 sem\_init — initialize an unnamed semaphore

59645 **SYNOPSIS**

59646 #include &lt;semaphore.h&gt;

59647 int sem\_init(sem\_t \*sem, int pshared, unsigned value);

59648 **DESCRIPTION**

59649 The *sem\_init()* function shall initialize the unnamed semaphore referred to by *sem*. The value of  
 59650 the initialized semaphore shall be *value*. Following a successful call to *sem\_init()*, the semaphore  
 59651 may be used in subsequent calls to *sem\_wait()*, *sem\_timedwait()*, *sem\_trywait()*, *sem\_post()*, and  
 59652 *sem\_destroy()*. This semaphore shall remain usable until the semaphore is destroyed.

59653 If the *pshared* argument has a non-zero value, then the semaphore is shared between processes;  
 59654 in this case, any process that can access the semaphore *sem* can use *sem* for performing  
 59655 *sem\_wait()*, *sem\_timedwait()*, *sem\_trywait()*, *sem\_post()*, and *sem\_destroy()* operations.

59656 If the *pshared* argument is zero, then the semaphore is shared between threads of the process; any  
 59657 thread in this process can use *sem* for performing *sem\_wait()*, *sem\_timedwait()*, *sem\_trywait()*,  
 59658 *sem\_post()*, and *sem\_destroy()* operations.

59659 See [Section 2.9.9](#) (on page 523) for further requirements.

59660 Attempting to initialize an already initialized semaphore results in undefined behavior.

59661 **RETURN VALUE**

59662 Upon successful completion, the *sem\_init()* function shall initialize the semaphore in *sem* and  
 59663 return 0. Otherwise, it shall return -1 and set *errno* to indicate the error.

59664 **ERRORS**

59665 The *sem\_init()* function shall fail if:

- |       |          |                                                                            |
|-------|----------|----------------------------------------------------------------------------|
| 59666 | [EINVAL] | The <i>value</i> argument exceeds {SEM_VALUE_MAX}.                         |
| 59667 | [ENOSPC] | A resource required to initialize the semaphore has been exhausted, or the |
| 59668 |          | limit on semaphores ({SEM_NSEMS_MAX}) has been reached.                    |
| 59669 | [EPERM]  | The process lacks appropriate privileges to initialize the semaphore.      |

59670 **EXAMPLES**

59671 None.

59672 **APPLICATION USAGE**

59673 None.

59674 **RATIONALE**

59675 None.

59676 **FUTURE DIRECTIONS**

59677 None.

59678 **SEE ALSO**

59679 [sem\\_destroy\(\)](#), [sem\\_post\(\)](#), [sem\\_timedwait\(\)](#), [sem\\_trywait\(\)](#)

59680 XBD [<semaphore.h>](#)

59681 **CHANGE HISTORY**

59682 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

59683 **Issue 6**

59684 The *sem\_init()* function is marked as part of the Semaphores option.

59685 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
59686 implementation does not support the Semaphores option.

59687 The *sem\_timedwait()* function is added to the SEE ALSO section for alignment with IEEE Std  
59688 1003.1d-1999.

59689 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/116 is applied, updating the  
59690 DESCRIPTION to add the *sem\_timedwait()* function for alignment with IEEE Std 1003.1d-1999.

59691 **Issue 7**

59692 SD5-XSH-ERN-176 is applied.

59693 The *sem\_init()* function is moved from the Semaphores option to the Base.

59694 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0526 [37] is applied.



**NAME**

sem\_open — initialize and open a named semaphore

**SYNOPSIS**

```
#include <semaphore.h>
```

```
sem_t *sem_open(const char *name, int oflag, ...);
```

**DESCRIPTION**

The *sem\_open()* function shall establish a connection between a named semaphore and a process. Following a call to *sem\_open()* with semaphore name *name*, the process may reference the semaphore associated with *name* using the address returned from the call. This semaphore may be used in subsequent calls to *sem\_wait()*, *sem\_timedwait()*, *sem\_trywait()*, *sem\_post()*, and *sem\_close()*. The semaphore remains usable by this process until the semaphore is closed by a successful call to *sem\_close()*, *\_exit()*, or one of the *exec* functions.

The *oflag* argument controls whether the semaphore is created or merely accessed by the call to *sem\_open()*. The following flag bits may be set in *oflag*:

**O\_CREAT** This flag is used to create a semaphore if it does not already exist. If **O\_CREAT** is set and the semaphore already exists, then **O\_CREAT** has no effect, except as noted under **O\_EXCL**. Otherwise, *sem\_open()* creates a named semaphore. The **O\_CREAT** flag requires a third and a fourth argument: *mode*, which is of type **mode\_t**, and *value*, which is of type **unsigned**. The semaphore is created with an initial value of *value*. Valid initial values for semaphores are less than or equal to **{SEM\_VALUE\_MAX}**.

The user ID of the semaphore shall be set to the effective user ID of the process. The group ID of the semaphore shall be set to the effective group ID of the process; however, if the *name* argument is visible in the file system, the group ID may be set to the group ID of the containing directory. The permission bits of the semaphore are set to the value of the *mode* argument except those set in the file mode creation mask of the process. When bits in *mode* other than the file permission bits are specified, the effect is unspecified.

After the semaphore named *name* has been created by *sem\_open()* with the **O\_CREAT** flag, other processes can connect to the semaphore by calling *sem\_open()* with the same value of *name*.

**O\_EXCL** If **O\_EXCL** and **O\_CREAT** are set, *sem\_open()* fails if the semaphore *name* exists. The check for the existence of the semaphore and the creation of the semaphore if it does not exist are atomic with respect to other processes executing *sem\_open()* with **O\_EXCL** and **O\_CREAT** set. If **O\_EXCL** is set and **O\_CREAT** is not set, the effect is undefined.

If flags other than **O\_CREAT** and **O\_EXCL** are specified in the *oflag* parameter, the effect is unspecified.

The *name* argument points to a string naming a semaphore object. It is unspecified whether the name appears in the file system and is visible to functions that take pathnames as arguments. The *name* argument conforms to the construction rules for a pathname, except that the interpretation of *<slash>* characters other than the leading *<slash>* character in *name* is implementation-defined, and that the length limits for the *name* argument are implementation-defined and need not be the same as the pathname limits **{PATH\_MAX}** and **{NAME\_MAX}**. If *name* begins with the *<slash>* character, then processes calling *sem\_open()* with the same value of *name* shall refer to the same semaphore object, as long as that name has not been removed. If *name* does not begin with the *<slash>* character, the effect is implementation-defined.

59742 If a process makes multiple successful calls to *sem\_open()* with the same value for *name*, the same  
 59743 semaphore address shall be returned for each such successful call, provided that there have been  
 59744 no calls to *sem\_unlink()* for this semaphore, and at least one previous successful *sem\_open()* call  
 59745 for this semaphore has not been matched with a *sem\_close()* call.

59746 References to copies of the semaphore produce undefined results.

#### 59747 RETURN VALUE

59748 Upon successful completion, the *sem\_open()* function shall return the address of the semaphore.  
 59749 Otherwise, it shall return a value of SEM\_FAILED and set *errno* to indicate the error. The symbol  
 59750 SEM\_FAILED is defined in the <semaphore.h> header. No successful return from *sem\_open()*  
 59751 shall return the value SEM\_FAILED.

#### 59752 ERRORS

59753 If any of the following conditions occur, the *sem\_open()* function shall return SEM\_FAILED and  
 59754 set *errno* to the corresponding value:

59755 [EACCES] The named semaphore exists and the permissions specified by *oflag* are  
 59756 denied, or the named semaphore does not exist and permission to create the  
 59757 named semaphore is denied.

59758 [EEXIST] O\_CREAT and O\_EXCL are set and the named semaphore already exists.

59759 [EINTR] The *sem\_open()* operation was interrupted by a signal.

59760 [EINVAL] The *sem\_open()* operation is not supported for the given name, or O\_CREAT  
 59761 was specified in *oflag* and *value* was greater than {SEM\_VALUE\_MAX}.

59762 [EMFILE] Too many semaphore descriptors or file descriptors are currently in use by this  
 59763 process.

59764 [ENFILE] Too many semaphores are currently open in the system.

59765 [ENOENT] O\_CREAT is not set and the named semaphore does not exist.

59766 [ENOMEM] There is insufficient memory for the creation of the new named semaphore.

59767 [ENOSPC] There is insufficient space on a storage device for the creation of the new  
 59768 named semaphore.

59769 If any of the following conditions occur, the *sem\_open()* function may return SEM\_FAILED and  
 59770 set *errno* to the corresponding value:

59771 [ENAMETOOLONG]

59772 The length of the *name* argument exceeds {\_POSIX\_PATH\_MAX} on systems  
 59773 XSI that do not support the XSI option or exceeds {\_XOPEN\_PATH\_MAX} on XSI  
 59774 systems, or has a pathname component that is longer than  
 59775 XSI {\_POSIX\_NAME\_MAX} on systems that do not support the XSI option or  
 59776 longer than {\_XOPEN\_NAME\_MAX} on XSI systems.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

Early drafts required an error return value of -1 with the type `sem_t *` for the `sem_open()` function, which is not guaranteed to be portable across implementations. The revised text provides the symbolic error code `SEM_FAILED` to eliminate the type conflict.

**FUTURE DIRECTIONS**

A future version might require the `sem_open()` and `sem_unlink()` functions to have semantics similar to normal file system operations.

**SEE ALSO**

`semctl()`, `semget()`, `semop()`, `sem_close()`, `sem_post()`, `sem_timedwait()`, `sem_trywait()`, `sem_unlink()`

XBD `<semaphore.h>`

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**Issue 6**

The `sem_open()` function is marked as part of the Semaphores option.

The `[ENOSYS]` error condition has been removed as stubs need not be provided if an implementation does not support the Semaphores option.

The `sem_timedwait()` function is added to the SEE ALSO section for alignment with IEEE Std 1003.1d-1999.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/117 is applied, updating the DESCRIPTION to add the `sem_timedwait()` function for alignment with IEEE Std 1003.1d-1999.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/118 is applied, updating the DESCRIPTION to describe the conditions to return the same semaphore address on a call to `sem_open()`. The words “and at least one previous successful `sem_open()` call for this semaphore has not been matched with a `sem_close()` call” are added.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #066 is applied, updating the `[ENOSPC]` error case and adding the `[ENOMEM]` error case.

Austin Group Interpretation 1003.1-2001 #077 is applied, clarifying the `name` argument and adding `[ENAMETOOLONG]` as a “may fail” error.

Austin Group Interpretation 1003.1-2001 #141 is applied, adding FUTURE DIRECTIONS.

SD5-XSH-ERN-170 is applied, updating the DESCRIPTION to clarify the wording for setting the user ID and group ID of the semaphore.

The `sem_open()` function is moved from the Semaphores option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0527 [37] is applied.

59815 **NAME**

59816 sem\_post — unlock a semaphore

59817 **SYNOPSIS**

59818 #include &lt;semaphore.h&gt;

59819 int sem\_post(sem\_t \*sem);

59820 **DESCRIPTION**59821 The *sem\_post()* function shall unlock the semaphore referenced by *sem* by performing a  
59822 semaphore unlock operation on that semaphore.59823 If the semaphore value resulting from this operation is positive, then no threads were blocked  
59824 waiting for the semaphore to become unlocked; the semaphore value is simply incremented.59825 If the value of the semaphore resulting from this operation is zero, then one of the threads  
59826 blocked waiting for the semaphore shall be allowed to return successfully from its call to  
59827 PS *sem\_wait()*. If the Process Scheduling option is supported, the thread to be unblocked shall be  
59828 chosen in a manner appropriate to the scheduling policies and parameters in effect for the  
59829 blocked threads. In the case of the schedulers SCHED\_FIFO and SCHED\_RR, the highest  
59830 priority waiting thread shall be unblocked, and if there is more than one highest priority thread  
59831 blocked waiting for the semaphore, then the highest priority thread that has been waiting the  
59832 longest shall be unblocked. If the Process Scheduling option is not defined, the choice of a thread  
59833 to unblock is unspecified.59834 SS If the Process Sporadic Server option is supported, and the scheduling policy is  
59835 SCHED\_SPORADIC, the semantics are as per SCHED\_FIFO above.59836 The *sem\_post()* function shall be async-signal-safe and may be invoked from a signal-catching  
59837 function.59838 **RETURN VALUE**59839 If successful, the *sem\_post()* function shall return zero; otherwise, the function shall return -1  
59840 and set *errno* to indicate the error.59841 **ERRORS**59842 The *sem\_post()* function may fail if:59843 [EINVAL] The *sem* argument does not refer to a valid semaphore.59844 **EXAMPLES**59845 See *sem\_timedwait()*.59846 **APPLICATION USAGE**

59847 None.

59848 **RATIONALE**

59849 None.

59850 **FUTURE DIRECTIONS**

59851 None.

59852 **SEE ALSO**59853 *semctl()*, *semget()*, *semop()*, *sem\_timedwait()*, *sem\_trywait()*

59854 XBD Section 4.12 (on page 111), &lt;semaphore.h&gt;

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**Issue 6**

The `sem_post()` function is marked as part of the Semaphores option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Semaphores option.

The `sem_timedwait()` function is added to the SEE ALSO section for alignment with IEEE Std 1003.1d-1999.

SCHED\_SPORADIC is added to the list of scheduling policies for which the thread that is to be unblocked is specified for alignment with IEEE Std 1003.1d-1999.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/119 is applied, updating the ERRORS section so that the [EINVAL] error becomes optional.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #156 is applied.

The `sem_post()` function is moved from the Semaphores option to the Base.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0528 [37] is applied.

59871 **NAME**

59872 sem\_timedwait — lock a semaphore

59873 **SYNOPSIS**

```
59874 #include <semaphore.h>
59875 #include <time.h>
59876 int sem_timedwait(sem_t *restrict sem,
59877                  const struct timespec *restrict abstime);
```

59878 **DESCRIPTION**

59879 The *sem\_timedwait()* function shall lock the semaphore referenced by *sem* as in the *sem\_wait()*  
 59880 function. However, if the semaphore cannot be locked without waiting for another process or  
 59881 thread to unlock the semaphore by performing a *sem\_post()* function, this wait shall be  
 59882 terminated when the specified timeout expires.

59883 The timeout shall expire when the absolute time specified by *abstime* passes, as measured by the  
 59884 clock on which timeouts are based (that is, when the value of that clock equals or exceeds  
 59885 *abstime*), or if the absolute time specified by *abstime* has already been passed at the time of the  
 59886 call.

59887 The timeout shall be based on the CLOCK\_REALTIME clock. The resolution of the timeout shall  
 59888 be the resolution of the clock on which it is based. The **timespec** data type is defined as a  
 59889 structure in the **<time.h>** header.

59890 Under no circumstance shall the function fail with a timeout if the semaphore can be locked  
 59891 immediately. The validity of the *abstime* need not be checked if the semaphore can be locked  
 59892 immediately.

59893 **RETURN VALUE**

59894 The *sem\_timedwait()* function shall return zero if the calling process successfully performed the  
 59895 semaphore lock operation on the semaphore designated by *sem*. If the call was unsuccessful, the  
 59896 state of the semaphore shall be unchanged, and the function shall return a value of -1 and set  
 59897 *errno* to indicate the error.

59898 **ERRORS**

59899 The *sem\_timedwait()* function shall fail if:

59900 [EINVAL] The process or thread would have blocked, and the *abstime* parameter  
 59901 specified a nanoseconds field value less than zero or greater than or equal to  
 59902 1 000 million.

59903 [ETIMEDOUT] The semaphore could not be locked before the specified timeout expired.

59904 The *sem\_timedwait()* function may fail if:

59905 [EDEADLK] A deadlock condition was detected.

59906 [EINTR] A signal interrupted this function.

59907 [EINVAL] The *sem* argument does not refer to a valid semaphore.

**EXAMPLES**

The program shown below operates on an unnamed semaphore. The program expects two command-line arguments. The first argument specifies a seconds value that is used to set an alarm timer to generate a SIGALRM signal. This handler performs a *sem\_post*(3) to increment the semaphore that is being waited on in *main*() using *sem\_timedwait*(). The second command-line argument specifies the length of the timeout, in seconds, for *sem\_timedwait*().

```

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <semaphore.h>
#include <time.h>
#include <assert.h>
#include <errno.h>
#include <signal.h>

sem_t sem;

static void
handler(int sig)
{
    int sav_errno = errno;
    static const char info_msg[] = "sem_post() from handler\n";
    write(STDOUT_FILENO, info_msg, sizeof info_msg - 1);
    if (sem_post(&sem) == -1) {
        static const char err_msg[] = "sem_post() failed\n";
        write(STDERR_FILENO, err_msg, sizeof err_msg - 1);
        _exit(EXIT_FAILURE);
    }
    errno = sav_errno;
}

int
main(int argc, char *argv[])
{
    struct sigaction sa;
    struct timespec ts;
    int s;

    if (argc != 3) {
        fprintf(stderr, "Usage: %s <alarm-secs> <wait-secs>\n",
            argv[0]);
        exit(EXIT_FAILURE);
    }

    if (sem_init(&sem, 0, 0) == -1) {
        perror("sem_init");
        exit(EXIT_FAILURE);
    }

    /* Establish SIGALRM handler; set alarm timer using argv[1] */

    sa.sa_handler = handler;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;
    if (sigaction(SIGALRM, &sa, NULL) == -1) {

```



```

59956         perror("sigaction");
59957         exit(EXIT_FAILURE);
59958     }
59959     alarm(atoi(argv[1]));
59960     /* Calculate relative interval as current time plus
59961        number of seconds given argv[2] */
59962     if (clock_gettime(CLOCK_REALTIME, &ts) == -1) {
59963         perror("clock_gettime");
59964         exit(EXIT_FAILURE);
59965     }
59966     ts.tv_sec += atoi(argv[2]);
59967     printf("main() about to call sem_timedwait()\n");
59968     while ((s = sem_timedwait(&sem, &ts)) == -1 && errno == EINTR)
59969         continue;          /* Restart if interrupted by handler */
59970     /* Check what happened */
59971     if (s == -1) {
59972         if (errno == ETIMEDOUT)
59973             printf("sem_timedwait() timed out\n");
59974         else
59975             perror("sem_timedwait");
59976     } else
59977         printf("sem_timedwait() succeeded\n");
59978     exit((s == 0) ? EXIT_SUCCESS : EXIT_FAILURE);
59979 }

```

## APPLICATION USAGE

Applications using these functions may be subject to priority inversion, as discussed in XBD Section 3.291 (on page 80).

## RATIONALE

None.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[\*sem\\_post\(\)\*](#), [\*sem\\_trywait\(\)\*](#), [\*semctl\(\)\*](#), [\*semget\(\)\*](#), [\*semop\(\)\*](#), [\*time\(\)\*](#)

XBD Section 3.291 (on page 80), [\*\*<semaphore.h>\*\*](#), [\*\*<time.h>\*\*](#)

## CHANGE HISTORY

First released in Issue 6. Derived from IEEE Std 1003.1d-1999.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/120 is applied, updating the ERRORS section so that the [EINVAL] error becomes optional.

## Issue 7

The *sem\_timedwait()* function is moved from the Semaphores option to the Base.

Functionality relating to the Timers option is moved to the Base.

An example is added.



59998

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0529 [138] is applied.

59999 **NAME**

60000 `sem_trywait`, `sem_wait` — lock a semaphore

60001 **SYNOPSIS**

```
60002 #include <semaphore.h>
60003 int sem_trywait(sem_t *sem);
60004 int sem_wait(sem_t *sem);
```

60005 **DESCRIPTION**

60006 The `sem_trywait()` function shall lock the semaphore referenced by `sem` only if the semaphore is  
60007 currently not locked; that is, if the semaphore value is currently positive. Otherwise, it shall not  
60008 lock the semaphore.

60009 The `sem_wait()` function shall lock the semaphore referenced by `sem` by performing a semaphore  
60010 lock operation on that semaphore. If the semaphore value is currently zero, then the calling  
60011 thread shall not return from the call to `sem_wait()` until it either locks the semaphore or the call is  
60012 interrupted by a signal.

60013 Upon successful return, the state of the semaphore shall be locked and shall remain locked until  
60014 the `sem_post()` function is executed and returns successfully.

60015 The `sem_wait()` function is interruptible by the delivery of a signal.

60016 **RETURN VALUE**

60017 The `sem_trywait()` and `sem_wait()` functions shall return zero if the calling process successfully  
60018 performed the semaphore lock operation on the semaphore designated by `sem`. If the call was  
60019 unsuccessful, the state of the semaphore shall be unchanged, and the function shall return a  
60020 value of `-1` and set `errno` to indicate the error.

60021 **ERRORS**

60022 The `sem_trywait()` function shall fail if:

60023 [EAGAIN] The semaphore was already locked, so it cannot be immediately locked by the  
60024 `sem_trywait()` operation.

60025 The `sem_trywait()` and `sem_wait()` functions may fail if:

60026 [EDEADLK] A deadlock condition was detected.

60027 [EINTR] A signal interrupted this function.

60028 [EINVAL] The `sem` argument does not refer to a valid semaphore.

60029 **EXAMPLES**

60030 None.

60031 **APPLICATION USAGE**

60032 Applications using these functions may be subject to priority inversion, as discussed in XBD  
60033 [Section 3.291](#) (on page 80).

60034 **RATIONALE**

60035 None.

60036 **FUTURE DIRECTIONS**

60037 None.

60038 **SEE ALSO**

60039 [`semctl\(\)`](#), [`semget\(\)`](#), [`semop\(\)`](#), [`sem\_post\(\)`](#), [`sem\_timedwait\(\)`](#)

60040 XBD [Section 3.291](#) (on page 80), [Section 4.12](#) (on page 111), [`<semaphore.h>`](#)

**CHANGE HISTORY**

60041 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**Issue 6**

60043 The *sem\_trywait()* and *sem\_wait()* functions are marked as part of the Semaphores option.

60045 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
60046 implementation does not support the Semaphores option.

60047 The *sem\_timedwait()* function is added to the SEE ALSO section for alignment with IEEE Std  
60048 1003.1d-1999.

60049 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/121 is applied, updating the ERRORS  
60050 section so that the [EINVAL] error becomes optional.

**Issue 7**

60051 SD5-XSH-ERN-54 is applied, removing the *sem\_wait()* function from the “shall fail” error cases.

60053 The *sem\_trywait()* and *sem\_wait()* functions are moved from the Semaphores option to the Base.  
60054 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0530 [37] is applied.

60055 **NAME**

60056 sem\_unlink — remove a named semaphore

60057 **SYNOPSIS**

60058 #include &lt;semaphore.h&gt;

60059 int sem\_unlink(const char \*name);

60060 **DESCRIPTION**

60061 The *sem\_unlink()* function shall remove the semaphore named by the string *name*. If the  
60062 semaphore named by *name* is currently referenced by other processes, then *sem\_unlink()* shall  
60063 have no effect on the state of the semaphore. If one or more processes have the semaphore open  
60064 when *sem\_unlink()* is called, destruction of the semaphore is postponed until all references to the  
60065 semaphore have been destroyed by calls to *sem\_close()*, *\_exit()*, or *exec*. Calls to *sem\_open()* to  
60066 recreate or reconnect to the semaphore refer to a new semaphore after *sem\_unlink()* is called. The  
60067 *sem\_unlink()* call shall not block until all references have been destroyed; it shall return  
60068 immediately.

60069 **RETURN VALUE**

60070 Upon successful completion, the *sem\_unlink()* function shall return a value of 0. Otherwise, the  
60071 semaphore shall not be changed and the function shall return a value of -1 and set *errno* to  
60072 indicate the error.

60073 **ERRORS**60074 The *sem\_unlink()* function shall fail if:

60075 [EACCES] Permission is denied to unlink the named semaphore.

60076 [ENOENT] The named semaphore does not exist.

60077 The *sem\_unlink()* function may fail if:

60078 [ENAMETOOLONG]

60079 The length of the *name* argument exceeds {\_POSIX\_PATH\_MAX} on systems  
60080 XSI that do not support the XSI option or exceeds {\_XOPEN\_PATH\_MAX} on XSI  
60081 systems, or has a pathname component that is longer than  
60082 XSI {\_POSIX\_NAME\_MAX} on systems that do not support the XSI option or  
60083 longer than {\_XOPEN\_NAME\_MAX} on XSI systems. A call to *sem\_unlink()*  
60084 with a *name* argument that contains the same semaphore name as was  
60085 previously used in a successful *sem\_open()* call shall not give an  
60086 [ENAMETOOLONG] error.

60087 **EXAMPLES**

60088 None.

60089 **APPLICATION USAGE**

60090 None.

60091 **RATIONALE**

60092 None.

60093 **FUTURE DIRECTIONS**

60094 A future version might require the *sem\_open()* and *sem\_unlink()* functions to have semantics  
60095 similar to normal file system operations.

60096 **SEE ALSO**60097 *semctl()*, *semget()*, *semop()*, *sem\_close()*, *sem\_open()*

60098 XBD &lt;semaphore.h&gt;

**CHANGE HISTORY**

60100 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**Issue 6**

60102 The *sem\_unlink()* function is marked as part of the Semaphores option.

60103 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
60104 implementation does not support the Semaphores option.

**Issue 7**

60106 Austin Group Interpretation 1003.1-2001 #077 is applied, changing [ENAMETOOLONG] from a  
60107 “shall fail” to a “may fail” error.

60108 Austin Group Interpretation 1003.1-2001 #141 is applied, adding FUTURE DIRECTIONS.

60109 The *sem\_unlink()* function is moved from the Semaphores option to the Base.

60110 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0531 [37] is applied.

60111 **NAME**

60112       sem\_wait — lock a semaphore

60113 **SYNOPSIS**

60114       #include &lt;semaphore.h&gt;

60115       int sem\_wait(sem\_t \*sem);

60116 **DESCRIPTION**60117       Refer to *sem\_trywait()*.

**NAME**

semctl — XSI semaphore control operations

**SYNOPSIS**

```
XSI    #include <sys/sem.h>
      int semctl(int semid, int semnum, int cmd, ...);
```

**DESCRIPTION**

The *semctl()* function operates on XSI semaphores (see XBD [Section 4.17](#), on page 114). It is unspecified whether this function interoperates with the realtime interprocess communication facilities defined in [Section 2.8](#) (on page 503).

The *semctl()* function provides a variety of semaphore control operations as specified by *cmd*. The fourth argument is optional and depends upon the operation requested. If required, it is of type **union semun**, which the application shall explicitly declare:

```
union semun {
    int val;
    struct semid_ds *buf;
    unsigned short *array;
} arg;
```

Each operation shall be performed atomically. +

The following semaphore control operations as specified by *cmd* are executed with respect to the semaphore specified by *semid* and *semnum*. The level of permission required for each operation is shown with each command; see [Section 2.7](#) (on page 501). The symbolic names for the values of *cmd* are defined in the `<sys/sem.h>` header:

|         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|---------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GETVAL  | Return the value of <i>semval</i> ; see <code>&lt;sys/sem.h&gt;</code> . Requires read permission.                                                                                                                                                                                                                                                                                                                                                                                   |
| SETVAL  | Set the value of <i>semval</i> to <i>arg.val</i> , where <i>arg</i> is the value of the fourth argument to <i>semctl()</i> . When this command is successfully executed, the <i>semadj</i> value corresponding to the specified semaphore in all processes is cleared. Also, the <i>sem_ctime</i> timestamp shall be set to the current time, as described in <a href="#">Section 2.7.1</a> (on page 502). Requires alter permission; see <a href="#">Section 2.7</a> (on page 501). |
| GETPID  | Return the value of <i>sempid</i> . Requires read permission.                                                                                                                                                                                                                                                                                                                                                                                                                        |
| GETNCNT | Return the value of <i>semmcnt</i> . Requires read permission.                                                                                                                                                                                                                                                                                                                                                                                                                       |
| GETZCNT | Return the value of <i>semzcnt</i> . Requires read permission.                                                                                                                                                                                                                                                                                                                                                                                                                       |

The following values of *cmd* operate on each *semval* in the set of semaphores:

|        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GETALL | Return the value of <i>semval</i> for each semaphore in the semaphore set and place into the array pointed to by <i>arg.array</i> , where <i>arg</i> is the fourth argument to <i>semctl()</i> . Requires read permission.                                                                                                                                                                                                                                                                              |
| SETALL | Set the value of <i>semval</i> for each semaphore in the semaphore set according to the array pointed to by <i>arg.array</i> , where <i>arg</i> is the fourth argument to <i>semctl()</i> . When this command is successfully executed, the <i>semadj</i> values corresponding to each specified semaphore in all processes are cleared. Also, the <i>sem_ctime</i> timestamp shall be set to the current time, as described in <a href="#">Section 2.7.1</a> (on page 502). Requires alter permission. |

|       |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------|-----------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 60159 | The following values of <i>cmd</i> are also available:                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60160 | IPC_STAT                                                                                            | Place the current value of each member of the <b>semid_ds</b> data structure associated with <i>semid</i> into the structure pointed to by <i>arg.buf</i> , where <i>arg</i> is the fourth argument to <i>semctl()</i> . The contents of this structure are defined in <b>&lt;sys/sem.h&gt;</b> . Requires read permission.                                                                                                                       |
| 60161 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60162 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60163 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60164 | IPC_SET                                                                                             | Set the value of the following members of the <b>semid_ds</b> data structure associated with <i>semid</i> to the corresponding value found in the structure pointed to by <i>arg.buf</i> , where <i>arg</i> is the fourth argument to <i>semctl()</i> :                                                                                                                                                                                           |
| 60165 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60166 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60167 |                                                                                                     | <i>sem_perm.uid</i>                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 60168 |                                                                                                     | <i>sem_perm.gid</i>                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 60169 |                                                                                                     | <i>sem_perm.mode</i>                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 60170 |                                                                                                     | The mode bits specified in <a href="#">Section 2.7.1</a> (on page 502) are copied into the corresponding bits of the <i>sem_perm.mode</i> associated with <i>semid</i> . The stored values of any other bits are unspecified. The <i>sem_ctime</i> timestamp shall be set to the current time, as described in <a href="#">Section 2.7.1</a> (on page 502).                                                                                       |
| 60171 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60172 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60173 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60174 |                                                                                                     | This command can only be executed by a process that has an effective user ID equal to either that of a process with appropriate privileges or to the value of <i>sem_perm.cuid</i> or <i>sem_perm.uid</i> in the <b>semid_ds</b> data structure associated with <i>semid</i> .                                                                                                                                                                    |
| 60175 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60176 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60177 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60178 | IPC_RMID                                                                                            | Remove the semaphore identifier specified by <i>semid</i> from the system and destroy the set of semaphores and <b>semid_ds</b> data structure associated with it. This command can only be executed by a process that has an effective user ID equal to either that of a process with appropriate privileges or to the value of <i>sem_perm.cuid</i> or <i>sem_perm.uid</i> in the <b>semid_ds</b> data structure associated with <i>semid</i> . |
| 60179 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60180 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60181 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60182 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60183 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60184 | <b>RETURN VALUE</b>                                                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60185 | If successful, the value returned by <i>semctl()</i> depends on <i>cmd</i> as follows:              |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60186 | GETVAL                                                                                              | The value of <i>semval</i> .                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 60187 | GETPID                                                                                              | The value of <i>sempid</i> .                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 60188 | GETNCNT                                                                                             | The value of <i>semmcnt</i> .                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 60189 | GETZCNT                                                                                             | The value of <i>semzcnt</i> .                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 60190 | All others                                                                                          | 0.                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 60191 | Otherwise, <i>semctl()</i> shall return <code>-1</code> and set <i>errno</i> to indicate the error. |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60192 | <b>ERRORS</b>                                                                                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60193 | The <i>semctl()</i> function shall fail if:                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60194 | [EACCES]                                                                                            | Operation permission is denied to the calling process; see <a href="#">Section 2.7</a> (on page 501).                                                                                                                                                                                                                                                                                                                                             |
| 60195 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60196 | [EINVAL]                                                                                            | The value of <i>semid</i> is not a valid semaphore identifier, or the value of <i>semnum</i> is less than 0 or greater than or equal to <i>sem_nsems</i> , or the value of <i>cmd</i> is not a valid command.                                                                                                                                                                                                                                     |
| 60197 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60198 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60199 | [EPERM]                                                                                             | The argument <i>cmd</i> is equal to <code>IPC_RMID</code> or <code>IPC_SET</code> and the effective user ID of the calling process is not equal to that of a process with appropriate privileges and it is not equal to the value of <i>sem_perm.cuid</i> or <i>sem_perm.uid</i> in                                                                                                                                                               |
| 60200 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 60201 |                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                   |



60202 the data structure associated with *semid*.  
60203 [ERANGE] The argument *cmd* is equal to SETVAL or SETALL and the value to which  
60204 *semval* is to be set is greater than the system-imposed maximum.

#### 60205 EXAMPLES

60206 Refer to *semop()*.

#### 60207 APPLICATION USAGE

60208 The fourth parameter in the SYNOPSIS section is now specified as ". . ." in order to avoid a  
60209 clash with the ISO C standard when referring to the union *semun* (as defined in Issue 3) and for  
60210 backwards-compatibility.

60211 The POSIX Realtime Extension defines alternative interfaces for interprocess communication.  
60212 Application developers who need to use IPC should design their applications so that modules  
60213 using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the  
60214 alternative interfaces.

#### 60215 RATIONALE

60216 None.

#### 60217 FUTURE DIRECTIONS

60218 None.

#### 60219 SEE ALSO

60220 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 503), *semget()*, *semop()*, *sem\_close()*, *sem\_destroy()*,  
60221 *sem\_getvalue()*, *sem\_init()*, *sem\_open()*, *sem\_post()*, *sem\_trywait()*, *sem\_unlink()*

60222 XBD [Section 4.17](#) (on page 114), [<sys/sem.h>](#)

#### 60223 CHANGE HISTORY

60224 First released in Issue 2. Derived from Issue 2 of the SVID.

#### 60225 Issue 5

60226 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE  
60227 DIRECTIONS to the APPLICATION USAGE section.

#### 60228 Issue 7

60229 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0532 [345], XSH/TC1-2008/0533 [345],  
60230 XSH/TC1-2008/0534 [345], and XSH/TC1-2008/0535 [335] are applied.

**NAME**

semget — get set of XSI semaphores

**SYNOPSIS**

```
XSI    #include <sys/sem.h>
      int semget(key_t key, int nsems, int semflg);
```

**DESCRIPTION**

The *semget()* function operates on XSI semaphores (see XBD [Section 4.17](#), on page 114). It is unspecified whether this function interoperates with the realtime interprocess communication facilities defined in [Section 2.8](#) (on page 503).

The *semget()* function shall return the semaphore identifier associated with *key*.

A semaphore identifier with its associated **semid\_ds** data structure and its associated set of *nsems* semaphores (see **<sys/sem.h>**) is created for *key* if one of the following is true:

- The argument *key* is equal to `IPC_PRIVATE`.
- The argument *key* does not already have a semaphore identifier associated with it and (*semflg* & `IPC_CREAT`) is non-zero.

Upon creation, the **semid\_ds** data structure associated with the new semaphore identifier is initialized as follows:

- In the operation permissions structure *sem\_perm.cuid*, *sem\_perm.uid*, *sem\_perm.cgid*, and *sem\_perm.gid* shall be set to the effective user ID and effective group ID, respectively, of the calling process.
- The low-order 9 bits of *sem\_perm.mode* shall be set to the low-order 9 bits of *semflg*.
- The variable *sem\_nsems* shall be set to the value of *nsems*.
- The variable *sem\_otime* shall be set to 0 and *sem\_ctime* shall be set to the current time, as described in [Section 2.7.1](#) (on page 502).
- The data structure associated with each semaphore in the set need not be initialized. The *semctl()* function with the command `SETVAL` or `SETALL` can be used to initialize each semaphore.

**RETURN VALUE**

Upon successful completion, *semget()* shall return a non-negative integer, namely a semaphore identifier; otherwise, it shall return `-1` and set *errno* to indicate the error.

**ERRORS**

The *semget()* function shall fail if:

- |          |                                                                                                                                                                                                                                                                                               |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [EACCES] | A semaphore identifier exists for <i>key</i> , but operation permission as specified by the low-order 9 bits of <i>semflg</i> would not be granted; see <a href="#">Section 2.7</a> (on page 501).                                                                                            |
| [EEXIST] | A semaphore identifier exists for the argument <i>key</i> but (( <i>semflg</i> & <code>IPC_CREAT</code> ) && ( <i>semflg</i> & <code>IPC_EXCL</code> )) is non-zero.                                                                                                                          |
| [EINVAL] | The value of <i>nsems</i> is either less than or equal to 0 or greater than the system-imposed limit, or a semaphore identifier exists for the argument <i>key</i> , but the number of semaphores in the set associated with it is less than <i>nsems</i> and <i>nsems</i> is not equal to 0. |

60272 [ENOENT] A semaphore identifier does not exist for the argument *key* and (*semflg*  
60273 &IPC\_CREAT) is equal to 0.

60274 [ENOSPC] A semaphore identifier is to be created but the system-imposed limit on the  
60275 maximum number of allowed semaphores system-wide would be exceeded.

#### 60276 EXAMPLES

60277 Refer to *semop()*.

#### 60278 APPLICATION USAGE

60279 The POSIX Realtime Extension defines alternative interfaces for interprocess communication.  
60280 Application developers who need to use IPC should design their applications so that modules  
60281 using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the  
60282 alternative interfaces.

#### 60283 RATIONALE

60284 None.

#### 60285 FUTURE DIRECTIONS

60286 A future version may require that the value of the *semval*, *sempid*, *semncnt*, and *semzcnt* members  
60287 of all semaphores in a semaphore set be initialized to zero when a call to *semget()* creates a  
60288 semaphore set. Many semaphore implementations already do this and it greatly simplifies what  
60289 an application must do to initialize a semaphore set.

#### 60290 SEE ALSO

60291 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 503), *ftok()*, *semctl()*, *semop()*, *sem\_close()*,  
60292 *sem\_destroy()*, *sem\_getvalue()*, *sem\_init()*, *sem\_open()*, *sem\_post()*, *sem\_trywait()*, *sem\_unlink()*

60293 XBD [Section 4.17](#) (on page 114), [<sys/sem.h>](#)

#### 60294 CHANGE HISTORY

60295 First released in Issue 2. Derived from Issue 2 of the SVID.

#### 60296 Issue 5

60297 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE  
60298 DIRECTIONS to a new APPLICATION USAGE section.

#### 60299 Issue 6

60300 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/122 is applied, updating the  
60301 DESCRIPTION from “each semaphore in the set shall not be initialized” to “each semaphore in  
60302 the set need not be initialized”.

#### 60303 Issue 7

60304 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0536 [335,439] and  
60305 XSH/TC1-2008/0537 [344] are applied.

**NAME**

semop — XSI semaphore operations

**SYNOPSIS**

```
XSI    #include <sys/sem.h>
int semop(int semid, struct sembuf *sops, size_t nsops);
```

**DESCRIPTION**

The *semop()* function operates on XSI semaphores (see XBD [Section 4.17](#), on page 114). It is unspecified whether this function interoperates with the realtime interprocess communication facilities defined in [Section 2.8](#) (on page 503).

The *semop()* function shall perform atomically a user-defined array of semaphore operations in array order on the set of semaphores associated with the semaphore identifier specified by the argument *semid*.

The argument *sops* is a pointer to a user-defined array of semaphore operation structures. The implementation shall not modify elements of this array unless the application uses implementation-defined extensions.

The argument *nsops* is the number of such structures in the array.

Each structure, **sembuf**, includes the following members:

| Member Type    | Member Name    | Description          |
|----------------|----------------|----------------------|
| unsigned short | <i>sem_num</i> | Semaphore number.    |
| short          | <i>sem_op</i>  | Semaphore operation. |
| short          | <i>sem_flg</i> | Operation flags.     |

Each semaphore operation specified by *sem\_op* is performed on the corresponding semaphore specified by *semid* and *sem\_num*.

The variable *sem\_op* specifies one of three semaphore operations:

1. If *sem\_op* is a negative integer and the calling process has alter permission, one of the following shall occur:
  - If *semval* (see **<sys/sem.h>**) is greater than or equal to the absolute value of *sem\_op*, the absolute value of *sem\_op* is subtracted from *semval*. Also, if (*sem\_flg* & SEM\_UNDO) is non-zero, the absolute value of *sem\_op* shall be added to the *semadj* value of the calling process for the specified semaphore.
  - If *semval* is less than the absolute value of *sem\_op* and (*sem\_flg* & IPC\_NOWAIT) is non-zero, *semop()* shall return immediately.
  - If *semval* is less than the absolute value of *sem\_op* and (*sem\_flg* & IPC\_NOWAIT) is 0, *semop()* shall increment the *semncnt* associated with the specified semaphore and suspend execution of the calling thread until one of the following conditions occurs:
    - The value of *semval* becomes greater than or equal to the absolute value of *sem\_op*. When this occurs, the value of *semncnt* associated with the specified semaphore shall be decremented, the absolute value of *sem\_op* shall be subtracted from *semval* and, if (*sem\_flg* & SEM\_UNDO) is non-zero, the absolute value of *sem\_op* shall be added to the *semadj* value of the calling process for the specified semaphore.

- 60347 — The *semid* for which the calling thread is awaiting action is removed from the  
 60348 system. When this occurs, *errno* shall be set to [EIDRM] and -1 shall be  
 60349 returned.
- 60350 — The calling thread receives a signal that is to be caught. When this occurs, the  
 60351 value of *semncnt* associated with the specified semaphore shall be  
 60352 decremented, and the calling thread shall resume execution in the manner  
 60353 prescribed in [sigaction\(\)](#).
- 60354 2. If *sem\_op* is a positive integer and the calling process has alter permission, the value of  
 60355 *sem\_op* shall be added to *semval* and, if (*sem\_flg* & SEM\_UNDO) is non-zero, the value of  
 60356 *sem\_op* shall be subtracted from the *semadj* value of the calling process for the specified  
 60357 semaphore.
- 60358 3. If *sem\_op* is 0 and the calling process has read permission, one of the following shall occur:
- 60359 • If *semval* is 0, *semop()* shall return immediately.
- 60360 • If *semval* is non-zero and (*sem\_flg* & IPC\_NOWAIT) is non-zero, *semop()* shall return  
 60361 immediately.
- 60362 • If *semval* is non-zero and (*sem\_flg* & IPC\_NOWAIT) is 0, *semop()* shall increment the  
 60363 *semzcnt* associated with the specified semaphore and suspend execution of the  
 60364 calling thread until one of the following occurs:
- 60365 — The value of *semval* becomes 0, at which time the value of *semzcnt* associated  
 60366 with the specified semaphore shall be decremented.
- 60367 — The *semid* for which the calling thread is awaiting action is removed from the  
 60368 system. When this occurs, *errno* shall be set to [EIDRM] and -1 shall be  
 60369 returned.
- 60370 — The calling thread receives a signal that is to be caught. When this occurs, the  
 60371 value of *semzcnt* associated with the specified semaphore shall be  
 60372 decremented, and the calling thread shall resume execution in the manner  
 60373 prescribed in [sigaction\(\)](#).
- 60374 Upon successful completion, the value of *sempid* for each semaphore specified in the array  
 60375 pointed to by *sops* shall be set to the process ID of the calling process. Also, the *sem\_otime*  
 60376 timestamp shall be set to the current time, as described in [Section 2.7.1](#) (on page 502).

**60377 RETURN VALUE**

60378 Upon successful completion, *semop()* shall return 0; otherwise, it shall return -1 and set *errno* to  
 60379 indicate the error.

**60380 ERRORS**

60381 The *semop()* function shall fail if:

- 60382 [E2BIG] The value of *nsops* is greater than the system-imposed maximum.
- 60383 [EACCES] Operation permission is denied to the calling process; see [Section 2.7](#) (on page  
 60384 501).
- 60385 [EAGAIN] The operation would result in suspension of the calling process but (*sem\_flg*  
 60386 & IPC\_NOWAIT) is non-zero.
- 60387 [EFBIG] The value of *sem\_num* is greater than or equal to the number of semaphores in  
 60388 the set associated with *semid*.

|       |          |                                                                                                                                                                                                |
|-------|----------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 60389 | [EIDRM]  | The semaphore identifier <i>semid</i> is removed from the system.                                                                                                                              |
| 60390 | [EINTR]  | The <i>semop()</i> function was interrupted by a signal.                                                                                                                                       |
| 60391 | [EINVAL] | The value of <i>semid</i> is not a valid semaphore identifier, or the number of individual semaphores for which the calling process requests a SEM_UNDO would exceed the system-imposed limit. |
| 60392 |          |                                                                                                                                                                                                |
| 60393 |          |                                                                                                                                                                                                |
| 60394 | [ENOSPC] | The limit on the number of individual processes requesting a SEM_UNDO would be exceeded.                                                                                                       |
| 60395 |          |                                                                                                                                                                                                |
| 60396 | [ERANGE] | An operation would cause a <i>semval</i> to overflow the system-imposed limit, or an operation would cause a <i>semadj</i> value to overflow the system-imposed limit.                         |
| 60397 |          |                                                                                                                                                                                                |
| 60398 |          |                                                                                                                                                                                                |

## 60399 EXAMPLES

### 60400 Setting Values in Semaphores

60401 The following example sets the values of the two semaphores associated with the *semid* identifier  
60402 to the values contained in the *sb* array.

```
60403 #include <sys/sem.h>
60404 ...
60405 int semid;
60406 struct sembuf sb[2];
60407 int nsops = 2;
60408 int result;

60409 /* Code to initialize semid. */
60410 ...

60411 /* Adjust value of semaphore in the semaphore array semid. */
60412 sb[0].sem_num = 0;
60413 sb[0].sem_op = -1;
60414 sb[0].sem_flg = SEM_UNDO | IPC_NOWAIT;
60415 sb[1].sem_num = 1;
60416 sb[1].sem_op = 1;
60417 sb[1].sem_flg = 0;

60418 result = semop(semid, sb, nsops);
```

### 60419 Creating a Semaphore Identifier

60420 The following example gets a unique semaphore key using the *ftok()* function, then gets a  
60421 semaphore ID associated with that key using the *semget()* function (the first call also tests to  
60422 make sure the semaphore exists). If the semaphore does not exist, the program creates it, as  
60423 shown by the second call to *semget()*. In creating the semaphore for the queuing process, the  
60424 program attempts to create one semaphore with read/write permission for all. It also uses the  
60425 IPC\_EXCL flag, which forces *semget()* to fail if the semaphore already exists.

60426 After creating the semaphore, the program uses calls to *semctl()* and *semop()* to initialize it to the  
60427 values in the *sbuf* array. The number of processes that can execute concurrently without queuing  
60428 is initially set to 2. The final call to *semget()* creates a semaphore identifier that can be used later  
60429 in the program.

60430 Processes that obtain *semid* without creating it check that *sem\_otime* is non-zero, to ensure that  
60431 the creating process has completed the *semop()* initialization.

The final call to *semop()* acquires the semaphore and waits until it is free; the SEM\_UNDO option releases the semaphore when the process exits, waiting until there are less than two processes running concurrently.

```

60432
60433
60434
60435 #include <stdio.h>
60436 #include <sys/sem.h>
60437 #include <sys/stat.h>
60438 #include <errno.h>
60439 #include <stdlib.h>
60440 ...
60441 key_t semkey;
60442 int semid;
60443 struct sembuf sbuf;
60444 union semun {
60445     int val;
60446     struct semid_ds *buf;
60447     unsigned short *array;
60448 } arg;
60449 struct semid_ds ds;
60450 ...
60451 /* Get unique key for semaphore. */
60452 if ((semkey = ftok("/tmp", 'a')) == (key_t) -1) {
60453     perror("IPC error: ftok"); exit(1);
60454 }
60455
60456 /* Get semaphore ID associated with this key. */
60457 if ((semid = semget(semkey, 0, 0)) == -1) {
60458     /* Semaphore does not exist - Create. */
60459     if ((semid = semget(semkey, 1, IPC_CREAT | IPC_EXCL | S_IRUSR |
60460         S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH)) != -1)
60461     {
60462         /* Initialize the semaphore. */
60463         arg.val = 0;
60464         sbuf.sem_num = 0;
60465         sbuf.sem_op = 2; /* This is the number of runs without queuing. */
60466         sbuf.sem_flg = 0;
60467         if (semctl(semid, 0, SETVAL, arg) == -1
60468             || semop(semid, &sbuf, 1) == -1) {
60469             perror("IPC error: semop"); exit(1);
60470         }
60471     }
60472     else if (errno == EEXIST) {
60473         if ((semid = semget(semkey, 0, 0)) == -1) {
60474             perror("IPC error 1: semget"); exit(1);
60475         }
60476         goto check_init;
60477     }
60478     else {
60479         perror("IPC error 2: semget"); exit(1);
60480     }
60481 }
else

```



```

60482     {
60483         /* Check that semid has completed initialization. */
60484         /* An application can use a retry loop at this point rather than
60485            exiting. */
60486         check_init:
60487         arg.buf = &ds;
60488         if (semctl(semid, 0, IPC_STAT, arg) < 0) {
60489             perror("IPC error 3: semctl"); exit(1);
60490         }
60491         if (ds.sem_otime == 0) {
60492             perror("IPC error 4: semctl"); exit(1);
60493         }
60494     }
60495     ...
60496     sbuf.sem_num = 0;
60497     sbuf.sem_op = -1;
60498     sbuf.sem_flg = SEM_UNDO;
60499     if (semop(semid, &sbuf, 1) == -1) {
60500         perror("IPC Error: semop"); exit(1);
60501     }

```

## APPLICATION USAGE

The POSIX Realtime Extension defines alternative interfaces for interprocess communication. Application developers who need to use IPC should design their applications so that modules using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the alternative interfaces.

## RATIONALE

None.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[Section 2.7](#) (on page 501), [Section 2.8](#) (on page 503), [exec](#), [exit\(\)](#), [fork\(\)](#), [semctl\(\)](#), [semget\(\)](#), [sem\\_close\(\)](#), [sem\\_destroy\(\)](#), [sem\\_getvalue\(\)](#), [sem\\_init\(\)](#), [sem\\_open\(\)](#), [sem\\_post\(\)](#), [sem\\_trywait\(\)](#), [sem\\_unlink\(\)](#)

XBD [Section 4.17](#) (on page 114), [<sys/ipc.h>](#), [<sys/sem.h>](#), [<sys/types.h>](#)

## CHANGE HISTORY

First released in Issue 2. Derived from Issue 2 of the SVID.

### Issue 5

The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE DIRECTIONS to a new APPLICATION USAGE section.

### Issue 7

SD5-XSH-ERN-171 is applied, updating the DESCRIPTION to clarify the order in which the operations in *sops* will be performed when there are multiple operations.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0538 [329,429], XSH/TC1-2008/0539 [345,428], XSH/TC1-2008/0540 [329,429], XSH/TC1-2008/0541 [335], and XSH/TC1-2008/0542 [291,429] are applied.



**NAME**

send — send a message on a socket

**SYNOPSIS**

```
#include <sys/socket.h>
```

```
ssize_t send(int socket, const void *buffer, size_t length, int flags);
```

**DESCRIPTION**

The *send()* function shall initiate transmission of a message from the specified socket to its peer. The *send()* function shall send a message only when the socket is connected. If the socket is a connectionless-mode socket, the message shall be sent to the pre-specified peer address.

The *send()* function takes the following arguments:

*socket* Specifies the socket file descriptor.

*buffer* Points to the buffer containing the message to send.

*length* Specifies the length of the message in bytes.

*flags* Specifies the type of message transmission. Values of this argument are formed by logically OR'ing zero or more of the following flags:

MSG\_EOR Terminates a record (if supported by the protocol).

MSG\_OOB Sends out-of-band data on sockets that support out-of-band communications. The significance and semantics of out-of-band data are protocol-specific.

MSG\_NOSIGNAL Requests not to send the SIGPIPE signal if an attempt to send is made on a stream-oriented socket that is no longer connected. The [EPIPE] error shall still be returned.

The length of the message to be sent is specified by the *length* argument. If the message is too long to pass through the underlying protocol, *send()* shall fail and no data shall be transmitted.

Successful completion of a call to *send()* does not guarantee delivery of the message. A return value of -1 indicates only locally-detected errors.

If space is not available at the sending socket to hold the message to be transmitted, and the socket file descriptor does not have O\_NONBLOCK set, *send()* shall block until space is available. If space is not available at the sending socket to hold the message to be transmitted, and the socket file descriptor does have O\_NONBLOCK set, *send()* shall fail. The *select()* and *poll()* functions can be used to determine when it is possible to send more data.

The socket in use may require the process to have appropriate privileges to use the *send()* function.

**RETURN VALUE**

Upon successful completion, *send()* shall return the number of bytes sent. Otherwise, -1 shall be returned and *errno* set to indicate the error.

**ERRORS**

The *send()* function shall fail if:

[EAGAIN] or [EWOULDBLOCK]

The socket's file descriptor is marked O\_NONBLOCK and the requested operation would block.

|       |                |                                                                                                                                                                                                                                                                               |
|-------|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 60569 | [EBADF]        | The <i>socket</i> argument is not a valid file descriptor.                                                                                                                                                                                                                    |
| 60570 | [ECONNRESET]   | A connection was forcibly closed by a peer.                                                                                                                                                                                                                                   |
| 60571 | [EDESTADDRREQ] |                                                                                                                                                                                                                                                                               |
| 60572 |                | The socket is not connection-mode and no peer address is set.                                                                                                                                                                                                                 |
| 60573 | [EINTR]        | A signal interrupted <i>send()</i> before any data was transmitted.                                                                                                                                                                                                           |
| 60574 | [EMSGSIZE]     | The message is too large to be sent all at once, as the socket requires.                                                                                                                                                                                                      |
| 60575 | [ENOTCONN]     | The socket is not connected.                                                                                                                                                                                                                                                  |
| 60576 | [ENOTSOCK]     | The <i>socket</i> argument does not refer to a socket.                                                                                                                                                                                                                        |
| 60577 | [EOPNOTSUPP]   | The <i>socket</i> argument is associated with a socket that does not support one or more of the values set in <i>flags</i> .                                                                                                                                                  |
| 60578 |                |                                                                                                                                                                                                                                                                               |
| 60579 | [EPIPE]        | The socket is shut down for writing, or the socket is connection-mode and is no longer connected. In the latter case, and if the socket is of type SOCK_STREAM or SOCK_SEQPACKET and the MSG_NOSIGNAL flag is not set, the SIGPIPE signal is generated to the calling thread. |
| 60580 |                |                                                                                                                                                                                                                                                                               |
| 60581 |                |                                                                                                                                                                                                                                                                               |
| 60582 |                |                                                                                                                                                                                                                                                                               |
| 60583 |                | The <i>send()</i> function may fail if:                                                                                                                                                                                                                                       |
| 60584 | [EACCES]       | The calling process does not have appropriate privileges.                                                                                                                                                                                                                     |
| 60585 | [EIO]          | An I/O error occurred while reading from or writing to the file system.                                                                                                                                                                                                       |
| 60586 | [ENETDOWN]     | The local network interface used to reach the destination is down.                                                                                                                                                                                                            |
| 60587 | [ENETUNREACH]  |                                                                                                                                                                                                                                                                               |
| 60588 |                | No route to the network is present.                                                                                                                                                                                                                                           |
| 60589 | [ENOBUFS]      | Insufficient resources were available in the system to perform the operation.                                                                                                                                                                                                 |

**EXAMPLES**

None.

**APPLICATION USAGE**

If the *socket* argument refers to a connection-mode socket, the *send()* function is equivalent to *sendto()* (with any value for the *dest\_addr* and *dest\_len* arguments, as they are ignored in this case). If the *socket* argument refers to a socket and the *flags* argument is 0, the *send()* function is equivalent to *write()*.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*connect()*, *getsockopt()*, *poll()*, *pselect()*, *recv()*, *recvfrom()*, *recvmsg()*, *sendmsg()*, *sendto()*, *setsockopt()*, *shutdown()*, *socket()*, *write()*

XBD &lt;sys/socket.h&gt;

**CHANGE HISTORY**

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

60607 **Issue 7**

60608 Austin Group Interpretation 1003.1-2001 #035 is applied, updating the DESCRIPTION to clarify  
60609 the behavior when the socket is a connectionless-mode socket.

60610 The MSG\_NOSIGNAL flag is added from The Open Group Technical Standard, 2006, Extended  
60611 API Set Part 2.

60612 The [EPIPE] error is modified.

60613 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0543 [463] is applied.

## NAME

sendmsg — send a message on a socket using a message structure

## SYNOPSIS

```
#include <sys/socket.h>
```

```
ssize_t sendmsg(int socket, const struct msghdr *message, int flags);
```

## DESCRIPTION

The *sendmsg()* function shall send a message through a connection-mode or connectionless-mode socket. If the socket is a connectionless-mode socket, the message shall be sent to the address specified by **msghdr** if no pre-specified peer address has been set. If a peer address has been pre-specified, either the message shall be sent to the address specified in **msghdr** (overriding the pre-specified peer address), or the function shall return `-1` and set *errno* to `[EISCONN]`. If the socket is connection-mode, the destination address in **msghdr** shall be ignored.

The *sendmsg()* function takes the following arguments:

|                |                                                                                                                                                                                                                                                 |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>socket</i>  | Specifies the socket file descriptor.                                                                                                                                                                                                           |
| <i>message</i> | Points to a <b>msghdr</b> structure, containing both the destination address and the buffers for the outgoing message. The length and format of the address depend on the address family of the socket. The <i>msg_flags</i> member is ignored. |
| <i>flags</i>   | Specifies the type of message transmission. The application may specify 0 or the following flag:                                                                                                                                                |
| MSG_EOR        | Terminates a record (if supported by the protocol).                                                                                                                                                                                             |
| MSG_OOB        | Sends out-of-band data on sockets that support out-of-band data. The significance and semantics of out-of-band data are protocol-specific.                                                                                                      |
| MSG_NOSIGNAL   | Requests not to send the SIGPIPE signal if an attempt to send is made on a stream-oriented socket that is no longer connected. The <code>[EPIPE]</code> error shall still be returned.                                                          |

The *msg\_iov* and *msg\_iovlen* fields of *message* specify zero or more buffers containing the data to be sent. *msg\_iov* points to an array of **iovec** structures; *msg\_iovlen* shall be set to the dimension of this array. In each **iovec** structure, the *iov\_base* field specifies a storage area and the *iov\_len* field gives its size in bytes. Some of these sizes can be zero. The data from each storage area indicated by *msg\_iov* is sent in turn.

Successful completion of a call to *sendmsg()* does not guarantee delivery of the message. A return value of `-1` indicates only locally-detected errors.

If space is not available at the sending socket to hold the message to be transmitted and the socket file descriptor does not have `O_NONBLOCK` set, the *sendmsg()* function shall block until space is available. If space is not available at the sending socket to hold the message to be transmitted and the socket file descriptor does have `O_NONBLOCK` set, the *sendmsg()* function shall fail.

If the socket protocol supports broadcast and the specified address is a broadcast address for the socket protocol, *sendmsg()* shall fail if the `SO_BROADCAST` option is not set for the socket.

The socket in use may require the process to have appropriate privileges to use the *sendmsg()* function.

**RETURN VALUE**

Upon successful completion, *sendmsg()* shall return the number of bytes sent. Otherwise, *-1* shall be returned and *errno* set to indicate the error.

**ERRORS**

The *sendmsg()* function shall fail if:

[EAGAIN] or [EWOULDBLOCK]

The socket's file descriptor is marked *O\_NONBLOCK* and the requested operation would block.

[EAFNOSUPPORT]

Addresses in the specified address family cannot be used with this socket.

[EBADF]

The *socket* argument is not a valid file descriptor.

[ECONNRESET] A connection was forcibly closed by a peer.

[EINTR]

A signal interrupted *sendmsg()* before any data was transmitted.

[EINVAL]

The sum of the *iov\_len* values overflows an *ssize\_t*.

[EMSGSIZE]

The message is too large to be sent all at once (as the socket requires), or the *msg\_iovlen* member of the **msghdr** structure pointed to by *message* is less than or equal to 0 or is greater than {*IOV\_MAX*}.

[ENOTCONN]

The socket is connection-mode but is not connected.

[ENOTSOCK]

The *socket* argument does not refer to a socket.

[EOPNOTSUPP]

The *socket* argument is associated with a socket that does not support one or more of the values set in *flags*.

[EPIPE]

The socket is shut down for writing, or the socket is connection-mode and is no longer connected. In the latter case, and if the socket is of type *SOCK\_STREAM* or *SOCK\_SEQPACKET* and the *MSG\_NOSIGNAL* flag is not set, the *SIGPIPE* signal is generated to the calling thread.

If the address family of the socket is *AF\_UNIX*, then *sendmsg()* shall fail if:

[EIO]

An I/O error occurred while reading from or writing to the file system.

[ELOOP]

A loop exists in symbolic links encountered during resolution of the pathname in the socket address.

[ENAMETOOLONG]

The length of a component of a pathname is longer than {*NAME\_MAX*}.

[ENOENT]

A component of the pathname does not name an existing file or the path name is an empty string.

[ENOTDIR]

A component of the path prefix of the pathname in the socket address names an existing file that is neither a directory nor a symbolic link to a directory, or the pathname in the socket address contains at least one non-*<slash>* character and ends with one or more trailing *<slash>* characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.

60697 The *sendmsg()* function may fail if:

60698 [EACCES] Search permission is denied for a component of the path prefix; or write access  
60699 to the named socket is denied.

60700 [EDESTADDRREQ]  
60701 The socket is not connection-mode and does not have its peer address set, and  
60702 no destination address was specified.

60703 [EHOSTUNREACH]  
60704 The destination host cannot be reached (probably because the host is down or  
60705 a remote router cannot reach it).

60706 [EIO] An I/O error occurred while reading from or writing to the file system.

60707 [EISCONN] A destination address was specified and the socket is already connected.

60708 [ENETDOWN] The local network interface used to reach the destination is down.

60709 [ENETUNREACH]  
60710 No route to the network is present.

60711 [ENOBUFS] Insufficient resources were available in the system to perform the operation.

60712 [ENOMEM] Insufficient memory was available to fulfill the request.

60713 If the address family of the socket is AF\_UNIX, then *sendmsg()* may fail if:

60714 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
60715 resolution of the pathname in the socket address.

60716 [ENAMETOOLONG]  
60717 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
60718 symbolic link produced an intermediate result with a length that exceeds  
60719 {PATH\_MAX}.

## 60720 EXAMPLES

60721 Done.

## 60722 APPLICATION USAGE

60723 The *select()* and *poll()* functions can be used to determine when it is possible to send more data.

## 60724 RATIONALE

60725 None.

## 60726 FUTURE DIRECTIONS

60727 None.

## 60728 SEE ALSO

60729 *getsockopt()*, *poll()*, *pselect()*, *recv()*, *recvfrom()*, *recvmsg()*, *send()*, *sendto()*, *setsockopt()*,  
60730 *shutdown()*, *socket()*

60731 XBD <sys/socket.h>

## 60732 CHANGE HISTORY

60733 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

60734 The wording of the mandatory [ELOOP] error condition is updated, and a second optional  
60735 [ELOOP] error condition is added.

60736 **Issue 7**

60737 Austin Group Interpretation 1003.1-2001 #073 is applied, updating the DESCRIPTION.

60738 Austin Group Interpretation 1003.1-2001 #143 is applied.

60739 The MSG\_NOSIGNAL flag is added from The Open Group Technical Standard, 2006, Extended  
60740 API Set Part 2.

60741 The [EPIPE] error is modified.

60742 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0544 [324] is applied.

**NAME**

sendto — send a message on a socket

**SYNOPSIS**

```
#include <sys/socket.h>

ssize_t sendto(int socket, const void *message, size_t length,
               int flags, const struct sockaddr *dest_addr,
               socklen_t dest_len);
```

**DESCRIPTION**

The *sendto()* function shall send a message through a connection-mode or connectionless-mode socket.

If the socket is a connectionless-mode socket, the message shall be sent to the address specified by *dest\_addr* if no pre-specified peer address has been set. If a peer address has been pre-specified, either the message shall be sent to the address specified by *dest\_addr* (overriding the pre-specified peer address), or the function shall return  $-1$  and set *errno* to [EISCONN].

If the socket is connection-mode, *dest\_addr* shall be ignored.

The *sendto()* function takes the following arguments:

|                  |                                                                                                                                                                           |
|------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>socket</i>    | Specifies the socket file descriptor.                                                                                                                                     |
| <i>message</i>   | Points to a buffer containing the message to be sent.                                                                                                                     |
| <i>length</i>    | Specifies the size of the message in bytes.                                                                                                                               |
| <i>flags</i>     | Specifies the type of message transmission. Values of this argument are formed by logically OR'ing zero or more of the following flags:                                   |
| MSG_EOR          | Terminates a record (if supported by the protocol).                                                                                                                       |
| MSG_OOB          | Sends out-of-band data on sockets that support out-of-band data. The significance and semantics of out-of-band data are protocol-specific.                                |
| MSG_NOSIGNAL     | Requests not to send the SIGPIPE signal if an attempt to send is made on a stream-oriented socket that is no longer connected. The [EPIPE] error shall still be returned. |
| <i>dest_addr</i> | Points to a <b>sockaddr</b> structure containing the destination address. The length and format of the address depend on the address family of the socket.                |
| <i>dest_len</i>  | Specifies the length of the <b>sockaddr</b> structure pointed to by the <i>dest_addr</i> argument.                                                                        |

If the socket protocol supports broadcast and the specified address is a broadcast address for the socket protocol, *sendto()* shall fail if the SO\_BROADCAST option is not set for the socket.

The *dest\_addr* argument specifies the address of the target.

The *length* argument specifies the length of the message.

Successful completion of a call to *sendto()* does not guarantee delivery of the message. A return value of  $-1$  indicates only locally-detected errors.

If space is not available at the sending socket to hold the message to be transmitted and the socket file descriptor does not have O\_NONBLOCK set, *sendto()* shall block until space is available. If space is not available at the sending socket to hold the message to be transmitted



60785 and the socket file descriptor does have O\_NONBLOCK set, *sendto()* shall fail.

60786 The socket in use may require the process to have appropriate privileges to use the *sendto()*

60787 function.

60788 **RETURN VALUE**

60789 Upon successful completion, *sendto()* shall return the number of bytes sent. Otherwise, -1 shall

60790 be returned and *errno* set to indicate the error.

60791 **ERRORS**

60792 The *sendto()* function shall fail if:

60793 [EAFNOSUPPORT]

60794 Addresses in the specified address family cannot be used with this socket.

60795 [EAGAIN] or [EWOULDBLOCK]

60796 The socket's file descriptor is marked O\_NONBLOCK and the requested

60797 operation would block.

60798 [EBADF] The *socket* argument is not a valid file descriptor.

60799 [ECONNRESET] A connection was forcibly closed by a peer.

60800 [EINTR] A signal interrupted *sendto()* before any data was transmitted.

60801 [EMSGSIZE] The message is too large to be sent all at once, as the socket requires.

60802 [ENOTCONN] The socket is connection-mode but is not connected.

60803 [ENOTSOCK] The *socket* argument does not refer to a socket.

60804 [EOPNOTSUPP] The *socket* argument is associated with a socket that does not support one or

60805 more of the values set in *flags*.

60806 [EPIPE] The socket is shut down for writing, or the socket is connection-mode and is

60807 no longer connected. In the latter case, and if the socket is of type

60808 SOCK\_STREAM or SOCK\_SEQPACKET and the MSG\_NOSIGNAL flag is not

60809 set, the SIGPIPE signal is generated to the calling thread.

60810 If the address family of the socket is AF\_UNIX, then *sendto()* shall fail if:

60811 [EIO] An I/O error occurred while reading from or writing to the file system.

60812 [ELOOP] A loop exists in symbolic links encountered during resolution of the pathname

60813 in the socket address.

60814 [ENAMETOOLONG]

60815 The length of a component of a pathname is longer than {NAME\_MAX}.

60816 [ENOENT] A component of the pathname does not name an existing file or the pathname

60817 is an empty string.

60818 [ENOTDIR] A component of the path prefix of the pathname in the socket address names

60819 an existing file that is neither a directory nor a symbolic link to a directory, or

60820 the pathname in the socket address contains at least one non-*<slash>* character

60821 and ends with one or more trailing *<slash>* characters and the last pathname

60822 component names an existing file that is neither a directory nor a symbolic

60823 link to a directory.

60824 The *sendto()* function may fail if:

60825 [EACCES] Search permission is denied for a component of the path prefix; or write access  
60826 to the named socket is denied.

60827 [EDESTADDRREQ]  
60828 The socket is not connection-mode and does not have its peer address set, and  
60829 no destination address was specified.

60830 [EHOSTUNREACH]  
60831 The destination host cannot be reached (probably because the host is down or  
60832 a remote router cannot reach it).

60833 [EINVAL] The *dest\_len* argument is not a valid length for the address family.

60834 [EIO] An I/O error occurred while reading from or writing to the file system.

60835 [EISCONN] A destination address was specified and the socket is already connected.

60836 [ENETDOWN] The local network interface used to reach the destination is down.

60837 [ENETUNREACH]  
60838 No route to the network is present.

60839 [ENOBUFS] Insufficient resources were available in the system to perform the operation.

60840 [ENOMEM] Insufficient memory was available to fulfill the request.

60841 If the address family of the socket is AF\_UNIX, then *sendto()* may fail if:

60842 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
60843 resolution of the pathname in the socket address.

60844 [ENAMETOOLONG]  
60845 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
60846 symbolic link produced an intermediate result with a length that exceeds  
60847 {PATH\_MAX}.

## 60848 EXAMPLES

60849 None.

## 60850 APPLICATION USAGE

60851 The *select()* and *poll()* functions can be used to determine when it is possible to send more data.

## 60852 RATIONALE

60853 None.

## 60854 FUTURE DIRECTIONS

60855 None.

## 60856 SEE ALSO

60857 *getsockopt()*, *poll()*, *pselect()*, *recv()*, *recvfrom()*, *recvmsg()*, *send()*, *sendmsg()*, *setsockopt()*,  
60858 *shutdown()*, *socket()*

60859 XBD <sys/socket.h>

## 60860 CHANGE HISTORY

60861 First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

60862 The wording of the mandatory [ELOOP] error condition is updated, and a second optional  
60863 [ELOOP] error condition is added.

60864 **Issue 7**

60865 Austin Group Interpretations 1003.1-2001 #035 and #073 are applied, updating the [EISCONN]  
60866 error and the DESCRIPTION.

60867 Austin Group Interpretation 1003.1-2001 #143 is applied, clarifying the [ENAMETOOLONG]  
60868 error condition.

60869 The MSG\_NOSIGNAL flag is added from The Open Group Technical Standard, 2006, Extended  
60870 API Set Part 2.

60871 The [EPIPE] error is modified.

60872 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0545 [324] is applied.

60873 **NAME**

60874       setbuf — assign buffering to a stream

60875 **SYNOPSIS**

60876       #include &lt;stdio.h&gt;

60877       void setbuf(FILE \*restrict stream, char \*restrict buf);

60878 **DESCRIPTION**

60879 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
60880 conflict between the requirements described here and the ISO C standard is unintentional. This  
60881 volume of POSIX.1-2008 defers to the ISO C standard.

60882       Except that it returns no value, the function call:

60883       setbuf(stream, buf)

60884       shall be equivalent to:

60885       setvbuf(stream, buf, \_IOFBF, BUFSIZ)

60886       if *buf* is not a null pointer, or to:

60887       setvbuf(stream, buf, \_IONBF, BUFSIZ)

60888       if *buf* is a null pointer.60889 **RETURN VALUE**60890       The *setbuf()* function shall not return a value.60891 **ERRORS**

60892       Although the *setvbuf()* interface may set *errno* in defined ways, the value of *errno* after a call to  
60893 *setbuf()* is unspecified.

60894 **EXAMPLES**

60895       None.

60896 **APPLICATION USAGE**

60897       A common source of error is allocating buffer space as an “automatic” variable in a code block,  
60898 and then failing to close the stream in the same block.

60899       With *setbuf()*, allocating a buffer of BUFSIZ bytes does not necessarily imply that all of BUFSIZ  
60900 bytes are used for the buffer area.

60901       Since *errno* is not required to be unchanged on success, in order to correctly detect and possibly  
60902 recover from errors, applications should use *setvbuf()* instead of *setbuf()*.

60903 **RATIONALE**

60904       None.

60905 **FUTURE DIRECTIONS**

60906       None.

60907 **SEE ALSO**60908       Section 2.5 (on page 495), *fopen()*, *setvbuf()*

60909       XBD &lt;stdio.h&gt;

60910 **CHANGE HISTORY**

60911       First released in Issue 1. Derived from Issue 1 of the SVID.

60912 **Issue 6**

60913 The prototype for *setbuf()* is updated for alignment with the ISO/IEC 9899:1999 standard.

60914 **Issue 7**

60915 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0546 [397], XSH/TC1-2008/0547 [397],  
60916 and XSH/TC1-2008/0548 [14] are applied.

**NAME**

setegid — set the effective group ID

**SYNOPSIS**

```
#include <unistd.h>
```

```
int setegid(gid_t gid);
```

**DESCRIPTION**

If *gid* is equal to the real group ID or the saved set-group-ID, or if the process has appropriate privileges, *setegid()* shall set the effective group ID of the calling process to *gid*; the real group ID, saved set-group-ID, and any supplementary group IDs shall remain unchanged.

The *setegid()* function shall not affect the supplementary group list in any way.

**RETURN VALUE**

Upon successful completion, 0 shall be returned; otherwise, -1 shall be returned and *errno* set to indicate the error.

**ERRORS**

The *setegid()* function shall fail if:

[EINVAL] The value of the *gid* argument is invalid and is not supported by the implementation.

[EPERM] The process does not have appropriate privileges and *gid* does not match the real group ID or the saved set-group-ID.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

Refer to the RATIONALE section in *setuid()*.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*exec*, *getegid()*, *geteuid()*, *getgid()*, *getuid()*, *seteuid()*, *setgid()*, *setregid()*, *setreuid()*, *setuid()*

XBD *<sys/types.h>*, *<unistd.h>*

**CHANGE HISTORY**

First released in Issue 6. Derived from the IEEE P1003.1a draft standard.

**NAME**

setenv — add or change environment variable

**SYNOPSIS**

```
CX      #include <stdlib.h>
        int setenv(const char *envname, const char *envval, int overwrite);
```

**DESCRIPTION**

The *setenv()* function shall update or add a variable in the environment of the calling process. The *envname* argument points to a string containing the name of an environment variable to be added or altered. The environment variable shall be set to the value to which *envval* points. The function shall fail if *envname* points to a string which contains an '=' character. If the environment variable named by *envname* already exists and the value of *overwrite* is non-zero, the function shall return success and the environment shall be updated. If the environment variable named by *envname* already exists and the value of *overwrite* is zero, the function shall return success and the environment shall remain unchanged.

The *setenv()* function shall update the list of pointers to which *environ* points.

The strings described by *envname* and *envval* are copied by this function.

The *setenv()* function need not be thread-safe.

**RETURN VALUE**

Upon successful completion, zero shall be returned. Otherwise, -1 shall be returned, *errno* set to indicate the error, and the environment shall be unchanged.

**ERRORS**

The *setenv()* function shall fail if:

- |          |                                                                                                          |
|----------|----------------------------------------------------------------------------------------------------------|
| [EINVAL] | The <i>envname</i> argument points to an empty string or points to a string containing an '=' character. |
| [ENOMEM] | Insufficient memory was available to add a variable or its value to the environment.                     |

**EXAMPLES**

None.

**APPLICATION USAGE**

See *exec()* for restrictions on changing the environment in multi-threaded applications.

**RATIONALE**

Unanticipated results may occur if *setenv()* changes the external variable *environ*. In particular, if the optional *envp* argument to *main()* is present, it is not changed, and thus may point to an obsolete copy of the environment (as may any other copy of *environ*). However, other than the aforementioned restriction, the standard developers intended that the traditional method of walking through the environment by way of the *environ* pointer must be supported.

It was decided that *setenv()* should be required by this version because it addresses a piece of missing functionality, and does not impose a significant burden on the implementor.

There was considerable debate as to whether the System V *putenv()* function or the BSD *setenv()* function should be required as a mandatory function. The *setenv()* function was chosen because it permitted the implementation of the *unsetenv()* function to delete environmental variables, without specifying an additional interface. The *putenv()* function is available as part of the XSI option.

60992 The standard developers considered requiring that *setenv()* indicate an error when a call to it  
60993 would result in exceeding {ARG\_MAX}. The requirement was rejected since the condition might  
60994 be temporary, with the application eventually reducing the environment size. The ultimate  
60995 success or failure depends on the size at the time of a call to *exec*, which returns an indication of  
60996 this error condition.

60997 See also the RATIONALE section in *getenv()*.

#### 60998 FUTURE DIRECTIONS

60999 None.

#### 61000 SEE ALSO

61001 *exec*, *getenv()*, *putenv()*, *unsetenv()*

61002 XBD *<stdlib.h>*, *<sys/types.h>*, *<unistd.h>*

#### 61003 CHANGE HISTORY

61004 First released in Issue 6. Derived from the IEEE P1003.1a draft standard.

61005 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/55 is applied, adding references to *exec* in  
61006 the APPLICATION USAGE and SEE ALSO sections.

#### 61007 Issue 7

61008 Austin Group Interpretation 1003.1-2001 #156 is applied.

61009 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0549 [167], XSH/TC1-2008/0550 [185],  
61010 XSH/TC1-2008/0551 [167], and XSH/TC1-2008/0552 [38] are applied.



61011 **NAME**

61012        seteuid — set effective user ID

61013 **SYNOPSIS**

61014        #include &lt;unistd.h&gt;

61015        int seteuid(uid\_t uid);

61016 **DESCRIPTION**

61017        If *uid* is equal to the real user ID or the saved set-user-ID, or if the process has appropriate  
 61018        privileges, *seteuid()* shall set the effective user ID of the calling process to *uid*; the real user ID  
 61019        and saved set-user-ID shall remain unchanged.

61020        The *seteuid()* function shall not affect the supplementary group list in any way.

61021 **RETURN VALUE**

61022        Upon successful completion, 0 shall be returned; otherwise, -1 shall be returned and *errno* set to  
 61023        indicate the error.

61024 **ERRORS**

61025        The *seteuid()* function shall fail if:

61026        [EINVAL]        The value of the *uid* argument is invalid and is not supported by the  
 61027        implementation.

61028        [EPERM]        The process does not have appropriate privileges and *uid* does not match the  
 61029        real user ID or the saved set-user-ID.

61030 **EXAMPLES**

61031        None.

61032 **APPLICATION USAGE**

61033        None.

61034 **RATIONALE**61035        Refer to the RATIONALE section in *setuid()*.61036 **FUTURE DIRECTIONS**

61037        None.

61038 **SEE ALSO**61039        *exec*, *getegid()*, *geteuid()*, *getgid()*, *getuid()*, *setegid()*, *setgid()*, *setregid()*, *setreuid()*, *setuid()*

61040        XBD &lt;sys/types.h&gt;, &lt;unistd.h&gt;

61041 **CHANGE HISTORY**

61042        First released in Issue 6. Derived from the IEEE P1003.1a draft standard.

61043        IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/123 is applied, making an editorial  
 61044        correction to the [EPERM] error in the ERRORS section.

61045 **NAME**

61046       setgid — set-group-ID

61047 **SYNOPSIS**

61048       #include &lt;unistd.h&gt;

61049       int setgid(gid\_t gid);

61050 **DESCRIPTION**61051       If the process has appropriate privileges, *setgid()* shall set the real group ID, effective group ID,  
61052       and the saved set-group-ID of the calling process to *gid*.61053       If the process does not have appropriate privileges, but *gid* is equal to the real group ID or the  
61054       saved set-group-ID, *setgid()* shall set the effective group ID to *gid*; the real group ID and saved  
61055       set-group-ID shall remain unchanged.61056       The *setgid()* function shall not affect the supplementary group list in any way.

61057       Any supplementary group IDs of the calling process shall remain unchanged.

61058 **RETURN VALUE**61059       Upon successful completion, 0 is returned. Otherwise, -1 shall be returned and *errno* set to  
61060       indicate the error.61061 **ERRORS**61062       The *setgid()* function shall fail if:61063       [EINVAL]       The value of the *gid* argument is invalid and is not supported by the  
61064       implementation.61065       [EPERM]       The process does not have appropriate privileges and *gid* does not match the  
61066       real group ID or the saved set-group-ID.61067 **EXAMPLES**

61068       None.

61069 **APPLICATION USAGE**

61070       None.

61071 **RATIONALE**61072       Refer to the RATIONALE section in *setuid()*.61073 **FUTURE DIRECTIONS**

61074       None.

61075 **SEE ALSO**61076       *exec*, *getegid()*, *geteuid()*, *getgid()*, *getuid()*, *setegid()*, *seteuid()*, *setregid()*, *setreuid()*, *setuid()*

61077       XBD &lt;sys/types.h&gt;, &lt;unistd.h&gt;

61078 **CHANGE HISTORY**

61079       First released in Issue 1. Derived from Issue 1 of the SVID.

61080 **Issue 6**

61081       In the SYNOPSIS, the optional include of the &lt;sys/types.h&gt; header is removed.

61082       The following new requirements on POSIX implementations derive from alignment with the  
61083       Single UNIX Specification:

- 61084
- The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was  
61085       required for conforming implementations of previous POSIX specifications, it was not  
61086       required for UNIX applications.

- 61087
- 61088
- Functionality associated with `_POSIX_SAVED_IDS` is now mandated. This is a FIPS requirement.
- 61089
- The following changes were made to align with the IEEE P1003.1a draft standard:
- 61090
- The effects of `setgid()` in processes without appropriate privileges are changed.
- 61091
- A requirement that the supplementary group list is not affected is added.

61092 **NAME**

61093       setgrent — reset the group database to the first entry

61094 **SYNOPSIS**

```
61095 XSI      #include <grp.h>  
61096          void setgrent(void);
```

61097 **DESCRIPTION**61098       Refer to *endgrent()*.

61099 **NAME**

61100 sethostent — network host database functions

61101 **SYNOPSIS**

61102 #include &lt;netdb.h&gt;

61103 void sethostent(int *stayopen*);61104 **DESCRIPTION**61105 Refer to *endhostent()*.

**61106 NAME**

61107       setitimer — set the value of an interval timer

**61108 SYNOPSIS**

61109 OB XSI   #include <sys/time.h>

```
61110       int setitimer(int which, const struct itimerval *restrict value,  
61111                    struct itimerval *restrict ovalue);
```

**61112 DESCRIPTION**

61113       Refer to *getitimer()*.

61114 **NAME**

61115       setjmp — set jump point for a non-local goto

61116 **SYNOPSIS**

61117       #include &lt;setjmp.h&gt;

61118       int setjmp(jmp\_buf env);

61119 **DESCRIPTION**

61120 CX     The functionality described on this reference page is aligned with the ISO C standard. Any  
 61121     conflict between the requirements described here and the ISO C standard is unintentional. This  
 61122     volume of POSIX.1-2008 defers to the ISO C standard.

61123     A call to *setjmp()* shall save the calling environment in its *env* argument for later use by  
 61124     *longjmp()*.

61125     It is unspecified whether *setjmp()* is a macro or a function. If a macro definition is suppressed in  
 61126     order to access an actual function, or a program defines an external identifier with the name  
 61127     *setjmp*, the behavior is undefined.

61128     An application shall ensure that an invocation of *setjmp()* appears in one of the following  
 61129     contexts only:

- 61130       • The entire controlling expression of a selection or iteration statement
- 61131       • One operand of a relational or equality operator with the other operand an integral  
 61132       constant expression, with the resulting expression being the entire controlling expression  
 61133       of a selection or iteration statement
- 61134       • The operand of a unary '!' operator with the resulting expression being the entire  
 61135       controlling expression of a selection or iteration
- 61136       • The entire expression of an expression statement (possibly cast to **void**)

61137     If the invocation appears in any other context, the behavior is undefined.

61138 **RETURN VALUE**

61139     If the return is from a direct invocation, *setjmp()* shall return 0. If the return is from a call to  
 61140     *longjmp()*, *setjmp()* shall return a non-zero value.

61141 **ERRORS**

61142     No errors are defined.

61143 **EXAMPLES**

61144     None.

61145 **APPLICATION USAGE**

61146     In general, *sigsetjmp()* is more useful in dealing with errors and interrupts encountered in a low-  
 61147     level subroutine of a program.

61148 **RATIONALE**

61149     None.

61150 **FUTURE DIRECTIONS**

61151     None.

61152 **SEE ALSO**

61153     *longjmp()*, *sigsetjmp()*

61154     XBD <setjmp.h>

**CHANGE HISTORY**

61155  
61156 First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 6**

61157  
61158 The normative text is updated to avoid use of the term “must” for application requirements.



61159 **NAME**61160 setkey — set encoding key (**CRYPT**)61161 **SYNOPSIS**

```
61162 XSI      #include <stdlib.h>
61163         void setkey(const char *key);
```

61164 **DESCRIPTION**

61165 The *setkey()* function provides access to an implementation-defined encoding algorithm. The  
 61166 argument of *setkey()* is an array of length 64 bytes containing only the bytes with numerical  
 61167 value of 0 and 1. If this string is divided into groups of 8, the low-order bit in each group is  
 61168 ignored; this gives a 56-bit key which is used by the algorithm. This is the key that shall be used  
 61169 with the algorithm to encode a string *block* passed to *encrypt()*.

61170 The *setkey()* function shall not change the setting of *errno* if successful. An application wishing to  
 61171 check for error situations should set *errno* to 0 before calling *setkey()*. If *errno* is non-zero on  
 61172 return, an error has occurred.

61173 The *setkey()* function need not be thread-safe.

61174 **RETURN VALUE**

61175 No values are returned.

61176 **ERRORS**

61177 The *setkey()* function shall fail if:

61178 [ENOSYS] The functionality is not supported on this implementation.

61179 **EXAMPLES**

61180 None.

61181 **APPLICATION USAGE**

61182 Decoding need not be implemented in all environments. This is related to government  
 61183 restrictions in some countries on encryption and decryption routines. Historical practice has  
 61184 been to ship a different version of the encryption library without the decryption feature in the  
 61185 routines supplied. Thus the exported version of *encrypt()* does encoding but not decoding.

61186 **RATIONALE**

61187 None.

61188 **FUTURE DIRECTIONS**

61189 A future version of the standard may mark this interface as obsolete or remove it altogether.

61190 **SEE ALSO**

61191 *crypt()*, *encrypt()*

61192 XBD <stdlib.h>

61193 **CHANGE HISTORY**

61194 First released in Issue 1. Derived from Issue 1 of the SVID.

61195 **Issue 5**

61196 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

61197 **Issue 7**

61198 Austin Group Interpretation 1003.1-2001 #156 is applied.

61199 **NAME**

61200       setlocale — set program locale

61201 **SYNOPSIS**

61202       #include &lt;locale.h&gt;

61203       char \*setlocale(int category, const char \*locale);

61204 **DESCRIPTION**

61205 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 61206 conflict between the requirements described here and the ISO C standard is unintentional. This  
 61207 volume of POSIX.1-2008 defers to the ISO C standard.

61208       The *setlocale()* function selects the appropriate piece of the global locale, as specified by the  
 61209 *category* and *locale* arguments, and can be used to change or query the entire global locale or  
 61210 portions thereof. The value LC\_ALL for *category* names the entire global locale; other values for  
 61211 *category* name only a part of the global locale:

61212       LC\_COLLATE       Affects the behavior of regular expressions and the collation functions.

61213       LC\_CTYPE         Affects the behavior of regular expressions, character classification, character  
 61214 conversion functions, and wide-character functions.

61215 CX       LC\_MESSAGES   Affects the affirmative and negative response expressions returned by  
 61216 *nl\_langinfo()* and the way message catalogs are located. It may also affect the  
 61217 behavior of functions that return or write message strings.

61218       LC\_MONETARY   Affects the behavior of functions that handle monetary values.

61219       LC\_NUMERIC     Affects the behavior of functions that handle numeric values.

61220       LC\_TIME         Affects the behavior of the time conversion functions.

61221       The *locale* argument is a pointer to a character string containing the required setting of *category*.  
 61222 The contents of this string are implementation-defined. In addition, the following preset values  
 61223 of *locale* are defined for all settings of *category*:

61224 CX       "POSIX"       Specifies the minimal environment for C-language translation called the  
 61225 POSIX locale. The POSIX locale is the default global locale at entry to *main()*.

61226       "C"             Equivalent to "POSIX".

61227 CX       ""            Specifies an implementation-defined native environment. The determination  
 61228 of the name of the new locale for the specified category depends on the value  
 61229 of the associated environment variables, *LC\_\** and *LANG*; see XBD Chapter 7  
 61230 (on page 135) and Chapter 8 (on page 173).

61231       A null pointer   Directs *setlocale()* to query the current global locale setting and return the  
 61232 name of the locale if *category* is not LC\_ALL, or a string which encodes the  
 61233 locale name(s) for all of the individual categories if *category* is LC\_ALL.

61234 CX       Setting all of the categories of the global locale is similar to successively setting each individual  
 61235 category of the global locale, except that all error checking is done before any actions are  
 61236 performed. To set all the categories of the global locale, *setlocale()* can be invoked as:

61237       setlocale(LC\_ALL, "");

61238       In this case, *setlocale()* shall first verify that the values of all the environment variables it needs  
 61239 according to the precedence rules (described in XBD Chapter 8, on page 173) indicate supported  
 61240 locales. If the value of any of these environment variable searches yields a locale that is not  
 61241 supported (and non-null), *setlocale()* shall return a null pointer and the global locale shall not be

changed. If all environment variables name supported locales, *setlocale()* shall proceed as if it had been called for each category, using the appropriate value from the associated environment variable or from the implementation-defined default if there is no such value.

The global locale established using *setlocale()* shall only be used in threads for which no current locale has been set using *uselocale()* or whose current locale has been set to the global locale using *uselocale(LC\_GLOBAL\_LOCALE)*.

The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls *setlocale()*.

61250 CX The *setlocale()* function need not be thread-safe. +

## 61251 RETURN VALUE

Upon successful completion, *setlocale()* shall return the string associated with the specified category for the new locale. Otherwise, *setlocale()* shall return a null pointer and the global locale shall not be changed.

A null pointer for *locale* shall cause *setlocale()* to return a pointer to the string associated with the specified *category* for the current global locale. The global locale shall not be changed.

The string returned by *setlocale()* is such that a subsequent call with that string and its associated *category* shall restore that part of the global locale. The application shall not modify the string returned. The returned string pointer might be invalidated or the string content might be overwritten by a subsequent call to *setlocale()*. The returned pointer might also be invalidated if the calling thread is terminated. +

## 61262 ERRORS

61263 No errors are defined.

## 61264 EXAMPLES

61265 None.

## 61266 APPLICATION USAGE

The following code illustrates how a program can initialize the international environment for one language, while selectively modifying the global locale such that regular expressions and string operations can be applied to text recorded in a different language:

```
61270 setlocale(LC_ALL, "De");
61271 setlocale(LC_COLLATE, "Fr@dict");
```

Internationalized programs can initiate language operation according to environment variable settings (see XBD Section 8.2, on page 174) by calling *setlocale()* as follows:

```
61274 setlocale(LC_ALL, "");
```

Changing the setting of *LC\_MESSAGES* has no effect on catalogs that have already been opened by calls to *catopen()*.

In order to make use of different locale settings while multiple threads are running, applications should use *uselocale()* in preference to *setlocale()*.

## 61279 RATIONALE

References to the international environment or locale in the following text relate to the global locale for the process. This can be overridden for individual threads using *uselocale()*.

The ISO C standard defines a collection of functions to support internationalization. One of the most significant aspects of these functions is a facility to set and query the *international environment*. The international environment is a repository of information that affects the behavior of certain functionality, namely:

1. Character handling
2. Collating
3. Date/time formatting
4. Numeric editing
5. Monetary formatting
6. Messaging

The `setlocale()` function provides the application developer with the ability to set all or portions, called *categories*, of the international environment. These categories correspond to the areas of functionality mentioned above. The syntax for `setlocale()` is as follows:

```
char *setlocale(int category, const char *locale);
```

where *category* is the name of one of following categories, namely:

```
LC_COLLATE
LC_CTYPE
LC_MESSAGES
LC_MONETARY
LC_NUMERIC
LC_TIME
```

In addition, a special value called `LC_ALL` directs `setlocale()` to set all categories.

There are two primary uses of `setlocale()`:

1. Querying the international environment to find out what it is set to
2. Setting the international environment, or *locale*, to a specific value

The behavior of `setlocale()` in these two areas is described below. Since it is difficult to describe the behavior in words, examples are used to illustrate the behavior of specific uses.

To query the international environment, `setlocale()` is invoked with a specific category and the null pointer as the locale. The null pointer is a special directive to `setlocale()` that tells it to query rather than set the international environment. The following syntax is used to query the name of the international environment:

```
setlocale({LC_ALL, LC_COLLATE, LC_CTYPE, LC_MESSAGES, LC_MONETARY, \
          LC_NUMERIC, LC_TIME}, (char *) NULL);
```

The `setlocale()` function shall return the string corresponding to the current international environment. This value may be used by a subsequent call to `setlocale()` to reset the international environment to this value. However, it should be noted that the return value from `setlocale()` may be a pointer to a static area within the function and is not guaranteed to remain unchanged (that is, it may be modified by a subsequent call to `setlocale()`). Therefore, if the purpose of calling `setlocale()` is to save the value of the current international environment so it can be changed and reset later, the return value should be copied to an array of `char` in the calling program.

There are three ways to set the international environment with `setlocale()`:

`setlocale(category, string)`

This usage sets a specific *category* in the international environment to a specific value corresponding to the value of the *string*. A specific example is provided below:

61327            `setlocale(LC_ALL, "fr_FR.ISO-8859-1");`

61328            In this example, all categories of the international environment are set to the locale  
61329            corresponding to the string "fr\_FR.ISO-8859-1", or to the French language as spoken in  
61330            France using the ISO/IEC 8859-1:1998 standard codeset.

61331            If the string does not correspond to a valid locale, *setlocale()* shall return a null pointer and  
61332            the international environment is not changed. Otherwise, *setlocale()* shall return the name of  
61333            the locale just set.

61334            *setlocale(category, "C")*

61335            The ISO C standard states that one locale must exist on all conforming implementations.  
61336            The name of the locale is C and corresponds to a minimal international environment needed  
61337            to support the C programming language.

61338            *setlocale(category, "")*

61339            This sets a specific category to an implementation-defined default. This corresponds to the  
61340            value of the environment variables.

## 61341 FUTURE DIRECTIONS

61342            None.

## 61343 SEE ALSO

61344            *catopen()*, *exec*, *fprintf()*, *fscanf()*, *isalnum()*, *isalpha()*, *isblank()*, *iscntrl()*, *isdigit()*, *isgraph()*,  
61345            *islower()*, *isprint()*, *ispunct()*, *isspace()*, *isupper()*, *iswalnum()*, *iswalpha()*, *iswblank()*, *iswcntrl()*,  
61346            *iswctype()*, *iswdigit()*, *iswgraph()*, *iswlower()*, *iswprint()*, *iswpunct()*, *iswspace()*, *iswupper()*,  
61347            *iswxdigit()*, *isxdigit()*, *localeconv()*, *mblen()*, *mbstowcs()*, *mbtowc()*, *newlocale()*, *nl\_langinfo()*,  
61348            *perror()*, *psiginfo()*, *strcoll()*, *strerror()*, *strfmon()*, *strsignal()*, *strtod()*, *strxfrm()*, *tolower()*,  
61349            *toupper()*, *towlower()*, *towupper()*, *uselocale()*, *wscoll()*, *wctod()*, *wctombs()*, *wcsxfrm()*, *wctomb()*

61350            XBD Chapter 7 (on page 135), Chapter 8 (on page 173), [<langinfo.h>](#), [<locale.h>](#)

## 61351 CHANGE HISTORY

61352            First released in Issue 3.

### 61353 Issue 5

61354            The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

### 61355 Issue 6

61356            Extensions beyond the ISO C standard are marked.

61357            The normative text is updated to avoid use of the term “must” for application requirements.

61358            IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/124 is applied, updating the  
61359            DESCRIPTION to clarify the behavior of:

61360            `setlocale(LC_ALL, "");`

### 61361 Issue 7

61362            Functionality relating to the Threads option is moved to the Base.

61363            POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0553 [302], XSH/TC1-2008/0554 [303],  
61364            XSH/TC1-2008/0555 [302], XSH/TC1-2008/0556 [302], XSH/TC1-2008/0557 [302],  
61365            XSH/TC1-2008/0558 [302], XSH/TC1-2008/0559 [302], XSH/TC1-2008/0560 [288],  
61366            XSH/TC1-2008/0561 [302], XSH/TC1-2008/0562 [302], XSH/TC1-2008/0563 [302],  
61367            XSH/TC1-2008/0564 [302], XSH/TC1-2008/0565 [302], XSH/TC1-2008/0566 [302],  
61368            XSH/TC1-2008/0567 [288], XSH/TC1-2008/0568 [288], and XSH/TC1-2008/0569 [303] are  
61369            applied.

61370 **NAME**

61371 setlogmask — set the log priority mask

61372 **SYNOPSIS**

```
61373 XSI    #include <syslog.h>
61374      int setlogmask(int maskpri);
```

61375 **DESCRIPTION**61376 Refer to *closelog()*.

61377 **NAME**

61378       setnetent — network database function

61379 **SYNOPSIS**

61380       #include &lt;netdb.h&gt;

61381       void setnetent(int *stayopen*);61382 **DESCRIPTION**61383       Refer to *endnetent()*.

61384 **NAME**

61385       setpgid — set process group ID for job control

61386 **SYNOPSIS**

61387       #include &lt;unistd.h&gt;

61388       int setpgid(pid\_t pid, pid\_t pgid);

61389 **DESCRIPTION**61390       The *setpgid()* function shall either join an existing process group or create a new process group within the session of the calling process.

61392       The process group ID of a session leader shall not change.

61393       Upon successful completion, the process group ID of the process with a process ID that matches *pid* shall be set to *pgid*.61395       As a special case, if *pid* is 0, the process ID of the calling process shall be used. Also, if *pgid* is 0, the process ID of the indicated process shall be used.61397 **RETURN VALUE**61398       Upon successful completion, *setpgid()* shall return 0; otherwise, -1 shall be returned and *errno* shall be set to indicate the error.61400 **ERRORS**61401       The *setpgid()* function shall fail if:61402       [EACCES]       The value of the *pid* argument matches the process ID of a child process of the calling process and the child process has successfully executed one of the *exec* functions.61405       [EINVAL]       The value of the *pgid* argument is less than 0, or is not a value supported by the implementation.61407       [EPERM]       The process indicated by the *pid* argument is a session leader.61408       [EPERM]       The value of the *pid* argument matches the process ID of a child process of the calling process and the child process is not in the same session as the calling process.61411       [EPERM]       The value of the *pgid* argument is valid but does not match the process ID of the process indicated by the *pid* argument and there is no process with a process group ID that matches the value of the *pgid* argument in the same session as the calling process.61415       [ESRCH]       The value of the *pid* argument does not match the process ID of the calling process or of a child process of the calling process.61417 **EXAMPLES**

61418       None.

61419 **APPLICATION USAGE**

61420       None.

61421 **RATIONALE**61422       The *setpgid()* function shall group processes together for the purpose of signaling, placement in foreground or background, and other job control actions.61424       The *setpgid()* function is similar to the *setpgrp()* function of 4.2 BSD, except that 4.2 BSD allowed the specified new process group to assume any value. This presents certain security problems and is more flexible than necessary to support job control.



To provide tighter security, *setpgid()* only allows the calling process to join a process group already in use inside its session or create a new process group whose process group ID was equal to its process ID.

When a job control shell spawns a new job, the processes in the job must be placed into a new process group via *setpgid()*. There are two timing constraints involved in this action:

1. The new process must be placed in the new process group before the appropriate program is launched via one of the *exec* functions.
2. The new process must be placed in the new process group before the shell can correctly send signals to the new process group.

To address these constraints, the following actions are performed. The new processes call *setpgid()* to alter their own process groups after *fork()* but before *exec*. This satisfies the first constraint. Under 4.3 BSD, the second constraint is satisfied by the synchronization property of *vfork()*; that is, the shell is suspended until the child has completed the *exec*, thus ensuring that the child has completed the *setpgid()*. A new version of *fork()* with this same synchronization property was considered, but it was decided instead to merely allow the parent shell process to adjust the process group of its child processes via *setpgid()*. Both timing constraints are now satisfied by having both the parent shell and the child attempt to adjust the process group of the child process; it does not matter which succeeds first.

Since it would be confusing to an application to have its process group change after it began executing (that is, after *exec*), and because the child process would already have adjusted its process group before this, the [EACCES] error was added to disallow this.

One non-obvious use of *setpgid()* is to allow a job control shell to return itself to its original process group (the one in effect when the job control shell was executed). A job control shell does this before returning control back to its parent when it is terminating or suspending itself as a way of restoring its job control “state” back to what its parent would expect. (Note that the original process group of the job control shell typically matches the process group of its parent, but this is not necessarily always the case.)

## FUTURE DIRECTIONS

None.

## SEE ALSO

*exec*, *getpgrp()*, *setsid()*, *tcsetpgrp()*

XBD [`<sys/types.h>`](#), [`<unistd.h>`](#)

## CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

### Issue 6

In the SYNOPSIS, the optional include of the [`<sys/types.h>`](#) header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include [`<sys/types.h>`](#) has been removed. Although [`<sys/types.h>`](#) was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- The *setpgid()* function is mandatory since `_POSIX_JOB_CONTROL` is required to be defined in this version. This is a FIPS requirement.

61470  
61471  
61472  
61473

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/56 is applied, changing the wording in the DESCRIPTION from “the process group ID of the indicated process shall be used” to “the process ID of the indicated process shall be used”. This change reverts the wording to as in the ISO POSIX-1: 1996 standard; it appeared to be an unintentional change.

61474 **NAME**

61475       setpgrp — set the process group ID

61476 **SYNOPSIS**

```
61477 OB XSI  #include <unistd.h>
61478         pid_t setpgrp(void);
```

61479 **DESCRIPTION**

61480       If the calling process is not already a session leader, *setpgrp()* sets the process group ID of the  
 61481       calling process to the process ID of the calling process. If *setpgrp()* creates a new session, then the  
 61482       new session has no controlling terminal.

61483       The *setpgrp()* function has no effect when the calling process is a session leader.

61484 **RETURN VALUE**

61485       Upon completion, *setpgrp()* shall return the process group ID.

61486 **ERRORS**

61487       No errors are defined.

61488 **EXAMPLES**

61489       None.

61490 **APPLICATION USAGE**

61491       It is unspecified whether this function behaves as *setpgid(0,0)* or *setsid()* unless the process is  
 61492       already a session leader. Therefore, applications are encouraged to use *setpgid()* or *setsid()* as  
 61493       appropriate.

61494 **RATIONALE**

61495       None.

61496 **FUTURE DIRECTIONS**

61497       The *setpgrp()* function may be removed in a future version.

61498 **SEE ALSO**

61499       *exec*, *fork()*, *getpid()*, *getsid()*, *kill()*, *setpgid()*, *setsid()*

61500       XBD [<unistd.h>](#)

61501 **CHANGE HISTORY**

61502       First released in Issue 4, Version 2.

61503 **Issue 5**

61504       Moved from X/OPEN UNIX extension to BASE.

61505 **Issue 7**

61506       The *setpgrp()* function is marked obsolescent.

61507 **NAME**

61508       setpriority — set the nice value

61509 **SYNOPSIS**

```
61510 XSI      #include <sys/resource.h>
61511          int setpriority(int which, id_t who, int nice);
```

61512 **DESCRIPTION**61513       Refer to *getpriority()*.

61514 **NAME**

61515       setprotoent — network protocol database functions

61516 **SYNOPSIS**

61517       #include &lt;netdb.h&gt;

61518       void setprotoent(int *stayopen*);61519 **DESCRIPTION**61520       Refer to *endprotoent()*.

61521 **NAME**

61522        setpwent — user database function

61523 **SYNOPSIS**61524 XSI        `#include <pwd.h>`61525        `void setpwent(void);`61526 **DESCRIPTION**61527        Refer to *endpwent()*.

**NAME**

setregid — set real and effective group IDs

**SYNOPSIS**

```
#include <unistd.h>

int setregid(gid_t rgid, gid_t egid);
```

**DESCRIPTION**

The *setregid()* function shall set the real and effective group IDs of the calling process.

If *rgid* is  $-1$ , the real group ID shall not be changed; if *egid* is  $-1$ , the effective group ID shall not be changed.

The real and effective group IDs may be set to different values in the same call.

Only a process with appropriate privileges can set the real group ID and the effective group ID to any valid value.

A non-privileged process can set either the real group ID to the saved set-group-ID from one of the *exec* family of functions, or the effective group ID to the saved set-group-ID or the real group ID.

If the real group ID is being set (*rgid* is not  $-1$ ), or the effective group ID is being set to a value not equal to the real group ID, then the saved set-group-ID of the current process shall be set equal to the new effective group ID.

Any supplementary group IDs of the calling process remain unchanged.

**RETURN VALUE**

Upon successful completion, 0 shall be returned. Otherwise,  $-1$  shall be returned and *errno* set to indicate the error, and neither of the group IDs are changed.

**ERRORS**

The *setregid()* function shall fail if:

- |          |                                                                                                                                                                                                                                  |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [EINVAL] | The value of the <i>rgid</i> or <i>egid</i> argument is invalid or out-of-range.                                                                                                                                                 |
| [EPERM]  | The process does not have appropriate privileges and a change other than changing the real group ID to the saved set-group-ID, or changing the effective group ID to the real group ID or the saved set-group-ID, was requested. |

**EXAMPLES**

None.

**APPLICATION USAGE**

If a non-privileged set-group-ID process sets its effective group ID to its real group ID, it can only set its effective group ID back to the previous value if *rgid* was  $-1$  in the *setregid()* call, since the saved-group-ID is not changed in that case. If *rgid* was equal to the real group ID in the *setregid()* call, then the saved set-group-ID will also have been changed to the real user ID.

**RATIONALE**

Earlier versions of this standard did not specify whether the saved set-group-ID was affected by *setregid()* calls. This version specifies common existing practice that constitutes an important security feature. The ability to set both the effective group ID and saved set-group-ID to be the same as the real group ID means that any security weakness in code that is executed after that point cannot result in malicious code being executed with the previous effective group ID. Privileged applications could already do this using just *setgid()*, but for non-privileged

61571 applications the only standard method available is to use this feature of *setregid()*.

61572 **FUTURE DIRECTIONS**

61573 None.

61574 **SEE ALSO**

61575 *exec*, *getegid()*, *geteuid()*, *getgid()*, *getuid()*, *setegid()*, *seteuid()*, *setgid()*, *setreuid()*, *setuid()*

61576 XBD <unistd.h>

61577 **CHANGE HISTORY**

61578 First released in Issue 4, Version 2.

61579 **Issue 5**

61580 Moved from X/OPEN UNIX extension to BASE.

61581 The DESCRIPTION is updated to indicate that the saved set-group-ID can be set by any of the  
61582 *exec* family of functions, not just *execve()*.

61583 **Issue 7**

61584 SD5-XSH-ERN-177 is applied, adding the ability to set both the effective group ID and saved set-  
61585 group-ID to be the same as the real group ID.



**NAME**

setreuid — set real and effective user IDs

**SYNOPSIS**

```
XSI    #include <unistd.h>
      int setreuid(uid_t ruid, uid_t euid);
```

**DESCRIPTION**

The *setreuid()* function shall set the real and effective user IDs of the current process to the values specified by the *ruid* and *euid* arguments. If *ruid* or *euid* is *-1*, the corresponding effective or real user ID of the current process shall be left unchanged.

A process with appropriate privileges can set either ID to any value. An unprivileged process can only set the effective user ID if the *euid* argument is equal to either the real, effective, or saved user ID of the process.

If the real user ID is being set (*ruid* is not *-1*), or the effective user ID is being set to a value not equal to the real user ID, then the saved set-user-ID of the current process shall be set equal to the new effective user ID.

It is unspecified whether a process without appropriate privileges is permitted to change the real user ID to match the current effective user ID or saved set-user-ID of the process.

**RETURN VALUE**

Upon successful completion, 0 shall be returned. Otherwise, *-1* shall be returned and *errno* set to indicate the error.

**ERRORS**

The *setreuid()* function shall fail if:

- |          |                                                                                                                                                                                                                                                                                            |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| [EINVAL] | The value of the <i>ruid</i> or <i>euid</i> argument is invalid or out-of-range.                                                                                                                                                                                                           |
| [EPERM]  | The current process does not have appropriate privileges, and either an attempt was made to change the effective user ID to a value other than the real user ID or the saved set-user-ID or an attempt was made to change the real user ID to a value not permitted by the implementation. |

**EXAMPLES****Setting the Effective User ID to the Real User ID**

The following example sets the effective user ID of the calling process to the real user ID, so that files created later will be owned by the current user. It also sets the saved set-user-ID to the real user ID, so any future attempt to set the effective user ID back to its previous value will fail.

```
#include <unistd.h>
#include <sys/types.h>
...
setreuid(getuid(), getuid());
...
```

**APPLICATION USAGE**

None.

**RATIONALE**

Earlier versions of this standard did not specify whether the saved set-user-ID was affected by *setreuid()* calls. This version specifies common existing practice that constitutes an important security feature. The ability to set both the effective user ID and saved set-user-ID to be the same as the real user ID means that any security weakness in code that is executed after that point cannot result in malicious code being executed with the previous effective user ID. Privileged applications could already do this using just *setuid()*, but for non-privileged applications the only standard method available is to use this feature of *setreuid()*.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*getegid()*, *geteuid()*, *getgid()*, *getuid()*, *setegid()*, *seteuid()*, *setgid()*, *setregid()*, *setuid()*

XBD <**unistd.h**>

**CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

**Issue 7**

SD5-XSH-ERN-177 is applied, adding the ability to set both the effective user ID and the saved set-user-ID to be the same as the real user ID.

61645 **NAME**

61646 setrlimit — control maximum resource consumption

61647 **SYNOPSIS**61648 XSI `#include <sys/resource.h>`61649 `int setrlimit(int resource, const struct rlimit *rlp);`61650 **DESCRIPTION**61651 Refer to *getrlimit()*.

61652 **NAME**

61653       setservent — network services database functions

61654 **SYNOPSIS**

61655       #include &lt;netdb.h&gt;

61656       void setservent(int *stayopen*);61657 **DESCRIPTION**61658       Refer to *endservent()*.

61659 **NAME**

61660 setsid — create session and set process group ID

61661 **SYNOPSIS**

61662 #include &lt;unistd.h&gt;

61663 pid\_t setsid(void);

61664 **DESCRIPTION**

61665 The *setsid()* function shall create a new session, if the calling process is not a process group  
 61666 leader. Upon return the calling process shall be the session leader of this new session, shall be  
 61667 the process group leader of a new process group, and shall have no controlling terminal. The  
 61668 process group ID of the calling process shall be set equal to the process ID of the calling process.  
 61669 The calling process shall be the only process in the new process group and the only process in  
 61670 the new session.

61671 **RETURN VALUE**

61672 Upon successful completion, *setsid()* shall return the value of the new process group ID of the  
 61673 calling process. Otherwise, it shall return -1 and set *errno* to indicate the error.

61674 **ERRORS**61675 The *setsid()* function shall fail if:

61676 [EPERM] The calling process is already a process group leader, or the process group ID  
 61677 of a process other than the calling process matches the process ID of the  
 61678 calling process.

61679 **EXAMPLES**

61680 None.

61681 **APPLICATION USAGE**

61682 None.

61683 **RATIONALE**

61684 The *setsid()* function is similar to the *setpgid()* function of System V. System V, without job  
 61685 control, groups processes into process groups and creates new process groups via *setpgid()*; only  
 61686 one process group may be part of a login session.

61687 Job control allows multiple process groups within a login session. In order to limit job control  
 61688 actions so that they can only affect processes in the same login session, this volume of  
 61689 POSIX.1-2008 adds the concept of a session that is created via *setsid()*. The *setsid()* function also  
 61690 creates the initial process group contained in the session. Additional process groups can be  
 61691 created via the *setpgid()* function. A System V process group would correspond to a POSIX  
 61692 System Interfaces session containing a single POSIX process group. Note that this function  
 61693 requires that the calling process not be a process group leader. The usual way to ensure this is  
 61694 true is to create a new process with *fork()* and have it call *setsid()*. The *fork()* function  
 61695 guarantees that the process ID of the new process does not match any existing process group ID.

61696 **FUTURE DIRECTIONS**

61697 None.

61698 **SEE ALSO**61699 *getsid()*, *setpgid()*, *setpgid()*

61700 XBD &lt;sys/types.h&gt;, &lt;unistd.h&gt;

61701 **CHANGE HISTORY**

61702 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

61703 **Issue 6**

61704 In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.

61705 The following new requirements on POSIX implementations derive from alignment with the  
61706 Single UNIX Specification:

- 61707 • The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was  
61708 required for conforming implementations of previous POSIX specifications, it was not  
61709 required for UNIX applications.

61710 **Issue 7**

61711 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0570 [421] is applied.

61712 **NAME**

61713       setsockopt — set the socket options

61714 **SYNOPSIS**

```
61715       #include <sys/socket.h>
61716       int setsockopt(int socket, int level, int option_name,
61717                      const void *option_value, socklen_t option_len);
```

61718 **DESCRIPTION**

61719       The *setsockopt()* function shall set the option specified by the *option\_name* argument, at the  
 61720       protocol level specified by the *level* argument, to the value pointed to by the *option\_value*  
 61721       argument for the socket associated with the file descriptor specified by the *socket* argument.

61722       The *level* argument specifies the protocol level at which the option resides. To set options at the  
 61723       socket level, specify the *level* argument as SOL\_SOCKET. To set options at other levels, supply  
 61724       the appropriate *level* identifier for the protocol controlling the option. For example, to indicate  
 61725       that an option is interpreted by the TCP (Transport Control Protocol), set *level* to IPPROTO\_TCP  
 61726       as defined in the **<netinet/in.h>** header.

61727       The *option\_name* argument specifies a single option to set. It can be one of the socket-level  
 61728       options defined in **<sys/socket.h>** and described in [Section 2.10.16](#) (on page 528). If *option\_name*  
 61729       is equal to SO\_RCVTIMEO or SO\_SNDTIMEO and the implementation supports setting the  
 61730       option, it is unspecified whether the **struct timeval** pointed to by *option\_value* is stored as  
 61731       provided by this function or is rounded up to align with the resolution of the clock being used. If  
 61732       *setsockopt()* is called with *option\_name* equal to SO\_ACCEPTCONN, SO\_ERROR, or SO\_TYPE,  
 61733       the behavior is unspecified.

61734 **RETURN VALUE**

61735       Upon successful completion, *setsockopt()* shall return 0. Otherwise, -1 shall be returned and  
 61736       *errno* set to indicate the error.

61737 **ERRORS**

61738       The *setsockopt()* function shall fail if:

- |       |               |                                                                                                         |
|-------|---------------|---------------------------------------------------------------------------------------------------------|
| 61739 | [EBADF]       | The <i>socket</i> argument is not a valid file descriptor.                                              |
| 61740 | [EDOM]        | The send and receive timeout values are too big to fit into the timeout fields in the socket structure. |
| 61742 | [EINVAL]      | The specified option is invalid at the specified socket level or the socket has been shut down.         |
| 61744 | [EISCONN]     | The socket is already connected, and a specified option cannot be set while the socket is connected.    |
| 61746 | [ENOPROTOOPT] | The option is not supported by the protocol.                                                            |
| 61748 | [ENOTSOCK]    | The <i>socket</i> argument does not refer to a socket.                                                  |

61749       The *setsockopt()* function may fail if:

- |       |           |                                                                          |
|-------|-----------|--------------------------------------------------------------------------|
| 61750 | [ENOMEM]  | There was insufficient memory available for the operation to complete.   |
| 61751 | [ENOBUFS] | Insufficient resources are available in the system to complete the call. |

**EXAMPLES**

None.

**APPLICATION USAGE**

The *setsockopt()* function provides an application program with the means to control socket behavior. An application program can use *setsockopt()* to allocate buffer space, control timeouts, or permit socket data broadcasts. The `<sys/socket.h>` header defines the socket-level options available to *setsockopt()*.

Options may exist at multiple protocol levels. The `SO_` options are always present at the uppermost socket level.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

Section 2.10 (on page 523), *bind()*, *endprotoent()*, *getsockopt()*, *socket()*

XBD `<netinet/in.h>`, `<sys/socket.h>`

**CHANGE HISTORY**

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/125 is applied, updating the `SO_LINGER` option in the DESCRIPTION to refer to the calling thread rather than the process.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #158 is applied, removing text relating to socket options that is now in Section 2.10.16 (on page 528).

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0571 [369] is applied.



61776 **NAME**

61777        setstate — switch pseudo-random number generator state arrays

61778 **SYNOPSIS**

```
61779 XSI      #include <stdlib.h>
61780          char *setstate(char *state);
```

61781 **DESCRIPTION**61782        Refer to *initstate()*.

61783 **NAME**

61784       setuid — set user ID

61785 **SYNOPSIS**

```
61786       #include <unistd.h>
61787       int setuid(uid_t uid);
```

61788 **DESCRIPTION**

61789       If the process has appropriate privileges, *setuid()* shall set the real user ID, effective user ID, and  
61790       the saved set-user-ID of the calling process to *uid*.

61791       If the process does not have appropriate privileges, but *uid* is equal to the real user ID or the  
61792       saved set-user-ID, *setuid()* shall set the effective user ID to *uid*; the real user ID and saved set-  
61793       user-ID shall remain unchanged.

61794       The *setuid()* function shall not affect the supplementary group list in any way.

61795 **RETURN VALUE**

61796       Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to  
61797       indicate the error.

61798 **ERRORS**

61799       The *setuid()* function shall fail, return -1, and set *errno* to the corresponding value if one or more  
61800       of the following are true:

61801       [EINVAL]       The value of the *uid* argument is invalid and not supported by the  
61802       implementation.

61803       [EPERM]        The process does not have appropriate privileges and *uid* does not match the  
61804       real user ID or the saved set-user-ID.

61805 **EXAMPLES**

61806       None.

61807 **APPLICATION USAGE**

61808       None.

61809 **RATIONALE**

61810       The various behaviors of the *setuid()* and *setgid()* functions when called by non-privileged  
61811       processes reflect the behavior of different historical implementations. For portability, it is  
61812       recommended that new non-privileged applications use the *seteuid()* and *setegid()* functions  
61813       instead.

61814       The saved set-user-ID capability allows a program to regain the effective user ID established at  
61815       the last *exec* call. Similarly, the saved set-group-ID capability allows a program to regain the  
61816       effective group ID established at the last *exec* call. These capabilities are derived from System V.  
61817       Without them, a program might have to run as superuser in order to perform the same  
61818       functions, because superuser can write on the user's files. This is a problem because such a  
61819       program can write on any user's files, and so must be carefully written to emulate the  
61820       permissions of the calling process properly. In System V, these capabilities have traditionally  
61821       been implemented only via the *setuid()* and *setgid()* functions for non-privileged processes. The  
61822       fact that the behavior of those functions was different for privileged processes made them  
61823       difficult to use. The POSIX.1-1990 standard defined the *setuid()* function to behave differently  
61824       for privileged and unprivileged users. When the caller had appropriate privileges, the function  
61825       set the real user ID, effective user ID, and saved set-user ID of the calling process on  
61826       implementations that supported it. When the caller did not have appropriate privileges, the  
61827       function set only the effective user ID, subject to permission checks. The former use is generally  
61828       needed for utilities like *login* and *su*, which are not conforming applications and thus outside the

scope of POSIX.1-2008. These utilities wish to change the user ID irrevocably to a new value, generally that of an unprivileged user. The latter use is needed for conforming applications that are installed with the set-user-ID bit and need to perform operations using the real user ID.

POSIX.1-2008 augments the latter functionality with a mandatory feature named `_POSIX_SAVED_IDS`. This feature permits a set-user-ID application to switch its effective user ID back and forth between the values of its *exec*-time real user ID and effective user ID. Unfortunately, the POSIX.1-1990 standard did not permit a conforming application using this feature to work properly when it happened to be executed with (implementation-defined) appropriate privileges. Furthermore, the application did not even have a means to tell whether it had this privilege. Since the saved set-user-ID feature is quite desirable for applications, as evidenced by the fact that NIST required it in FIPS 151-2, it has been mandated by POSIX.1-2008. However, there are implementors who have been reluctant to support it given the limitation described above.

The 4.3BSD system handles the problem by supporting separate functions: *setuid()* (which always sets both the real and effective user IDs, like *setuid()* in POSIX.1-2008 for privileged users), and *seteuid()* (which always sets just the effective user ID, like *setuid()* in POSIX.1-2008 for non-privileged users). This separation of functionality into distinct functions seems desirable. 4.3BSD does not support the saved set-user-ID feature. It supports similar functionality of switching the effective user ID back and forth via *setreuid()*, which permits reversing the real and effective user IDs. This model seems less desirable than the saved set-user-ID because the real user ID changes as a side-effect. The current 4.4BSD includes saved effective IDs and uses them for *seteuid()* and *setegid()* as described above. The *setreuid()* and *setregid()* functions will be deprecated or removed.

The solution here is:

- Require that all implementations support the functionality of the saved set-user-ID, which is set by the *exec* functions and by privileged calls to *setuid()*.
- Add the *seteuid()* and *setegid()* functions as portable alternatives to *setuid()* and *setgid()* for non-privileged and privileged processes.

Historical systems have provided two mechanisms for a set-user-ID process to change its effective user ID to be the same as its real user ID in such a way that it could return to the original effective user ID: the use of the *setuid()* function in the presence of a saved set-user-ID, or the use of the BSD *setreuid()* function, which was able to swap the real and effective user IDs. The changes included in POSIX.1-2008 provide a new mechanism using *seteuid()* in conjunction with a saved set-user-ID. Thus, all implementations with the new *seteuid()* mechanism will have a saved set-user-ID for each process, and most of the behavior controlled by `_POSIX_SAVED_IDS` has been changed to agree with the case where the option was defined. The *kill()* function is an exception. Implementors of the new *seteuid()* mechanism will generally be required to maintain compatibility with the older mechanisms previously supported by their systems. However, compatibility with this use of *setreuid()* and with the `_POSIX_SAVED_IDS` behavior of *kill()* is unfortunately complicated. If an implementation with a saved set-user-ID allows a process to use *setreuid()* to swap its real and effective user IDs, but were to leave the saved set-user-ID unmodified, the process would then have an effective user ID equal to the original real user ID, and both real and saved set-user-ID would be equal to the original effective user ID. In that state, the real user would be unable to kill the process, even though the effective user ID of the process matches that of the real user, if the *kill()* behavior of `_POSIX_SAVED_IDS` was used. This is obviously not acceptable. The alternative choice, which is used in at least one implementation, is to change the saved set-user-ID to the effective user ID during most calls to *setreuid()*. The standard developers considered that alternative to be less correct than the retention of the old behavior of *kill()* in such systems. Current conforming applications shall

61878 accommodate either behavior from *kill()*, and there appears to be no strong reason for *kill()* to  
61879 check the saved set-user-ID rather than the effective user ID.

#### 61880 FUTURE DIRECTIONS

61881 None.

#### 61882 SEE ALSO

61883 *exec*, *getegid()*, *geteuid()*, *getgid()*, *getuid()*, *setegid()*, *seteuid()*, *setgid()*, *setregid()*, *setreuid()*

61884 XBD *<sys/types.h>*, *<unistd.h>*

#### 61885 CHANGE HISTORY

61886 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 61887 Issue 6

61888 In the SYNOPSIS, the optional include of the *<sys/types.h>* header is removed.

61889 The following new requirements on POSIX implementations derive from alignment with the  
61890 Single UNIX Specification:

- 61891 • The requirement to include *<sys/types.h>* has been removed. Although *<sys/types.h>* was  
61892 required for conforming implementations of previous POSIX specifications, it was not  
61893 required for UNIX applications.
- 61894 • The functionality associated with *\_POSIX\_SAVED\_IDS* is now mandatory. This is a FIPS  
61895 requirement.

61896 The following changes were made to align with the IEEE P1003.1a draft standard:

- 61897 • The effects of *setuid()* in processes without appropriate privileges are changed.
- 61898 • A requirement that the supplementary group list is not affected is added.

61899 **NAME**

61900       setutxent — reset the user accounting database to the first entry

61901 **SYNOPSIS**

```
61902 XSI      #include <utmpx.h>  
61903          void setutxent(void);
```

61904 **DESCRIPTION**61905       Refer to *endutxent()*.

**NAME**

setvbuf — assign buffering to a stream

**SYNOPSIS**

```
#include <stdio.h>
```

```
int setvbuf(FILE *restrict stream, char *restrict buf, int type,  
            size_t size);
```

**DESCRIPTION**

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *setvbuf()* function may be used after the stream pointed to by *stream* is associated with an open file but before any other operation (other than an unsuccessful call to *setvbuf()*) is performed on the stream. The argument *type* determines how *stream* shall be buffered, as follows:

- {\_IOFBF} shall cause input/output to be fully buffered.
- {\_IOLBF} shall cause input/output to be line buffered.
- {\_IONBF} shall cause input/output to be unbuffered.

If *buf* is not a null pointer, the array it points to may be used instead of a buffer allocated by *setvbuf()* and the argument *size* specifies the size of the array; otherwise, *size* may determine the size of a buffer allocated by the *setvbuf()* function. The contents of the array at any time are unspecified.

For information about streams, see [Section 2.5](#) (on page 495).

**RETURN VALUE**

CX Upon successful completion, *setvbuf()* shall return 0. Otherwise, it shall return a non-zero value if an invalid value is given for *type* or if the request cannot be honored, and may set *errno* to indicate the error.

**ERRORS**

The *setvbuf()* function may fail if:

CX [EBADF] The file descriptor underlying *stream* is not valid.

**EXAMPLES**

None.

**APPLICATION USAGE**

A common source of error is allocating buffer space as an “automatic” variable in a code block, and then failing to close the stream in the same block.

With *setvbuf()*, allocating a buffer of *size* bytes does not necessarily imply that all of *size* bytes are used for the buffer area.

Applications should note that many implementations only provide line buffering on input from terminal devices.

**RATIONALE**

None.

61946 **FUTURE DIRECTIONS**

61947 None.

61948 **SEE ALSO**61949 [Section 2.5](#) (on page 495), [fopen\(\)](#), [setbuf\(\)](#)61950 XBD [<stdio.h>](#)61951 **CHANGE HISTORY**

61952 First released in Issue 1. Derived from Issue 1 of the SVID.

61953 **Issue 6**

61954 Extensions beyond the ISO C standard are marked.

61955 The *setvbuf()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

**NAME**shm\_open — open a shared memory object (**REALTIME**)**SYNOPSIS**

```
SHM    #include <sys/mman.h>
int shm_open(const char *name, int oflag, mode_t mode);
```

**DESCRIPTION**

The *shm\_open()* function shall establish a connection between a shared memory object and a file descriptor. It shall create an open file description that refers to the shared memory object and a file descriptor that refers to that open file description. The file descriptor shall be allocated as described in [Section 2.14](#) (on page 549), and can be used by other functions to refer to that shared memory object. The *name* argument points to a string naming a shared memory object. It is unspecified whether the name appears in the file system and is visible to other functions that take pathnames as arguments. The *name* argument conforms to the construction rules for a pathname, except that the interpretation of <slash> characters other than the leading <slash> character in *name* is implementation-defined, and that the length limits for the *name* argument are implementation-defined and need not be the same as the pathname limits {PATH\_MAX} and {NAME\_MAX}. If *name* begins with the <slash> character, then processes calling *shm\_open()* with the same value of *name* refer to the same shared memory object, as long as that name has not been removed. If *name* does not begin with the <slash> character, the effect is implementation-defined.

If successful, *shm\_open()* shall return a file descriptor for the shared memory object. The open file description is new, and therefore the file descriptor does not share it with any other processes. It is unspecified whether the file offset is set. The FD\_CLOEXEC file descriptor flag associated with the new file descriptor is set.

The file status flags and file access modes of the open file description are according to the value of *oflag*. The *oflag* argument is the bitwise-inclusive OR of the following flags defined in the <fcntl.h> header. Applications specify exactly one of the first two values (access modes) below in the value of *oflag*:

O\_RDONLY      Open for read access only.

O\_RDWR        Open for read or write access.

Any combination of the remaining flags may be specified in the value of *oflag*:

O\_CREAT        If the shared memory object exists, this flag has no effect, except as noted under O\_EXCL below. Otherwise, the shared memory object is created. The user ID of the shared memory object shall be set to the effective user ID of the process. The group ID of the shared memory object shall be set to the effective group ID of the process; however, if the *name* argument is visible in the file system, the group ID may be set to the group ID of the containing directory. The permission bits of the shared memory object shall be set to the value of the *mode* argument except those set in the file mode creation mask of the process. When bits in *mode* other than the file permission bits are set, the effect is unspecified. The *mode* argument does not affect whether the shared memory object is opened for reading, for writing, or for both. The shared memory object has a size of zero.

O\_EXCL        If O\_EXCL and O\_CREAT are set, *shm\_open()* fails if the shared memory object exists. The check for the existence of the shared memory object and the creation of the object if it does not exist is atomic with respect to other



processes executing *shm\_open()* naming the same shared memory object with *O\_EXCL* and *O\_CREAT* set. If *O\_EXCL* is set and *O\_CREAT* is not set, the result is undefined.

*O\_TRUNC* If the shared memory object exists, and it is successfully opened *O\_RDWR*, the object shall be truncated to zero length and the mode and owner shall be unchanged by this function call. The result of using *O\_TRUNC* with *O\_RDONLY* is undefined.

When a shared memory object is created, the state of the shared memory object, including all data associated with the shared memory object, persists until the shared memory object is unlinked and all other references are gone. It is unspecified whether the name and shared memory object state remain valid after a system reboot.

#### 62013 RETURN VALUE

Upon successful completion, the *shm\_open()* function shall return a non-negative integer representing the file descriptor. Otherwise, it shall return *-1* and set *errno* to indicate the error.

#### 62016 ERRORS

The *shm\_open()* function shall fail if:

[EACCES] The shared memory object exists and the permissions specified by *oflag* are denied, or the shared memory object does not exist and permission to create the shared memory object is denied, or *O\_TRUNC* is specified and write permission is denied.

[EEXIST] *O\_CREAT* and *O\_EXCL* are set and the named shared memory object already exists.

[EINTR] The *shm\_open()* operation was interrupted by a signal.

[EINVAL] The *shm\_open()* operation is not supported for the given name.

[EMFILE] All file descriptors available to the process are currently open.

[ENFILE] Too many shared memory objects are currently open in the system.

[ENOENT] *O\_CREAT* is not set and the named shared memory object does not exist.

[ENOSPC] There is insufficient space for the creation of the new shared memory object.

The *shm\_open()* function may fail if:

[ENAMETOOLONG]

The length of the *name* argument exceeds *{\_POSIX\_PATH\_MAX}* on systems that do not support the XSI option or exceeds *{\_XOPEN\_PATH\_MAX}* on XSI systems, or has a pathname component that is longer than *{\_POSIX\_NAME\_MAX}* on systems that do not support the XSI option or longer than *{\_XOPEN\_NAME\_MAX}* on XSI systems.

62037 **EXAMPLES**62038 **Creating and Mapping a Shared Memory Object**

62039 The following code segment demonstrates the use of *shm\_open()* to create a shared memory  
 62040 object which is then sized using *ftruncate()* before being mapped into the process address space  
 62041 using *mmap()*:

```
62042 #include <unistd.h>
62043 #include <sys/mman.h>
62044 ...
62045 #define MAX_LEN 10000
62046 struct region {          /* Defines "structure" of shared memory */
62047     int len;
62048     char buf[MAX_LEN];
62049 };
62050 struct region *rptr;
62051 int fd;
62052
62053 /* Create shared memory object and set its size */
62054 fd = shm_open("/myregion", O_CREAT | O_RDWR, S_IRUSR | S_IWUSR);
62055 if (fd == -1)
62056     /* Handle error */;
62057
62058 if (ftruncate(fd, sizeof(struct region)) == -1)
62059     /* Handle error */;
62060
62061 /* Map shared memory object */
62062 rptr = mmap(NULL, sizeof(struct region),
62063             PROT_READ | PROT_WRITE, MAP_SHARED, fd, 0);
62064 if (rptr == MAP_FAILED)
62065     /* Handle error */;
62066
62067 /* Now we can refer to mapped region using fields of rptr;
62068    for example, rptr->len */
62069 ...
```

62066 **APPLICATION USAGE**

62067 None.

62068 **RATIONALE**

62069 When the Memory Mapped Files option is supported, the normal *open()* call is used to obtain a  
 62070 descriptor to a file to be mapped according to existing practice with *mmap()*. When the Shared  
 62071 Memory Objects option is supported, the *shm\_open()* function shall obtain a descriptor to the  
 62072 shared memory object to be mapped.

62073 There is ample precedent for having a file descriptor represent several types of objects. In the  
 62074 POSIX.1-1990 standard, a file descriptor can represent a file, a pipe, a FIFO, a tty, or a directory.  
 62075 Many implementations simply have an operations vector, which is indexed by the file descriptor  
 62076 type and does very different operations. Note that in some cases the file descriptor passed to  
 62077 generic operations on file descriptors is returned by *open()* or *creat()* and in some cases returned  
 62078 by alternate functions, such as *pipe()*. The latter technique is used by *shm\_open()*.

62079 Note that such shared memory objects can actually be implemented as mapped files. In both  
 62080 cases, the size can be set after the open using *ftruncate()*. The *shm\_open()* function itself does not

create a shared object of a specified size because this would duplicate an extant function that set the size of an object referenced by a file descriptor.

On implementations where memory objects are implemented using the existing file system, the *shm\_open()* function may be implemented using a macro that invokes *open()*, and the *shm\_unlink()* function may be implemented using a macro that invokes *unlink()*.

For implementations without a permanent file system, the definition of the name of the memory objects is allowed not to survive a system reboot. Note that this allows systems with a permanent file system to implement memory objects as data structures internal to the implementation as well.

On implementations that choose to implement memory objects using memory directly, a *shm\_open()* followed by an *ftruncate()* and *close()* can be used to preallocate a shared memory area and to set the size of that preallocation. This may be necessary for systems without virtual memory hardware support in order to ensure that the memory is contiguous.

The set of valid open flags to *shm\_open()* was restricted to *O\_RDONLY*, *O\_RDWR*, *O\_CREAT*, and *O\_TRUNC* because these could be easily implemented on most memory mapping systems. This volume of POSIX.1-2008 is silent on the results if the implementation cannot supply the requested file access because of implementation-defined reasons, including hardware ones.

The error conditions [EACCES] and [ENOTSUP] are provided to inform the application that the implementation cannot complete a request.

[EACCES] indicates for implementation-defined reasons, probably hardware-related, that the implementation cannot comply with a requested mode because it conflicts with another requested mode. An example might be that an application desires to open a memory object two times, mapping different areas with different access modes. If the implementation cannot map a single area into a process space in two places, which would be required if different access modes were required for the two areas, then the implementation may inform the application at the time of the second open.

[ENOTSUP] indicates for implementation-defined reasons, probably hardware-related, that the implementation cannot comply with a requested mode at all. An example would be that the hardware of the implementation cannot support write-only shared memory areas.

On all implementations, it may be desirable to restrict the location of the memory objects to specific file systems for performance (such as a RAM disk) or implementation-defined reasons (shared memory supported directly only on certain file systems). The *shm\_open()* function may be used to enforce these restrictions. There are a number of methods available to the application to determine an appropriate name of the file or the location of an appropriate directory. One way is from the environment via *getenv()*. Another would be from a configuration file.

This volume of POSIX.1-2008 specifies that memory objects have initial contents of zero when created. This is consistent with current behavior for both files and newly allocated memory. For those implementations that use physical memory, it would be possible that such implementations could simply use available memory and give it to the process uninitialized. This, however, is not consistent with standard behavior for the uninitialized data area, the stack, and of course, files. Finally, it is highly desirable to set the allocated memory to zero for security reasons. Thus, initializing memory objects to zero is required.

## FUTURE DIRECTIONS

A future version might require the *shm\_open()* and *shm\_unlink()* functions to have semantics similar to normal file system operations.

62126 **SEE ALSO**

62127       Section 2.14 (on page 549), *close()*, *dup()*, *exec*, *fcntl()*, *mmap()*, *shmat()*, *shmctl()*, *shmdt()*,  
62128       *shm\_unlink()*, *umask()*

62129       XBD <fcntl.h>, <sys/mman.h>

62130 **CHANGE HISTORY**

62131       First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

62132 **Issue 6**

62133       The *shm\_open()* function is marked as part of the Shared Memory Objects option.

62134       The [ENOSYS] error condition has been removed as stubs need not be provided if an  
62135       implementation does not support the Shared Memory Objects option.

62136       IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/126 is applied, adding the example to the  
62137       EXAMPLES section.

62138 **Issue 7**

62139       Austin Group Interpretation 1003.1-2001 #077 is applied, clarifying the *name* argument and  
62140       changing [ENAMETOOLONG] from a “shall fail” to a “may fail” error.

62141       Austin Group Interpretation 1003.1-2001 #141 is applied, adding FUTURE DIRECTIONS.

62142       SD5-XSH-ERN-170 is applied, updating the DESCRIPTION to clarify the wording for setting the  
62143       user ID and group ID of the shared memory object.

**NAME**

shm\_unlink — remove a shared memory object (**REALTIME**)

**SYNOPSIS**

```
#include <sys/mman.h>

int shm_unlink(const char *name);
```

**DESCRIPTION**

The *shm\_unlink()* function shall remove the name of the shared memory object named by the string pointed to by *name*.

If one or more references to the shared memory object exist when the object is unlinked, the name shall be removed before *shm\_unlink()* returns, but the removal of the memory object contents shall be postponed until all open and map references to the shared memory object have been removed.

Even if the object continues to exist after the last *shm\_unlink()*, reuse of the name shall subsequently cause *shm\_open()* to behave as if no shared memory object of this name exists (that is, *shm\_open()* will fail if **O\_CREAT** is not set, or will create a new shared memory object if **O\_CREAT** is set).

**RETURN VALUE**

Upon successful completion, a value of zero shall be returned. Otherwise, a value of **-1** shall be returned and *errno* set to indicate the error. If **-1** is returned, the named shared memory object shall not be changed by this function call.

**ERRORS**

The *shm\_unlink()* function shall fail if:

[**EACCES**] Permission is denied to unlink the named shared memory object.

[**ENOENT**] The named shared memory object does not exist.

The *shm\_unlink()* function may fail if:

[**ENAMETOOLONG**]

The length of the *name* argument exceeds **{\_POSIX\_PATH\_MAX}** on systems that do not support the XSI option or exceeds **{\_XOPEN\_PATH\_MAX}** on XSI systems, or has a pathname component that is longer than **{\_POSIX\_NAME\_MAX}** on systems that do not support the XSI option or longer than **{\_XOPEN\_NAME\_MAX}** on XSI systems. A call to *shm\_unlink()* with a *name* argument that contains the same shared memory object name as was previously used in a successful *shm\_open()* call shall not give an [**ENAMETOOLONG**] error.

**EXAMPLES**

None.

**APPLICATION USAGE**

Names of memory objects that were allocated with *open()* are deleted with *unlink()* in the usual fashion. Names of memory objects that were allocated with *shm\_open()* are deleted with *shm\_unlink()*. Note that the actual memory object is not destroyed until the last close and unmap on it have occurred if it was already in use.

62185 **RATIONALE**

62186 None.

62187 **FUTURE DIRECTIONS**

62188 A future version might require the *shm\_open()* and *shm\_unlink()* functions to have semantics  
62189 similar to normal file system operations.

62190 **SEE ALSO**

62191 *close(), mmap(), munmap(), shmat(), shmctl(), shmdt(), shm\_open()*

62192 XBD <sys/mman.h>

62193 **CHANGE HISTORY**

62194 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

62195 **Issue 6**

62196 The *shm\_unlink()* function is marked as part of the Shared Memory Objects option.

62197 In the DESCRIPTION, text is added to clarify that reusing the same name after a *shm\_unlink()*  
62198 will not attach to the old shared memory object.

62199 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
62200 implementation does not support the Shared Memory Objects option.

62201 **Issue 7**

62202 Austin Group Interpretation 1003.1-2001 #077 is applied, changing [ENAMETOOLONG] from a  
62203 “shall fail” to a “may fail” error.

62204 Austin Group Interpretation 1003.1-2001 #141 is applied, adding FUTURE DIRECTIONS.

62205 **NAME**

62206 shmat — XSI shared memory attach operation

62207 **SYNOPSIS**

```
62208 XSI      #include <sys/shm.h>
62209      void *shmat(int shmid, const void *shmaddr, int shmflg);
```

62210 **DESCRIPTION**

62211 The *shmat()* function operates on XSI shared memory (see XBD [Section 3.346](#), on page 89). It is  
 62212 unspecified whether this function interoperates with the realtime interprocess communication  
 62213 facilities defined in [Section 2.8](#) (on page 503).

62214 The *shmat()* function attaches the shared memory segment associated with the shared memory  
 62215 identifier specified by *shmid* to the address space of the calling process. The segment is attached  
 62216 at the address specified by one of the following criteria:

- 62217 • If *shmaddr* is a null pointer, the segment is attached at the first available address as selected  
 62218 by the system.
- 62219 • If *shmaddr* is not a null pointer and (*shmflg* &SHM\_RND) is non-zero, the segment is  
 62220 attached at the address given by (*shmaddr* - ((uintptr\_t)*shmaddr* %SHMLBA)). The character  
 62221 ' % ' is the C-language remainder operator.
- 62222 • If *shmaddr* is not a null pointer and (*shmflg* &SHM\_RND) is 0, the segment is attached at  
 62223 the address given by *shmaddr*.
- 62224 • The segment is attached for reading if (*shmflg* &SHM\_RDONLY) is non-zero and the  
 62225 calling process has read permission; otherwise, if it is 0 and the calling process has read  
 62226 and write permission, the segment is attached for reading and writing.

62227 **RETURN VALUE**

62228 Upon successful completion, *shmat()* shall increment the value of *shm\_nattch* in the data  
 62229 structure associated with the shared memory ID of the attached shared memory segment and  
 62230 return the segment's start address. Also, the *shm\_atime* timestamp shall be set to the current  
 62231 time, as described in [Section 2.7.1](#) (on page 502).

62232 Otherwise, the shared memory segment shall not be attached, *shmat()* shall return (**void \***)−1, |  
 62233 and *errno* shall be set to indicate the error.

62234 **ERRORS**

62235 The *shmat()* function shall fail if:

- |                                           |          |                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------------------------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 62236<br>62237                            | [EACCES] | Operation permission is denied to the calling process; see <a href="#">Section 2.7</a> (on page 501).                                                                                                                                                                                                                                                                                                     |
| 62238<br>62239<br>62240<br>62241<br>62242 | [EINVAL] | The value of <i>shmid</i> is not a valid shared memory identifier, the <i>shmaddr</i> is not a null pointer, and the value of ( <i>shmaddr</i> - ((uintptr_t) <i>shmaddr</i> %SHMLBA)) is an illegal address for attaching shared memory; or the <i>shmaddr</i> is not a null pointer, ( <i>shmflg</i> &SHM_RND) is 0, and the value of <i>shmaddr</i> is an illegal address for attaching shared memory. |
| 62243<br>62244                            | [EMFILE] | The number of shared memory segments attached to the calling process would exceed the system-imposed limit.                                                                                                                                                                                                                                                                                               |
| 62245<br>62246                            | [ENOMEM] | The available data space is not large enough to accommodate the shared memory segment.                                                                                                                                                                                                                                                                                                                    |



**62247 EXAMPLES**

62248 None.

**62249 APPLICATION USAGE**

62250 The POSIX Realtime Extension defines alternative interfaces for interprocess communication.  
62251 Application developers who need to use IPC should design their applications so that modules  
62252 using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the  
62253 alternative interfaces.

**62254 RATIONALE**

62255 None.

**62256 FUTURE DIRECTIONS**

62257 None.

**62258 SEE ALSO**

62259 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 503), *exec*, *exit()*, *fork()*, *shmctl()*, *shmdt()*,  
62260 *shmget()*, *shm\_open()*, *shm\_unlink()*  
62261 XBD [Section 3.346](#) (on page 89), [<sys/shm.h>](#)

**62262 CHANGE HISTORY**

62263 First released in Issue 2. Derived from Issue 2 of the SVID.

**62264 Issue 5**

62265 Moved from SHARED MEMORY to BASE.

62266 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE  
62267 DIRECTIONS to a new APPLICATION USAGE section.

**62268 Issue 6**

62269 The Open Group Corrigendum U021/13 is applied.

**62270 Issue 7**

62271 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0572 [345] is applied.



62272 **NAME**

62273 shmctl — XSI shared memory control operations

62274 **SYNOPSIS**

```
62275 XSI      #include <sys/shm.h>
62276      int shmctl(int shmid, int cmd, struct shmids *buf);
```

62277 **DESCRIPTION**

62278 The *shmctl()* function operates on XSI shared memory (see XBD [Section 3.346](#), on page 89). It is  
 62279 unspecified whether this function interoperates with the realtime interprocess communication  
 62280 facilities defined in [Section 2.8](#) (on page 503).

62281 The *shmctl()* function provides a variety of shared memory control operations as specified by  
 62282 *cmd*. The following values for *cmd* are available:

62283 **IPC\_STAT** Place the current value of each member of the **shmids** data structure  
 62284 associated with *shmid* into the structure pointed to by *buf*. The contents of the  
 62285 structure are defined in **<sys/shm.h>**.

62286 **IPC\_SET** Set the value of the following members of the **shmids** data structure  
 62287 associated with *shmid* to the corresponding value found in the structure  
 62288 pointed to by *buf*:

```
62289     shm_perm.uid
62290     shm_perm.gid
62291     shm_perm.mode    Low-order nine bits.
```

62292 Also, the *shm\_ctime* timestamp shall be set to the current time, as described in  
 62293 [Section 2.7.1](#) (on page 502).

62294 **IPC\_SET** can only be executed by a process that has an effective user ID equal  
 62295 to either that of a process with appropriate privileges or to the value of  
 62296 *shm\_perm.cuid* or *shm\_perm.uid* in the **shmids** data structure associated with  
 62297 *shmid*.

62298 **IPC\_RMID** Remove the shared memory identifier specified by *shmid* from the system and  
 62299 destroy the shared memory segment and **shmids** data structure associated  
 62300 with it. **IPC\_RMID** can only be executed by a process that has an effective user  
 62301 ID equal to either that of a process with appropriate privileges or to the value  
 62302 of *shm\_perm.cuid* or *shm\_perm.uid* in the **shmids** data structure associated  
 62303 with *shmid*.

62304 **RETURN VALUE**

62305 Upon successful completion, *shmctl()* shall return 0; otherwise, it shall return -1 and set *errno* to  
 62306 indicate the error.

62307 **ERRORS**

62308 The *shmctl()* function shall fail if:

62309 **[EACCES]** The argument *cmd* is equal to **IPC\_STAT** and the calling process does not have  
 62310 read permission; see [Section 2.7](#) (on page 501).

62311 **[EINVAL]** The value of *shmid* is not a valid shared memory identifier, or the value of *cmd*  
 62312 is not a valid command.

62313 **[EPERM]** The argument *cmd* is equal to **IPC\_RMID** or **IPC\_SET** and the effective user ID  
 62314 of the calling process is not equal to that of a process with appropriate  
 62315 privileges and it is not equal to the value of *shm\_perm.cuid* or *shm\_perm.uid* in

62316 the data structure associated with *shmid*.

62317 The *shmctl()* function may fail if:

62318 [EOVERFLOW] The *cmd* argument is IPC\_STAT and the *gid* or *uid* value is too large to be  
62319 stored in the structure pointed to by the *buf* argument.

#### 62320 EXAMPLES

62321 None.

#### 62322 APPLICATION USAGE

62323 The POSIX Realtime Extension defines alternative interfaces for interprocess communication.  
62324 Application developers who need to use IPC should design their applications so that modules  
62325 using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the  
62326 alternative interfaces.

#### 62327 RATIONALE

62328 None.

#### 62329 FUTURE DIRECTIONS

62330 None.

#### 62331 SEE ALSO

62332 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 503), *shmat()*, *shmdt()*, *shmget()*, *shm\_open()*,  
62333 *shm\_unlink()*

62334 XBD [Section 3.346](#) (on page 89), [<sys/shm.h>](#)

#### 62335 CHANGE HISTORY

62336 First released in Issue 2. Derived from Issue 2 of the SVID.

##### 62337 Issue 5

62338 Moved from SHARED MEMORY to BASE.

62339 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE  
62340 DIRECTIONS to a new APPLICATION USAGE section.

##### 62341 Issue 7

62342 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0573 [345] is applied.

62343 **NAME**

62344 shmdt — XSI shared memory detach operation

62345 **SYNOPSIS**

```
62346 XSI      #include <sys/shm.h>
62347      int shmdt(const void *shmaddr);
```

62348 **DESCRIPTION**

62349 The *shmdt()* function operates on XSI shared memory (see XBD [Section 3.346](#), on page 89). It is  
 62350 unspecified whether this function interoperates with the realtime interprocess communication  
 62351 facilities defined in [Section 2.8](#) (on page 503).

62352 The *shmdt()* function detaches the shared memory segment located at the address specified by  
 62353 *shmaddr* from the address space of the calling process.

62354 **RETURN VALUE**

62355 Upon successful completion, *shmdt()* shall decrement the value of *shm\_nattch* in the data  
 62356 structure associated with the shared memory ID of the attached shared memory segment and  
 62357 return 0. Also, the *shm\_dtime* timestamp shall be set to the current time, as described in [Section](#)  
 62358 [2.7.1](#) (on page 502).

62359 Otherwise, the shared memory segment shall not be detached, *shmdt()* shall return -1, and *errno*  
 62360 shall be set to indicate the error.

62361 **ERRORS**

62362 The *shmdt()* function shall fail if:

|       |          |                                                                                               |
|-------|----------|-----------------------------------------------------------------------------------------------|
| 62363 | [EINVAL] | The value of <i>shmaddr</i> is not the data segment start address of a shared memory segment. |
| 62364 |          |                                                                                               |

62365 **EXAMPLES**

62366 None.

62367 **APPLICATION USAGE**

62368 The POSIX Realtime Extension defines alternative interfaces for interprocess communication.  
 62369 Application developers who need to use IPC should design their applications so that modules  
 62370 using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the  
 62371 alternative interfaces.

62372 **RATIONALE**

62373 None.

62374 **FUTURE DIRECTIONS**

62375 None.

62376 **SEE ALSO**

62377 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 503), *exec*, *exit()*, *fork()*, *shmat()*, *shmctl()*,  
 62378 *shmget()*, *shm\_open()*, *shm\_unlink()*

62379 XBD [Section 3.346](#) (on page 89), [<sys/shm.h>](#)

62380 **CHANGE HISTORY**

62381 First released in Issue 2. Derived from Issue 2 of the SVID.

62382 **Issue 5**

62383 Moved from SHARED MEMORY to BASE.

62384 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE  
62385 DIRECTIONS to a new APPLICATION USAGE section.62386 **Issue 7**

62387 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0574 [345] is applied.

62388 **NAME**

62389 shmget — get an XSI shared memory segment

62390 **SYNOPSIS**

```
62391 XSI    #include <sys/shm.h>
62392      int shmget(key_t key, size_t size, int shmflg);
```

62393 **DESCRIPTION**

62394 The *shmget()* function operates on XSI shared memory (see XBD [Section 3.346](#), on page 89). It is  
 62395 unspecified whether this function interoperates with the realtime interprocess communication  
 62396 facilities defined in [Section 2.8](#) (on page 503).

62397 The *shmget()* function shall return the shared memory identifier associated with *key*.

62398 A shared memory identifier, associated data structure, and shared memory segment of at least  
 62399 *size* bytes (see [<sys/shm.h>](#)) are created for *key* if one of the following is true:

- 62400 • The argument *key* is equal to `IPC_PRIVATE`.
- 62401 • The argument *key* does not already have a shared memory identifier associated with it and  
 62402 (*shmflg* & `IPC_CREAT`) is non-zero.

62403 Upon creation, the data structure associated with the new shared memory identifier shall be  
 62404 initialized as follows:

- 62405 • The values of *shm\_perm.cuid*, *shm\_perm.uid*, *shm\_perm.cgid*, and *shm\_perm.gid* are set to the  
 62406 effective user ID and effective group ID, respectively, of the calling process.
- 62407 • The low-order nine bits of *shm\_perm.mode* are set to the low-order nine bits of *shmflg*.
- 62408 • The value of *shm\_segsz* is set to the value of *size*.
- 62409 • The values of *shm\_lpid*, *shm\_nattch*, *shm\_atime*, and *shm\_dtime* are set to 0.
- 62410 • The value of *shm\_ctime* is set to the current time, as described in [Section 2.7.1](#) (on page 502).

62411 When the shared memory segment is created, it shall be initialized with all zero values.

62412 **RETURN VALUE**

62413 Upon successful completion, *shmget()* shall return a non-negative integer, namely a shared  
 62414 memory identifier; otherwise, it shall return -1 and set *errno* to indicate the error.

62415 **ERRORS**

62416 The *shmget()* function shall fail if:

- |                         |          |                                                                                                                                                                                                         |
|-------------------------|----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 62417<br>62418<br>62419 | [EACCES] | A shared memory identifier exists for <i>key</i> but operation permission as specified by the low-order nine bits of <i>shmflg</i> would not be granted; see <a href="#">Section 2.7</a> (on page 501). |
| 62420<br>62421          | [EEXIST] | A shared memory identifier exists for the argument <i>key</i> but ( <i>shmflg</i> & <code>IPC_CREAT</code> ) && ( <i>shmflg</i> & <code>IPC_EXCL</code> ) is non-zero.                                  |
| 62422<br>62423          | [EINVAL] | A shared memory segment is to be created and the value of <i>size</i> is less than the system-imposed minimum or greater than the system-imposed maximum.                                               |
| 62424<br>62425          | [EINVAL] | No shared memory segment is to be created and a shared memory segment exists for <i>key</i> but the size of the segment associated with it is less than <i>size</i> .                                   |
| 62426<br>62427          | [ENOENT] | A shared memory identifier does not exist for the argument <i>key</i> and ( <i>shmflg</i> & <code>IPC_CREAT</code> ) is 0.                                                                              |

62428 [ENOMEM] A shared memory identifier and associated shared memory segment are to be  
62429 created, but the amount of available physical memory is not sufficient to fill  
62430 the request.

62431 [ENOSPC] A shared memory identifier is to be created, but the system-imposed limit on  
62432 the maximum number of allowed shared memory identifiers system-wide  
62433 would be exceeded.

#### 62434 EXAMPLES

62435 None.

#### 62436 APPLICATION USAGE

62437 The POSIX Realtime Extension defines alternative interfaces for interprocess communication.  
62438 Application developers who need to use IPC should design their applications so that modules  
62439 using the IPC routines described in [Section 2.7](#) (on page 501) can be easily modified to use the  
62440 alternative interfaces.

#### 62441 RATIONALE

62442 None.

#### 62443 FUTURE DIRECTIONS

62444 None.

#### 62445 SEE ALSO

62446 [Section 2.7](#) (on page 501), [Section 2.8](#) (on page 503), [ftok\(\)](#), [shmat\(\)](#), [shmctl\(\)](#), [shmdt\(\)](#), [shm\\_open\(\)](#),  
62447 [shm\\_unlink\(\)](#)

62448 XBD [Section 3.346](#) (on page 89), [<sys/shm.h>](#)

#### 62449 CHANGE HISTORY

62450 First released in Issue 2. Derived from Issue 2 of the SVID.

##### 62451 Issue 5

62452 Moved from SHARED MEMORY to BASE.

62453 The note about use of POSIX Realtime Extension IPC routines has been moved from FUTURE  
62454 DIRECTIONS to a new APPLICATION USAGE section.

##### 62455 Issue 7

62456 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0575 [345], XSH/TC1-2008/0576 [363],  
62457 and XSH/TC1-2008/0577 [344] are applied.

62458 **NAME**

62459 shutdown — shut down socket send and receive operations

62460 **SYNOPSIS**

62461 #include &lt;sys/socket.h&gt;

62462 int shutdown(int *socket*, int *how*);62463 **DESCRIPTION**62464 The *shutdown()* function shall cause all or part of a full-duplex connection on the socket  
62465 associated with the file descriptor *socket* to be shut down.62466 The *shutdown()* function takes the following arguments:62467 *socket* Specifies the file descriptor of the socket.62468 *how* Specifies the type of shutdown. The values are as follows:

62469 SHUT\_RD Disables further receive operations.

62470 SHUT\_WR Disables further send operations.

62471 SHUT\_RDWR Disables further send and receive operations.

62472 The *shutdown()* function disables subsequent send and/or receive operations on a socket,  
62473 depending on the value of the *how* argument.62474 **RETURN VALUE**62475 Upon successful completion, *shutdown()* shall return 0; otherwise, -1 shall be returned and *errno*  
62476 set to indicate the error.62477 **ERRORS**62478 The *shutdown()* function shall fail if:62479 [EBADF] The *socket* argument is not a valid file descriptor.62480 [EINVAL] The *how* argument is invalid.

62481 [ENOTCONN] The socket is not connected.

62482 [ENOTSOCK] The *socket* argument does not refer to a socket.62483 The *shutdown()* function may fail if:

62484 [ENOBUFS] Insufficient resources were available in the system to perform the operation.

62485 **EXAMPLES**

62486 None.

62487 **APPLICATION USAGE**

62488 None.

62489 **RATIONALE**

62490 None.

62491 **FUTURE DIRECTIONS**

62492 None.

62493 **SEE ALSO**62494 *getsockopt()*, *pselect()*, *read()*, *recv()*, *recvfrom()*, *recvmsg()*, *send()*, *sendto()*, *setsockopt()*, *socket()*,  
62495 *write()*

62496 XBD &lt;sys/socket.h&gt;

|       |                                                                           |
|-------|---------------------------------------------------------------------------|
| 62497 | <b>CHANGE HISTORY</b>                                                     |
| 62498 | First released in Issue 6. Derived from the XNS, Issue 5.2 specification. |



62499 **NAME**

62500 sigaction — examine and change a signal action

62501 **SYNOPSIS**

```
62502 CX    #include <signal.h>
62503
62503    int sigaction(int sig, const struct sigaction *restrict act,
62504                  struct sigaction *restrict oact);
```

62505 **DESCRIPTION**

62506 The *sigaction()* function allows the calling process to examine and/or specify the action to be  
 62507 associated with a specific signal. The argument *sig* specifies the signal; acceptable values are  
 62508 defined in **<signal.h>**.

62509 The structure **sigaction**, used to describe an action to be taken, is defined in the **<signal.h>**  
 62510 header to include at least the following members:

| Member Type                               | Member Name         | Description                                                                           |
|-------------------------------------------|---------------------|---------------------------------------------------------------------------------------|
| <b>void(*) (int)</b>                      | <i>sa_handler</i>   | Pointer to a signal-catching function or one of the macros SIG_IGN or SIG_DFL.        |
| <b>sigset_t</b>                           | <i>sa_mask</i>      | Additional set of signals to be blocked during execution of signal-catching function. |
| <b>int</b>                                | <i>sa_flags</i>     | Special flags to affect behavior of signal.                                           |
| <b>void(*) (int, siginfo_t *, void *)</b> | <i>sa_sigaction</i> | Pointer to a signal-catching function.                                                |

62520 The storage occupied by *sa\_handler* and *sa\_sigaction* may overlap, and a conforming application  
 62521 shall not use both simultaneously.

62522 If the argument *act* is not a null pointer, it points to a structure specifying the action to be  
 62523 associated with the specified signal. If the argument *oact* is not a null pointer, the action  
 62524 previously associated with the signal is stored in the location pointed to by the argument *oact*. If  
 62525 the argument *act* is a null pointer, signal handling is unchanged; thus, the call can be used to  
 62526 enquire about the current handling of a given signal. The SIGKILL and SIGSTOP signals shall  
 62527 not be added to the signal mask using this mechanism; this restriction shall be enforced by the  
 62528 system without causing an error to be indicated.

62529 If the SA\_SIGINFO flag (see below) is cleared in the *sa\_flags* field of the **sigaction** structure, the  
 62530 *sa\_handler* field identifies the action to be associated with the specified signal. If the  
 62531 SA\_SIGINFO flag is set in the *sa\_flags* field, the *sa\_sigaction* field specifies a signal-catching  
 62532 function.

62533 The *sa\_flags* field can be used to modify the behavior of the specified signal.

62534 The following flags, defined in the **<signal.h>** header, can be set in *sa\_flags*:

62535 XSI SA\_NOCLDSTOP Do not generate SIGCHLD when children stop or stopped children  
 62536 continue.

62537 If *sig* is SIGCHLD and the SA\_NOCLDSTOP flag is not set in *sa\_flags*, and  
 62538 the implementation supports the SIGCHLD signal, then a SIGCHLD  
 62539 signal shall be generated for the calling process whenever any of its child  
 62540 XSI processes stop and a SIGCHLD signal may be generated for the calling  
 62541 process whenever any of its stopped child processes are continued. If *sig*  
 62542 is SIGCHLD and the SA\_NOCLDSTOP flag is set in *sa\_flags*, then the  
 62543 implementation shall not generate a SIGCHLD signal in this way.

|       |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------|-----|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 62544 | XSI | SA_ONSTACK   | If set and an alternate signal stack has been declared with <i>sigaltstack()</i> , the signal shall be delivered to the calling process on that stack. Otherwise, the signal shall be delivered on the current stack.                                                                                                                                                                                                                                                                                                                                                                                                           |
| 62545 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62546 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62547 |     | SA_RESETHAND | If set, the disposition of the signal shall be reset to SIG_DFL and the SA_SIGINFO flag shall be cleared on entry to the signal handler.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 62548 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62549 |     |              | <b>Note:</b> SIGILL and SIGTRAP cannot be automatically reset when delivered;                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 62550 |     |              | the system silently enforces this restriction.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 62551 |     |              | Otherwise, the disposition of the signal shall not be modified on entry to the signal handler.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 62552 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62553 |     |              | In addition, if this flag is set, <i>sigaction()</i> may behave as if the SA_NODEFER flag were also set.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 62554 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62555 |     | SA_RESTART   | This flag affects the behavior of interruptible functions; that is, those specified to fail with <i>errno</i> set to [EINTR]. If set, and a function specified as interruptible is interrupted by this signal, the function shall restart and shall not fail with [EINTR] unless otherwise specified. If an interruptible function which uses a timeout is restarted, the duration of the timeout following the restart is set to an unspecified value that does not exceed the original timeout value. If the flag is not set, interruptible functions interrupted by this signal shall fail with <i>errno</i> set to [EINTR]. |
| 62556 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62557 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62558 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62559 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62560 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62561 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62562 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62563 |     | SA_SIGINFO   | If cleared and the signal is caught, the signal-catching function shall be entered as:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 62564 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62565 |     |              | <pre>void func(int signo);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 62566 |     |              | where <i>signo</i> is the only argument to the signal-catching function. In this case, the application shall use the <i>sa_handler</i> member to describe the signal-catching function and the application shall not modify the <i>sa_sigaction</i> member.                                                                                                                                                                                                                                                                                                                                                                     |
| 62567 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62568 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62569 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62570 |     |              | If SA_SIGINFO is set and the signal is caught, the signal-catching function shall be entered as:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 62571 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62572 |     |              | <pre>void func(int signo, siginfo_t *info, void *context);</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 62573 |     |              | where two additional arguments are passed to the signal-catching function. The second argument shall point to an object of type <b>siginfo_t</b> explaining the reason why the signal was generated; the third argument can be cast to a pointer to an object of type <b>ucontext_t</b> to refer to the receiving thread's context that was interrupted when the signal was delivered. In this case, the application shall use the <i>sa_sigaction</i> member to describe the signal-catching function and the application shall not modify the <i>sa_handler</i> member.                                                       |
| 62574 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62575 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62576 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62577 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62578 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62579 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62580 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62581 |     |              | The <i>si_signo</i> member contains the system-generated signal number.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 62582 | XSI |              | The <i>si_errno</i> member may contain implementation-defined additional error information; if non-zero, it contains an error number identifying the condition that caused the signal to be generated.                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 62583 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62584 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 62585 |     |              | The <i>si_code</i> member contains a code identifying the cause of the signal, as described in <a href="#">Section 2.4.3</a> (on page 490).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 62586 |     |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |

|       |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
|-------|-----|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|
| 62587 | XSI | SA_NOCLDWAIT        | If <i>sig</i> does not equal SIGCHLD, the behavior is unspecified. Otherwise, the behavior of the SA_NOCLDWAIT flag is as specified in <a href="#">Consequences of Process Termination</a> (on page 553).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |  |
| 62588 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62589 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62590 |     | SA_NODEFER          | If set and <i>sig</i> is caught, <i>sig</i> shall not be added to the thread's signal mask on entry to the signal handler unless it is included in <i>sa_mask</i> . Otherwise, <i>sig</i> shall always be added to the thread's signal mask on entry to the signal handler.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |  |
| 62591 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62592 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62593 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62594 |     |                     | When a signal is caught by a signal-catching function installed by <i>sigaction()</i> , a new signal mask is calculated and installed for the duration of the signal-catching function (or until a call to either <i>sigprocmask()</i> or <i>sigsuspend()</i> is made). This mask is formed by taking the union of the current signal mask and the value of the <i>sa_mask</i> for the signal being delivered, and unless SA_NODEFER or SA_RESETHAND is set, then including the signal being delivered. If and when the user's signal handler returns normally, the original signal mask is restored.                                                                                                                                                                                     |  |
| 62595 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62596 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62597 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62598 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62599 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62600 |     |                     | Once an action is installed for a specific signal, it shall remain installed until another action is explicitly requested (by another call to <i>sigaction()</i> ), until the SA_RESETHAND flag causes resetting of the handler, or until one of the <i>exec</i> functions is called.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |  |
| 62601 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62602 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62603 |     |                     | If the previous action for <i>sig</i> had been established by <i>signal()</i> , the values of the fields returned in the structure pointed to by <i>oact</i> are unspecified, and in particular <i>oact-&gt;sa_handler</i> is not necessarily the same value passed to <i>signal()</i> . However, if a pointer to the same structure or a copy thereof is passed to a subsequent call to <i>sigaction()</i> via the <i>act</i> argument, handling of the signal shall be as if the original call to <i>signal()</i> were repeated.                                                                                                                                                                                                                                                        |  |
| 62604 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62605 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62606 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62607 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62608 |     |                     | If <i>sigaction()</i> fails, no new signal handler is installed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |  |
| 62609 |     |                     | It is unspecified whether an attempt to set the action for a signal that cannot be caught or ignored to SIG_DFL is ignored or causes an error to be returned with <i>errno</i> set to [EINVAL].                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62610 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62611 |     |                     | If SA_SIGINFO is not set in <i>sa_flags</i> , then the disposition of subsequent occurrences of <i>sig</i> when it is already pending is implementation-defined; the signal-catching function shall be invoked with a single argument. If SA_SIGINFO is set in <i>sa_flags</i> , then subsequent occurrences of <i>sig</i> generated by <i>sigqueue()</i> or as a result of any signal-generating function that supports the specification of an application-defined value (when <i>sig</i> is already pending) shall be queued in FIFO order until delivered or accepted; the signal-catching function shall be invoked with three arguments. The application specified value is passed to the signal-catching function as the <i>si_value</i> member of the <b>siginfo_t</b> structure. |  |
| 62612 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62613 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62614 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62615 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62616 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62617 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62618 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62619 |     |                     | The result of the use of <i>sigaction()</i> and a <i>sigwait()</i> function concurrently within a process on the same signal is unspecified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |  |
| 62620 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62621 |     | <b>RETURN VALUE</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |
| 62622 |     |                     | Upon successful completion, <i>sigaction()</i> shall return 0; otherwise, -1 shall be returned, <i>errno</i> shall be set to indicate the error, and no new signal-catching function shall be installed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |  |
| 62623 |     |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |  |

**ERRORS**

The *sigaction()* function shall fail if:

[EINVAL] The *sig* argument is not a valid signal number or an attempt is made to catch a signal that cannot be caught or ignore a signal that cannot be ignored.

The *sigaction()* function may fail if:

[EINVAL] An attempt was made to set the action to SIG\_DFL for a signal that cannot be caught or ignored (or both).

In addition, the *sigaction()* function may fail if the SA\_SIGINFO flag is set in the *sa\_flags* field of the **sigaction** structure for a signal not in the range SIGRTMIN to SIGRTMAX.

**EXAMPLES****Establishing a Signal Handler**

The following example demonstrates the use of *sigaction()* to establish a handler for the SIGINT signal.

```
#include <signal.h>

static void handler(int signum)
{
    /* Take appropriate actions for signal delivery */
}

int main()
{
    struct sigaction sa;

    sa.sa_handler = handler;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = SA_RESTART; /* Restart functions if
                               interrupted by handler */
    if (sigaction(SIGINT, &sa, NULL) == -1)
        /* Handle error */;

    /* Further code */
}
```

**APPLICATION USAGE**

The *sigaction()* function supersedes the *signal()* function, and should be used in preference. In particular, *sigaction()* and *signal()* should not be used in the same process to control the same signal. The behavior of async-signal-safe functions, as defined in their respective DESCRIPTION sections, is as specified by this volume of POSIX.1-2008, regardless of invocation from a signal-catching function. This is the only intended meaning of the statement that async-signal-safe functions may be used in signal-catching functions without restrictions. Applications must still consider all effects of such functions on such things as data structures, files, and process state. In particular, application developers need to consider the restrictions on interactions when interrupting *sleep()* and interactions among multiple handles for a file description. The fact that any specific function is listed as async-signal-safe does not necessarily mean that invocation of that function from a signal-catching function is recommended.

In order to prevent errors arising from interrupting non-async-signal-safe function calls, applications should protect calls to these functions either by blocking the appropriate signals or through the use of some programmatic semaphore (see *semget()*, *sem\_init()*, *sem\_open()*, and so

on). Note in particular that even the “safe” functions may modify *errno*; the signal-catching function, if not executing as an independent thread, should save and restore its value in order to avoid the possibility that delivery of a signal in between an error return from a function that sets *errno* and the subsequent examination of *errno* could result in the signal-catching function changing the value of *errno*. Naturally, the same principles apply to the async-signal-safety of application routines and asynchronous data access. Note that *longjmp()* and *siglongjmp()* are not in the list of async-signal-safe functions. This is because the code executing after *longjmp()* and *siglongjmp()* can call any unsafe functions with the same danger as calling those unsafe functions directly from the signal handler. Applications that use *longjmp()* and *siglongjmp()* from within signal handlers require rigorous protection in order to be portable. Many of the other functions that are excluded from the list are traditionally implemented using either *malloc()* or *free()* functions or the standard I/O library, both of which traditionally use data structures in a non-async-signal-safe manner. Since any combination of different functions using a common data structure can cause async-signal-safety problems, this volume of POSIX.1-2008 does not define the behavior when any unsafe function is called in a signal handler that interrupts an unsafe function.

Usually, the signal is executed on the stack that was in effect before the signal was delivered. An alternate stack may be specified to receive a subset of the signals being caught.

When the signal handler returns, the receiving thread resumes execution at the point it was interrupted unless the signal handler makes other arrangements. If *longjmp()* or *\_longjmp()* is used to leave the signal handler, then the signal mask must be explicitly restored.

This volume of POSIX.1-2008 defines the third argument of a signal handling function when SA\_SIGINFO is set as a **void \*** instead of a **ucontext\_t \***, but without requiring type checking. New applications should explicitly cast the third argument of the signal handling function to **ucontext\_t \***.

The BSD optional four argument signal handling function is not supported by this volume of POSIX.1-2008. The BSD declaration would be:

```
void handler(int sig, int code, struct sigcontext *scp,
             char *addr);
```

where *sig* is the signal number, *code* is additional information on certain signals, *scp* is a pointer to the **sigcontext** structure, and *addr* is additional address information. Much the same information is available in the objects pointed to by the second argument of the signal handler specified when SA\_SIGINFO is set.

Since the *sigaction()* function is allowed but not required to set SA\_NODEFER when the application sets the SA\_RESETHAND flag, applications which depend on the SA\_RESETHAND functionality for the newly installed signal handler must always explicitly set SA\_NODEFER when they set SA\_RESETHAND in order to be portable.

See also the rationale for Realtime Signal Generation and Delivery in XRAT [Section B.2.4.2](#) (on page 3568).

## RATIONALE

Although this volume of POSIX.1-2008 requires that signals that cannot be ignored shall not be added to the signal mask when a signal-catching function is entered, there is no explicit requirement that subsequent calls to *sigaction()* reflect this in the information returned in the *oact* argument. In other words, if SIGKILL is included in the *sa\_mask* field of *act*, it is unspecified whether or not a subsequent call to *sigaction()* returns with SIGKILL included in the *sa\_mask* field of *oact*.

The SA\_NOCLDSTOP flag, when supplied in the *act->sa\_flags* parameter, allows overloading



SIGCHLD with the System V semantics that each SIGCHLD signal indicates a single terminated child. Most conforming applications that catch SIGCHLD are expected to install signal-catching functions that repeatedly call the *waitpid()* function with the WNOHANG flag set, acting on each child for which status is returned, until *waitpid()* returns zero. If stopped children are not of interest, the use of the SA\_NOCLDSTOP flag can prevent the overhead from invoking the signal-catching routine when they stop.

Some historical implementations also define other mechanisms for stopping processes, such as the *ptrace()* function. These implementations usually do not generate a SIGCHLD signal when processes stop due to this mechanism; however, that is beyond the scope of this volume of POSIX.1-2008.

This volume of POSIX.1-2008 requires that calls to *sigaction()* that supply a NULL *act* argument succeed, even in the case of signals that cannot be caught or ignored (that is, SIGKILL or SIGSTOP). The System V *signal()* and BSD *sigvec()* functions return [EINVAL] in these cases and, in this respect, their behavior varies from *sigaction()*.

This volume of POSIX.1-2008 requires that *sigaction()* properly save and restore a signal action set up by the ISO C standard *signal()* function. However, there is no guarantee that the reverse is true, nor could there be given the greater amount of information conveyed by the **sigaction** structure. Because of this, applications should avoid using both functions for the same signal in the same process. Since this cannot always be avoided in case of general-purpose library routines, they should always be implemented with *sigaction()*.

It was intended that the *signal()* function should be implementable as a library routine using *sigaction()*.

The POSIX Realtime Extension extends the *sigaction()* function as specified by the POSIX.1-1990 standard to allow the application to request on a per-signal basis via an additional signal action flag that the extra parameters, including the application-defined signal value, if any, be passed to the signal-catching function.

## FUTURE DIRECTIONS

None.

## SEE ALSO

Section 2.4 (on page 488), *exec*, *\_Exit()*, *kill()*, *\_longjmp()*, *longjmp()*, *pthread\_sigmask()*, *raise()*, *semget()*, *sem\_init()*, *sem\_open()*, *sigaddset()*, *sigaltstack()*, *sigdelset()*, *sigemptyset()*, *sigfillset()*, *sigismember()*, *signal()*, *sigsuspend()*, *wait()*, *waitid()*

XBD <signal.h>

## CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

## Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and POSIX Threads Extension.

In the DESCRIPTION, the second argument to *func* when SA\_SIGINFO is set is no longer permitted to be NULL, and the description of permitted **siginfo\_t** contents is expanded by reference to <signal.h>.

Since the X/OPEN UNIX Extension functionality is now folded into the BASE, the [ENOTSUP] error is deleted.

**Issue 6**

The Open Group Corrigendum U028/7 is applied. In the paragraph entitled “Signal Effects on Other Functions”, a reference to *sigpending()* is added.

In the DESCRIPTION, the text “Signal Generation and Delivery”, “Signal Actions”, and “Signal Effects on Other Functions” are moved to a separate section of this volume of POSIX.1-2008.

Text describing functionality from the Realtime Signals Extension option is marked.

The following changes are made for alignment with the ISO POSIX-1: 1996 standard:

- The [ENOTSUP] error condition is added.

The normative text is updated to avoid use of the term “must” for application requirements.

The **restrict** keyword is added to the *sigaction()* prototype for alignment with the ISO/IEC 9899: 1999 standard.

References to the *wait3()* function are removed.

The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an extension over the ISO C standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/57 is applied, changing text in the table describing the **sigaction** structure.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/127 is applied, removing text from the DESCRIPTION duplicated later in the same section.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/128 is applied, updating the DESCRIPTION and APPLICATION USAGE sections. Changes are made to refer to the thread rather than the process.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/129 is applied, adding the example to the EXAMPLES section.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #004 is applied.

Austin Group Interpretations 1003.1-2001 #065 and #084 are applied, clarifying the role of the SA\_NODEFER flag with respect to the signal mask, and clarifying the SA\_RESTART flag for interrupted functions which use timeouts.

Austin Group Interpretation 1003.1-2001 #156 is applied.

SD5-XSH-ERN-167 is applied, updating the APPLICATION USAGE section.

SD5-XSH-ERN-172 is applied, updating the DESCRIPTION to make optional the requirement that when the SA\_RESETHAND flag is set, *sigaction()* shall behave as if the SA\_NODEFER flag were also set.

Functionality relating to the Realtime Signals Extension option is moved to the Base.

The description of the *si\_code* member is replaced with a reference to [Section 2.4.3](#) (on page 490).

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0578 [66] and XSH/TC1-2008/0579 [140] are applied.

62795 **NAME**

62796 sigaddset — add a signal to a signal set

62797 **SYNOPSIS**

```
62798 CX #include <signal.h>
62799 int sigaddset(sigset_t *set, int signo);
```

62800 **DESCRIPTION**

62801 The *sigaddset()* function adds the individual signal specified by the *signo* to the signal set pointed  
62802 to by *set*.

62803 Applications shall call either *sigemptyset()* or *sigfillset()* at least once for each object of type  
62804 **sigset\_t** prior to any other use of that object. If such an object is not initialized in this way, but is  
62805 nonetheless supplied as an argument to any of *pthread\_sigmask()*, *sigaction()*, *sigaddset()*,  
62806 *sigdelset()*, *sigismember()*, *sigpending()*, *sigprocmask()*, *sigsuspend()*, *sigtimedwait()*, *sigwait()*, or  
62807 *sigwaitinfo()*, the results are undefined.

62808 **RETURN VALUE**

62809 Upon successful completion, *sigaddset()* shall return 0; otherwise, it shall return -1 and set *errno*  
62810 to indicate the error.

62811 **ERRORS**62812 The *sigaddset()* function may fail if:

62813 [EINVAL] The value of the *signo* argument is an invalid or unsupported signal number.

62814 **EXAMPLES**

62815 None.

62816 **APPLICATION USAGE**

62817 None.

62818 **RATIONALE**

62819 None.

62820 **FUTURE DIRECTIONS**

62821 None.

62822 **SEE ALSO**

62823 [Section 2.4](#) (on page 488), *pthread\_sigmask()*, *sigaction()*, *sigdelset()*, *sigemptyset()*, *sigfillset()*,  
62824 *sigismember()*, *sigpending()*, *sigsuspend()*

62825 XBD [<signal.h>](#)62826 **CHANGE HISTORY**

62827 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

62828 **Issue 5**

62829 The last paragraph of the DESCRIPTION was included as an APPLICATION USAGE note in  
62830 previous issues.

62831 **Issue 6**

62832 The normative text is updated to avoid use of the term “must” for application requirements.

62833 The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an  
62834 extension over the ISO C standard.



**NAME**

sigaltstack — set and get signal alternate stack context

**SYNOPSIS**

```
#include <signal.h>

int sigaltstack(const stack_t *restrict ss, stack_t *restrict oss);
```

**DESCRIPTION**

The *sigaltstack()* function allows a process to define and examine the state of an alternate stack for signal handlers for the current thread. Signals that have been explicitly declared to execute on the alternate stack shall be delivered on the alternate stack.

If *ss* is not a null pointer, it points to a **stack\_t** structure that specifies the alternate signal stack that shall take effect upon return from *sigaltstack()*. The *ss\_flags* member specifies the new stack state. If it is set to *SS\_DISABLE*, the stack is disabled and *ss\_sp* and *ss\_size* are ignored. Otherwise, the stack shall be enabled, and the *ss\_sp* and *ss\_size* members specify the new address and size of the stack.

The range of addresses starting at *ss\_sp* up to but not including *ss\_sp+ss\_size* is available to the implementation for use as the stack. This function makes no assumptions regarding which end is the stack base and in which direction the stack grows as items are pushed.

If *oss* is not a null pointer, upon successful completion it shall point to a **stack\_t** structure that specifies the alternate signal stack that was in effect prior to the call to *sigaltstack()*. The *ss\_sp* and *ss\_size* members specify the address and size of that stack. The *ss\_flags* member specifies the stack's state, and may contain one of the following values:

**SS\_ONSTACK**    The process is currently executing on the alternate signal stack. Attempts to modify the alternate signal stack while the process is executing on it fail. This flag shall not be modified by processes.

**SS\_DISABLE**    The alternate signal stack is currently disabled.

The value *SIGSTKSZ* is a system default specifying the number of bytes that would be used to cover the usual case when manually allocating an alternate stack area. The value *MINSIGSTKSZ* is defined to be the minimum stack size for a signal handler. In computing an alternate stack size, a program should add that amount to its stack requirements to allow for the system implementation overhead. The constants *SS\_ONSTACK*, *SS\_DISABLE*, *SIGSTKSZ*, and *MINSIGSTKSZ* are defined in **<signal.h>**.

After a successful call to one of the *exec* functions, there are no alternate signal stacks in the new process image.

In some implementations, a signal (whether or not indicated to execute on the alternate stack) shall always execute on the alternate stack if it is delivered while another signal is being caught using the alternate stack.

Use of this function by library threads that are not bound to kernel-scheduled entities results in undefined behavior.

**RETURN VALUE**

Upon successful completion, *sigaltstack()* shall return 0; otherwise, it shall return -1 and set *errno* to indicate the error.

**ERRORS**

The *sigaltstack()* function shall fail if:

[EINVAL] The *ss* argument is not a null pointer, and the *ss\_flags* member pointed to by *ss* contains flags other than *SS\_DISABLE*.

[ENOMEM] The size of the alternate stack area is less than *MINSIGSTKSZ*.

[EPERM] An attempt was made to modify an active stack.

**EXAMPLES****Allocating Memory for an Alternate Stack**

The following example illustrates a method for allocating memory for an alternate stack.

```
#include <signal.h>
...
if ((sigstk.ss_sp = malloc(SIGSTKSZ)) == NULL)
    /* Error return. */
sigstk.ss_size = SIGSTKSZ;
sigstk.ss_flags = 0;
if (sigaltstack(&sigstk, (stack_t *)0) < 0)
    perror("sigaltstack");
```

**APPLICATION USAGE**

On some implementations, stack space is automatically extended as needed. On those implementations, automatic extension is typically not available for an alternate stack. If the stack overflows, the behavior is undefined.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 2.4](#) (on page 488), *exec*, *sigaction()*, *sigsetjmp()*

XBD [<signal.h>](#)

**CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

The last sentence of the DESCRIPTION was included as an APPLICATION USAGE note in previous issues.

**Issue 6**

The normative text is updated to avoid use of the term “must” for application requirements.

The **restrict** keyword is added to the *sigaltstack()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/58 is applied, updating the first sentence to include “for the current thread”.

62916 **NAME**

62917 sigdelset — delete a signal from a signal set

62918 **SYNOPSIS**

```
62919 CX    #include <signal.h>
62920    int sigdelset(sigset_t *set, int signo);
```

62921 **DESCRIPTION**

62922 The *sigdelset()* function deletes the individual signal specified by *signo* from the signal set  
 62923 pointed to by *set*.

62924 Applications should call either *sigemptyset()* or *sigfillset()* at least once for each object of type  
 62925 **sigset\_t** prior to any other use of that object. If such an object is not initialized in this way, but is  
 62926 nonetheless supplied as an argument to any of *pthread\_sigmask()*, *sigaction()*, *sigaddset()*,  
 62927 *sigdelset()*, *sigismember()*, *sigpending()*, *sigprocmask()*, *sigsuspend()*, *sigtimedwait()*, *sigwait()*, or  
 62928 *sigwaitinfo()*, the results are undefined.

62929 **RETURN VALUE**

62930 Upon successful completion, *sigdelset()* shall return 0; otherwise, it shall return -1 and set *errno*  
 62931 to indicate the error.

62932 **ERRORS**

62933 The *sigdelset()* function may fail if:

|       |          |                                                                                             |
|-------|----------|---------------------------------------------------------------------------------------------|
| 62934 | [EINVAL] | The <i>signo</i> argument is not a valid signal number, or is an unsupported signal number. |
| 62935 |          |                                                                                             |

62936 **EXAMPLES**

62937 None.

62938 **APPLICATION USAGE**

62939 None.

62940 **RATIONALE**

62941 None.

62942 **FUTURE DIRECTIONS**

62943 None.

62944 **SEE ALSO**

62945 Section 2.4 (on page 488), *pthread\_sigmask()*, *sigaction()*, *sigaddset()*, *sigemptyset()*, *sigfillset()*,  
 62946 *sigismember()*, *sigpending()*, *sigsuspend()*

62947 XBD **<signal.h>**

62948 **CHANGE HISTORY**

62949 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

62950 **Issue 5**

62951 The last paragraph of the DESCRIPTION was included as an APPLICATION USAGE note in  
 62952 previous issues.

62953 **Issue 6**

62954 The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an  
 62955 extension over the ISO C standard.

**NAME**

sigemptyset — initialize and empty a signal set

**SYNOPSIS**

```
CX      #include <signal.h>
        int sigemptyset(sigset_t *set);
```

**DESCRIPTION**

The *sigemptyset()* function initializes the signal set pointed to by *set*, such that all signals defined in POSIX.1-2008 are excluded.

**RETURN VALUE**

Upon successful completion, *sigemptyset()* shall return 0; otherwise, it shall return -1 and set *errno* to indicate the error.

**ERRORS**

No errors are defined.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

The implementation of the *sigemptyset()* (or *sigfillset()*) function could quite trivially clear (or set) all the bits in the signal set. Alternatively, it would be reasonable to initialize part of the structure, such as a version field, to permit binary-compatibility between releases where the size of the set varies. For such reasons, either *sigemptyset()* or *sigfillset()* must be called prior to any other use of the signal set, even if such use is read-only (for example, as an argument to *sigpending()*). This function is not intended for dynamic allocation.

The *sigfillset()* and *sigemptyset()* functions require that the resulting signal set include (or exclude) all the signals defined in this volume of POSIX.1-2008. Although it is outside the scope of this volume of POSIX.1-2008 to place this requirement on signals that are implemented as extensions, it is recommended that implementation-defined signals also be affected by these functions. However, there may be a good reason for a particular signal not to be affected. For example, blocking or ignoring an implementation-defined signal may have undesirable side-effects, whereas the default action for that signal is harmless. In such a case, it would be preferable for such a signal to be excluded from the signal set returned by *sigfillset()*.

In early proposals there was no distinction between invalid and unsupported signals (the names of optional signals that were not supported by an implementation were not defined by that implementation). The [EINVAL] error was thus specified as a required error for invalid signals. With that distinction, it is not necessary to require implementations of these functions to determine whether an optional signal is actually supported, as that could have a significant performance impact for little value. The error could have been required for invalid signals and optional for unsupported signals, but this seemed unnecessarily complex. Thus, the error is optional in both cases.

**FUTURE DIRECTIONS**

None.

62998 **SEE ALSO**

62999       Section 2.4 (on page 488), *pthread\_sigmask()*, *sigaction()*, *sigaddset()*, *sigdelset()*, *sigfillset()*,  
63000       *sigismember()*, *sigpending()*, *sigsuspend()*

63001       XBD <signal.h>

63002 **CHANGE HISTORY**

63003       First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

63004 **Issue 6**

63005       The SYNOPSIS is marked CX since the presence of this function in the <signal.h> header is an  
63006       extension over the ISO C standard.

63007 **NAME**

63008 sigfillset — initialize and fill a signal set

63009 **SYNOPSIS**

```
63010 CX    #include <signal.h>
63011        int sigfillset(sigset_t *set);
```

63012 **DESCRIPTION**

63013 The *sigfillset()* function shall initialize the signal set pointed to by *set*, such that all signals  
63014 defined in this volume of POSIX.1-2008 are included.

63015 **RETURN VALUE**

63016 Upon successful completion, *sigfillset()* shall return 0; otherwise, it shall return -1 and set *errno*  
63017 to indicate the error.

63018 **ERRORS**

63019 No errors are defined.

63020 **EXAMPLES**

63021 None.

63022 **APPLICATION USAGE**

63023 None.

63024 **RATIONALE**63025 Refer to *sigemptyset()* (on page 1952).63026 **FUTURE DIRECTIONS**

63027 None.

63028 **SEE ALSO**

63029 Section 2.4 (on page 488), *pthread\_sigmask()*, *sigaction()*, *sigaddset()*, *sigdelset()*, *sigemptyset()*,  
63030 *sigismember()*, *sigpending()*, *sigsuspend()*

63031 XBD **<signal.h>**63032 **CHANGE HISTORY**

63033 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

63034 **Issue 6**

63035 The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an  
63036 extension over the ISO C standard.

63037 **NAME**

63038 sighold, sigignore, sigpause, sigrelse, sigset — signal management

63039 **SYNOPSIS**

```

63040 OB XSI #include <signal.h>
63041
63041 int sighold(int sig);
63042 int sigignore(int sig);
63043 int sigpause(int sig);
63044 int sigrelse(int sig);
63045 void (*sigset(int sig, void (*disp)(int)))(int);

```

63046 **DESCRIPTION**

63047 Use of any of these functions is unspecified in a multi-threaded process.

63048 The *sighold()*, *sigignore()*, *sigpause()*, *sigrelse()*, and *sigset()* functions provide simplified signal management.

63050 The *sigset()* function shall modify signal dispositions. The *sig* argument specifies the signal, which may be any signal except SIGKILL and SIGSTOP. The *disp* argument specifies the signal's disposition, which may be SIG\_DFL, SIG\_IGN, or the address of a signal handler. If *sigset()* is used, and *disp* is the address of a signal handler, the system shall add *sig* to the signal mask of the calling process before executing the signal handler; when the signal handler returns, the system shall restore the signal mask of the calling process to its state prior to the delivery of the signal. In addition, if *sigset()* is used, and *disp* is equal to SIG\_HOLD, *sig* shall be added to the signal mask of the calling process and *sig*'s disposition shall remain unchanged. If *sigset()* is used, and *disp* is not equal to SIG\_HOLD, *sig* shall be removed from the signal mask of the calling process.

63060 The *sighold()* function shall add *sig* to the signal mask of the calling process.63061 The *sigrelse()* function shall remove *sig* from the signal mask of the calling process.63062 The *sigignore()* function shall set the disposition of *sig* to SIG\_IGN.

63063 The *sigpause()* function shall remove *sig* from the signal mask of the calling process and suspend the calling process until a signal is received. The *sigpause()* function shall restore the signal mask of the process to its original state before returning.

63066 If the action for the SIGCHLD signal is set to SIG\_IGN, child processes of the calling processes shall not be transformed into zombie processes when they terminate. If the calling process subsequently waits for its children, and the process has no unwaited-for children that were transformed into zombie processes, it shall block until all of its children terminate, and *wait()*, *waitid()*, and *waitpid()* shall fail and set *errno* to [ECHILD].

63071 **RETURN VALUE**

63072 Upon successful completion, *sigset()* shall return SIG\_HOLD if the signal had been blocked and the signal's previous disposition if it had not been blocked. Otherwise, SIG\_ERR shall be returned and *errno* set to indicate the error.

63075 The *sigpause()* function shall suspend execution of the thread until a signal is received, whereupon it shall return -1 and set *errno* to [EINTR].

63077 For all other functions, upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to indicate the error.

**ERRORS**

These functions shall fail if:

[EINVAL] The *sig* argument is an illegal signal number.

The *sigset*() and *sigignore*() functions shall fail if:

[EINVAL] An attempt is made to catch a signal that cannot be caught, or to ignore a signal that cannot be ignored.

**EXAMPLES**

None.

**APPLICATION USAGE**

The *sigaction*() function provides a more comprehensive and reliable mechanism for controlling signals; new applications should use the *sigaction*() function instead of the obsolescent *sigset*() function.

The *sighold*() function, in conjunction with *sigrelse*() or *sigpause*(), may be used to establish critical regions of code that require the delivery of a signal to be temporarily deferred. For broader portability, the *pthread\_sigmask*() or *sigprocmask*() functions should be used instead of the obsolescent *sighold*() and *sigrelse*() functions.

For broader portability, the *sigsuspend*() function should be used instead of the obsolescent *sigpause*() function.

**RATIONALE**

Each of these historic functions has a direct analog in the other functions which are required to be per-thread and thread-safe (aside from *sigprocmask*(), which is replaced by *pthread\_sigmask*()). The *sigset*() function can be implemented as a simple wrapper for *sigaction*(). The *sighold*() function is equivalent to *sigprocmask*() or *pthread\_sigmask*() with SIG\_BLOCK set. The *sigignore*() function is equivalent to *sigaction*() with SIG\_IGN set. The *sigpause*() function is equivalent to *sigsuspend*(). The *sigrelse*() function is equivalent to *sigprocmask*() or *pthread\_sigmask*() with SIG\_UNBLOCK set.

**FUTURE DIRECTIONS**

These functions may be removed in a future version.

**SEE ALSO**

Section 2.4 (on page 488), *exec*, *pause*(), *pthread\_sigmask*(), *sigaction*(), *signal*(), *sigsuspend*(), *wait*(), *waitid*()

XBD <signal.h>

**CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

The DESCRIPTION is updated to indicate that the *sigpause*() function restores the signal mask of the process to its original state before returning.

The RETURN VALUE section is updated to indicate that the *sigpause*() function suspends execution of the process until a signal is received, whereupon it returns -1 and sets *errno* to [EINTR].



63120 **Issue 6**

63121 The normative text is updated to avoid use of the term “must” for application requirements.

63122 References to the *wait3()* function are removed.

63123 The XSI functions are split out into their own reference page.

63124 **Issue 7**

63125 SD5-XSH-ERN-113 and SD5-XSH-ERN-42 are applied, marking these functions obsolescent and  
63126 updating the APPLICATION USAGE and RATIONALE sections.

63127 **NAME**

63128 siginterrupt — allow signals to interrupt functions

63129 **SYNOPSIS**

```
63130 OB XSI #include <signal.h>
63131 int siginterrupt(int sig, int flag);
```

63132 **DESCRIPTION**

63133 The *siginterrupt()* function shall change the restart behavior when a function is interrupted by  
63134 the specified signal. The function *siginterrupt(sig, flag)* has an effect as if implemented as:

```
63135 int siginterrupt(int sig, int flag) {
63136     int ret;
63137     struct sigaction act;
63138     (void) sigaction(sig, NULL, &act);
63139     if (flag)
63140         act.sa_flags &= ~SA_RESTART;
63141     else
63142         act.sa_flags |= SA_RESTART;
63143     ret = sigaction(sig, &act, NULL);
63144     return ret;
63145 }
```

63146 **RETURN VALUE**

63147 Upon successful completion, *siginterrupt()* shall return 0; otherwise, -1 shall be returned and  
63148 *errno* set to indicate the error.

63149 **ERRORS**

63150 The *siginterrupt()* function shall fail if:

63151 [EINVAL] The *sig* argument is not a valid signal number.

63152 **EXAMPLES**

63153 None.

63154 **APPLICATION USAGE**

63155 The *siginterrupt()* function supports programs written to historical system interfaces.  
63156 Applications should use the *sigaction()* with the SA\_RESTART flag instead of the obsolescent  
63157 *siginterrupt()* function.

63158 **RATIONALE**

63159 None.

63160 **FUTURE DIRECTIONS**

63161 None.

63162 **SEE ALSO**

63163 [Section 2.4](#) (on page 488), *sigaction()*

63164 XBD [<signal.h>](#)

63165 **CHANGE HISTORY**

63166 First released in Issue 4, Version 2.

63167 **Issue 5**

63168 Moved from X/OPEN UNIX extension to BASE.

63169 **Issue 6**63170 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/59 is applied, correcting the declaration in  
63171 the sample implementation given in the DESCRIPTION.63172 **Issue 7**63173 The *siginterrupt()* function is marked obsolescent.

63174 **NAME**

63175 sigismember — test for a signal in a signal set

63176 **SYNOPSIS**

```
63177 CX #include <signal.h>
63178 int sigismember(const sigset_t *set, int signo);
```

63179 **DESCRIPTION**

63180 The *sigismember()* function shall test whether the signal specified by *signo* is a member of the set  
63181 pointed to by *set*.

63182 Applications should call either *sigemptyset()* or *sigfillset()* at least once for each object of type  
63183 **sigset\_t** prior to any other use of that object. If such an object is not initialized in this way, but is  
63184 nonetheless supplied as an argument to any of *pthread\_sigmask()*, *sigaction()*, *sigaddset()*,  
63185 *sigdelset()*, *sigismember()*, *sigpending()*, *sigprocmask()*, *sigsuspend()*, *sigtimedwait()*, *sigwait()*, or  
63186 *sigwaitinfo()*, the results are undefined.

63187 **RETURN VALUE**

63188 Upon successful completion, *sigismember()* shall return 1 if the specified signal is a member of  
63189 the specified set, or 0 if it is not. Otherwise, it shall return -1 and set *errno* to indicate the error.

63190 **ERRORS**

63191 The *sigismember()* function may fail if:

63192 [EINVAL] The *signo* argument is not a valid signal number, or is an unsupported signal  
63193 number.

63194 **EXAMPLES**

63195 None.

63196 **APPLICATION USAGE**

63197 None.

63198 **RATIONALE**

63199 None.

63200 **FUTURE DIRECTIONS**

63201 None.

63202 **SEE ALSO**

63203 Section 2.4 (on page 488), *pthread\_sigmask()*, *sigaction()*, *sigaddset()*, *sigdelset()*, *sigfillset()*,  
63204 *sigemptyset()*, *sigpending()*, *sigsuspend()*

63205 XBD <signal.h>

63206 **CHANGE HISTORY**

63207 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

63208 **Issue 5**

63209 The last paragraph of the DESCRIPTION was included as an APPLICATION USAGE note in  
63210 previous issues.

63211 **Issue 6**

63212 The SYNOPSIS is marked CX since the presence of this function in the <signal.h> header is an  
63213 extension over the ISO C standard.

63214 **NAME**

63215 siglongjmp — non-local goto with signal handling

63216 **SYNOPSIS**

```
63217 CX    #include <setjmp.h>
63218    void siglongjmp(sigjmp_buf env, int val);
```

63219 **DESCRIPTION**63220 The *siglongjmp()* function shall be equivalent to the *longjmp()* function, except as follows:

- 63221 • References to *setjmp()* shall be equivalent to *sigsetjmp()*.
- 63222 • The *siglongjmp()* function shall restore the saved signal mask if and only if the *env*
- 63223 argument was initialized by a call to *sigsetjmp()* with a non-zero *savemask* argument.

63224 **RETURN VALUE**

63225 After *siglongjmp()* is completed, program execution shall continue as if the corresponding

63226 invocation of *sigsetjmp()* had just returned the value specified by *val*. The *siglongjmp()* function

63227 shall not cause *sigsetjmp()* to return 0; if *val* is 0, *sigsetjmp()* shall return the value 1.

63228 **ERRORS**

63229 No errors are defined.

63230 **EXAMPLES**

63231 None.

63232 **APPLICATION USAGE**

63233 The distinction between *setjmp()* or *longjmp()* and *sigsetjmp()* or *siglongjmp()* is only significant

63234 for programs which use *sigaction()*, *sigprocmask()*, or *sigsuspend()*.

63235 **RATIONALE**

63236 None.

63237 **FUTURE DIRECTIONS**

63238 None.

63239 **SEE ALSO**63240 *longjmp()*, *pthread\_sigmask()*, *setjmp()*, *sigsetjmp()*, *sigsuspend()*63241 XBD **<setjmp.h>**63242 **CHANGE HISTORY**

63243 First released in Issue 3. Included for alignment with the ISO POSIX-1 standard.

63244 **Issue 5**

63245 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

63246 **Issue 6**63247 The DESCRIPTION is rewritten in terms of *longjmp()*.

63248 The SYNOPSIS is marked CX since the presence of this function in the **<setjmp.h>** header is an

63249 extension over the ISO C standard.

## 63250 NAME

63251 signal — signal management

## 63252 SYNOPSIS

63253 #include &lt;signal.h&gt;

63254 void (\*signal(int sig, void (\*func)(int)))(int);

## 63255 DESCRIPTION

63256 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 63257 conflict between the requirements described here and the ISO C standard is unintentional. This  
 63258 volume of POSIX.1-2008 defers to the ISO C standard.

63259 The *signal()* function chooses one of three ways in which receipt of the signal number *sig* is to be  
 63260 subsequently handled. If the value of *func* is SIG\_DFL, default handling for that signal shall  
 63261 occur. If the value of *func* is SIG\_IGN, the signal shall be ignored. Otherwise, the application  
 63262 shall ensure that *func* points to a function to be called when that signal occurs. An invocation of  
 63263 such a function because of a signal, or (recursively) of any further functions called by that  
 63264 invocation (other than functions in the standard library), is called a “signal handler”.

63265 When a signal occurs, and *func* points to a function, it is implementation-defined whether the  
 63266 equivalent of a:

63267 signal(*sig*, SIG\_DFL);

63268 is executed or the implementation prevents some implementation-defined set of signals (at least  
 63269 including *sig*) from occurring until the current signal handling has completed. (If the value of *sig*  
 63270 is SIGILL, the implementation may alternatively define that no action is taken.) Next the  
 63271 equivalent of:

63272 (\*func)(*sig*);

63273 is executed. If and when the function returns, if the value of *sig* was SIGFPE, SIGILL, or  
 63274 SIGSEGV or any other implementation-defined value corresponding to a computational  
 63275 exception, the behavior is undefined. Otherwise, the program shall resume execution at the  
 63276 point it was interrupted. The ISO C standard places a restriction on applications relating to the  
 63277 CX use of *raise()* from signal handlers. This restriction does not apply to POSIX applications, as  
 63278 POSIX.1-2008 requires *raise()* to be async-signal-safe (see Section 2.4.3, on page 490).

63279 CX If the process is multi-threaded, or if the process is single-threaded and a signal handler is  
 63280 executed other than as the result of:

63281 CX • The process calling *abort()*, *raise()*, *kill()*, *pthread\_kill()*, or *sigqueue()* to generate a signal  
 63282 that is not blocked

63283 CX • A pending signal being unblocked and being delivered before the call that unblocked it  
 63284 returns

63285 CX the behavior is undefined if the signal handler refers to any object other than *errno* with static  
 63286 storage duration other than by assigning a value to an object declared as *volatile sig\_atomic\_t*,  
 63287 CX or if the signal handler calls any function defined in this standard other than one of the  
 63288 functions listed in Section 2.4 (on page 488).

63289 At program start-up, the equivalent of:

63290 signal(*sig*, SIG\_IGN);

63291 is executed for some signals, and the equivalent of:

63292 signal(*sig*, SIG\_DFL);

63293 CX is executed for all other signals (see *exec*).

63294 The *signal()* function shall not change the setting of *errno* if successful.

#### 63295 RETURN VALUE

63296 If the request can be honored, *signal()* shall return the value of *func* for the most recent call to  
 63297 *signal()* for the specified signal *sig*. Otherwise, *SIG\_ERR* shall be returned and a positive value  
 63298 shall be stored in *errno*.

#### 63299 ERRORS

63300 The *signal()* function shall fail if:

63301 CX [EINVAL] The *sig* argument is not a valid signal number or an attempt is made to catch a  
 63302 signal that cannot be caught or ignore a signal that cannot be ignored.

63303 The *signal()* function may fail if:

63304 CX [EINVAL] An attempt was made to set the action to *SIG\_DFL* for a signal that cannot be  
 63305 caught or ignored (or both).

#### 63306 EXAMPLES

63307 None.

#### 63308 APPLICATION USAGE

63309 The *sigaction()* function provides a more comprehensive and reliable mechanism for controlling  
 63310 signals; new applications should use *sigaction()* rather than *signal()*.

#### 63311 RATIONALE

63312 None.

#### 63313 FUTURE DIRECTIONS

63314 None.

#### 63315 SEE ALSO

63316 Section 2.4 (on page 488), *exec*, *pause()*, *raise()*, *sigaction()*, *sigsuspend()*, *waitid()*

63317 XBD <signal.h>

#### 63318 CHANGE HISTORY

63319 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 63320 Issue 5

63321 Moved from X/OPEN UNIX extension to BASE.

63322 The DESCRIPTION is updated to indicate that the *sigpause()* function restores the signal mask of  
 63323 the process to its original state before returning.

63324 The RETURN VALUE section is updated to indicate that the *sigpause()* function suspends  
 63325 execution of the process until a signal is received, whereupon it returns *-1* and sets *errno* to  
 63326 [EINTR].

#### 63327 Issue 6

63328 Extensions beyond the ISO C standard are marked.

63329 The normative text is updated to avoid use of the term “must” for application requirements.

63330 The DESCRIPTION is updated for alignment with the ISO/IEC 9899:1999 standard.

63331 References to the *wait3()* function are removed.

63332 The *sighold()*, *sigignore()*, *sigelse()*, and *sigset()* functions are split out onto their own reference  
 63333 page.

63334 **Issue 7**

63335 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0580 [275], XSH/TC1-2008/0581 [66],  
63336 and XSH/TC1-2008/0582 [105] are applied.



63337 **NAME**

63338 signbit — test sign

63339 **SYNOPSIS**

63340 #include &lt;math.h&gt;

63341 int signbit(real-floating x);

63342 **DESCRIPTION**

63343 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
63344 conflict between the requirements described here and the ISO C standard is unintentional. This  
63345 volume of POSIX.1-2008 defers to the ISO C standard.

63346 The *signbit()* macro shall determine whether the sign of its argument value is negative. NaNs,  
63347 zeros, and infinities have a sign bit.

63348 **RETURN VALUE**

63349 The *signbit()* macro shall return a non-zero value if and only if the sign of its argument value is  
63350 negative.

63351 **ERRORS**

63352 No errors are defined.

63353 **EXAMPLES**

63354 None.

63355 **APPLICATION USAGE**

63356 None.

63357 **RATIONALE**

63358 None.

63359 **FUTURE DIRECTIONS**

63360 None.

63361 **SEE ALSO**63362 *fpclassify(), isfinite(), isinf(), isnan(), isnormal()*

63363 XBD &lt;math.h&gt;

63364 **CHANGE HISTORY**

63365 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

63366 **NAME**

63367 signgam — log gamma function

63368 **SYNOPSIS**

```
63369 XSI      #include <math.h>
63370          extern int signgam;
```

63371 **DESCRIPTION**63372 Refer to *lgamma()*.

63373 **NAME**

63374 sigpause — remove a signal from the signal mask and suspend the thread

63375 **SYNOPSIS**

```
63376 OB XSI #include <signal.h>  
63377 int sigpause(int sig);
```

63378 **DESCRIPTION**63379 Refer to *sighold()*.

63380 **NAME**

63381 sigpending — examine pending signals

63382 **SYNOPSIS**

```
63383 CX    #include <signal.h>
63384      int sigpending(sigset_t *set);
```

63385 **DESCRIPTION**

63386 The *sigpending()* function shall store, in the location referenced by the *set* argument, the set of  
63387 signals that are blocked from delivery to the calling thread and that are pending on the process  
63388 or the calling thread.

63389 **RETURN VALUE**

63390 Upon successful completion, *sigpending()* shall return 0; otherwise, -1 shall be returned and  
63391 *errno* set to indicate the error.

63392 **ERRORS**

63393 No errors are defined.

63394 **EXAMPLES**

63395 None.

63396 **APPLICATION USAGE**

63397 None.

63398 **RATIONALE**

63399 None.

63400 **FUTURE DIRECTIONS**

63401 None.

63402 **SEE ALSO**63403 *exec*, *pthread\_sigmask()*, *sigaddset()*, *sigdelset()*, *sigemptyset()*, *sigfillset()*, *sigismember()*63404 XBD [\*\*<signal.h>\*\*](#)63405 **CHANGE HISTORY**

63406 First released in Issue 3.

63407 **Issue 5**

63408 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

63409 **Issue 6**

63410 The SYNOPSIS is marked CX since the presence of this function in the **<signal.h>** header is an  
63411 extension over the ISO C standard.

63412 **NAME**

63413 sigprocmask — examine and change blocked signals

63414 **SYNOPSIS**

```
63415 CX    #include <signal.h>
63416        int sigprocmask(int how, const sigset_t *restrict set,
63417                        sigset_t *restrict oset);
```

63418 **DESCRIPTION**63419 Refer to *pthread\_sigmask()*.

63420 **NAME**

63421 sigqueue — queue a signal to a process

63422 **SYNOPSIS**

```
63423 CX #include <signal.h>
63424 int sigqueue(pid_t pid, int signo, union sigval value);
```

63425 **DESCRIPTION**

63426 The *sigqueue()* function shall cause the signal specified by *signo* to be sent with the value  
63427 specified by *value* to the process specified by *pid*. If *signo* is zero (the null signal), error checking  
63428 is performed but no signal is actually sent. The null signal can be used to check the validity of  
63429 *pid*.

63430 The conditions required for a process to have permission to queue a signal to another process are  
63431 the same as for the *kill()* function.

63432 The *sigqueue()* function shall return immediately. If SA\_SIGINFO is set for *signo* and if the  
63433 resources were available to queue the signal, the signal shall be queued and sent to the receiving  
63434 process. If SA\_SIGINFO is not set for *signo*, then *signo* shall be sent at least once to the receiving  
63435 process; it is unspecified whether *value* shall be sent to the receiving process as a result of this  
63436 call.

63437 If the value of *pid* causes *signo* to be generated for the sending process, and if *signo* is not blocked  
63438 for the calling thread and if no other thread has *signo* unblocked or is waiting in a *sigwait()*  
63439 function for *signo*, either *signo* or at least the pending, unblocked signal shall be delivered to the  
63440 calling thread before the *sigqueue()* function returns. Should any multiple pending signals in the  
63441 range SIGRTMIN to SIGRTMAX be selected for delivery, it shall be the lowest numbered one.  
63442 The selection order between realtime and non-realtime signals, or between multiple pending  
63443 non-realtime signals, is unspecified.

63444 **RETURN VALUE**

63445 Upon successful completion, the specified signal shall have been queued, and the *sigqueue()*  
63446 function shall return a value of zero. Otherwise, the function shall return a value of -1 and set  
63447 *errno* to indicate the error.

63448 **ERRORS**63449 The *sigqueue()* function shall fail if:

- |       |          |                                                                                    |
|-------|----------|------------------------------------------------------------------------------------|
| 63450 | [EAGAIN] | No resources are available to queue the signal. The process has already            |
| 63451 |          | queued {SIGQUEUE_MAX} signals that are still pending at the receiver(s), or        |
| 63452 |          | a system-wide resource limit has been exceeded.                                    |
| 63453 | [EINVAL] | The value of the <i>signo</i> argument is an invalid or unsupported signal number. |
| 63454 | [EPERM]  | The process does not have appropriate privileges to send the signal to the         |
| 63455 |          | receiving process.                                                                 |
| 63456 | [ESRCH]  | The process <i>pid</i> does not exist.                                             |

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

The *sigqueue()* function allows an application to queue a realtime signal to itself or to another process, specifying the application-defined value. This is common practice in realtime applications on existing realtime systems. It was felt that specifying another function in the *sig...* name space already carved out for signals was preferable to extending the interface to *kill()*.

Such a function became necessary when the put/get event function of the message queues was removed. It should be noted that the *sigqueue()* function implies reduced performance in a security-conscious implementation as the access permissions between the sender and receiver have to be checked on each send when the *pid* is resolved into a target process. Such access checks were necessary only at message queue open in the previous interface.

The standard developers required that *sigqueue()* have the same semantics with respect to the null signal as *kill()*, and that the same permission checking be used. But because of the difficulty of implementing the “broadcast” semantic of *kill()* (for example, to process groups) and the interaction with resource allocation, this semantic was not adopted. The *sigqueue()* function queues a signal to a single process specified by the *pid* argument.

The *sigqueue()* function can fail if the system has insufficient resources to queue the signal. An explicit limit on the number of queued signals that a process could send was introduced. While the limit is “per-sender”, this volume of POSIX.1-2008 does not specify that the resources be part of the state of the sender. This would require either that the sender be maintained after exit until all signals that it had sent to other processes were handled or that all such signals that had not yet been acted upon be removed from the queue(s) of the receivers. This volume of POSIX.1-2008 does not preclude this behavior, but an implementation that allocated queuing resources from a system-wide pool (with per-sender limits) and that leaves queued signals pending after the sender exits is also permitted.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 2.8.1](#) (on page 503)

XBD [<signal.h>](#)

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

**Issue 6**

The *sigqueue()* function is marked as part of the Realtime Signals Extension option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Realtime Signals Extension option.

**Issue 7**

The *sigqueue()* function is moved from the Realtime Signals Extension option to the Base.

63500 **NAME**

63501 sigrelse, sigset — signal management

63502 **SYNOPSIS**

```
63503 OB XSI #include <signal.h>
63504 int sigrelse(int sig);
63505 void (*sigset(int sig, void (*disp)(int)))(int);
```

63506 **DESCRIPTION**63507 Refer to *sighold()*.



63508 **NAME**

63509 sigsetjmp — set jump point for a non-local goto

63510 **SYNOPSIS**

```
63511 CX      #include <setjmp.h>
63512      int sigsetjmp(sigjmp_buf env, int savemask);
```

63513 **DESCRIPTION**63514 The *sigsetjmp()* function shall be equivalent to the *setjmp()* function, except as follows:

- 63515 • References to *setjmp()* are equivalent to *sigsetjmp()*.
- 63516 • References to *longjmp()* are equivalent to *siglongjmp()*.
- 63517 • If the value of the *savemask* argument is not 0, *sigsetjmp()* shall also save the current signal mask of the calling thread as part of the calling environment.

63519 **RETURN VALUE**

63520 If the return is from a successful direct invocation, *sigsetjmp()* shall return 0. If the return is from  
 63521 a call to *siglongjmp()*, *sigsetjmp()* shall return a non-zero value.

63522 **ERRORS**

63523 No errors are defined.

63524 **EXAMPLES**

63525 None.

63526 **APPLICATION USAGE**

63527 The distinction between *setjmp()/longjmp()* and *sigsetjmp()/siglongjmp()* is only significant for  
 63528 programs which use *sigaction()*, *sigprocmask()*, or *sigsuspend()*.

63529 Note that since this function is defined in terms of *setjmp()*, if *savemask* is zero, it is unspecified  
 63530 whether the signal mask is saved.

63531 **RATIONALE**

63532 The ISO C standard specifies various restrictions on the usage of the *setjmp()* macro in order to  
 63533 permit implementors to recognize the name in the compiler and not implement an actual  
 63534 function. These same restrictions apply to the *sigsetjmp()* macro.

63535 There are processors that cannot easily support these calls, but this was not considered a  
 63536 sufficient reason to exclude them.

63537 4.2 BSD, 4.3 BSD, and XSI-conformant systems provide functions named *\_setjmp()* and  
 63538 *\_longjmp()* that, together with *setjmp()* and *longjmp()*, provide the same functionality as  
 63539 *sigsetjmp()* and *siglongjmp()*. On those systems, *setjmp()* and *longjmp()* save and restore signal  
 63540 masks, while *\_setjmp()* and *\_longjmp()* do not. On System V Release 3 and in corresponding  
 63541 issues of the SVID, *setjmp()* and *longjmp()* are explicitly defined not to save and restore signal  
 63542 masks. In order to permit existing practice in both cases, the relation of *setjmp()* and *longjmp()* to  
 63543 signal masks is not specified, and a new set of functions is defined instead.

63544 The *longjmp()* and *siglongjmp()* functions operate as in the previous issue provided the matching  
 63545 *setjmp()* or *sigsetjmp()* has been performed in the same thread. Non-local jumps into contexts  
 63546 saved by other threads would be at best a questionable practice and were not considered worthy  
 63547 of standardization.

63548 **FUTURE DIRECTIONS**

63549 None.

63550 **SEE ALSO**63551 *pthread\_sigmask(), siglongjmp(), signal(), sigsuspend()*63552 XBD **<setjmp.h>**63553 **CHANGE HISTORY**

63554 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

63555 **Issue 5**

63556 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

63557 **Issue 6**63558 The DESCRIPTION is reworded in terms of *setjmp()*.63559 The SYNOPSIS is marked CX since the presence of this function in the **<setjmp.h>** header is an  
63560 extension over the ISO C standard.

63561 **NAME**

63562 sigsuspend — wait for a signal

63563 **SYNOPSIS**

```
63564 CX    #include <signal.h>
63565      int sigsuspend(const sigset_t *sigmask);
```

63566 **DESCRIPTION**

63567 The *sigsuspend()* function shall replace the current signal mask of the calling thread with the set  
63568 of signals pointed to by *sigmask* and then suspend the thread until delivery of a signal whose  
63569 action is either to execute a signal-catching function or to terminate the process. This shall not  
63570 cause any other signals that may have been pending on the process to become pending on the  
63571 thread.

63572 If the action is to terminate the process then *sigsuspend()* shall never return. If the action is to  
63573 execute a signal-catching function, then *sigsuspend()* shall return after the signal-catching  
63574 function returns, with the signal mask restored to the set that existed prior to the *sigsuspend()*  
63575 call.

63576 It is not possible to block signals that cannot be ignored. This is enforced by the system without  
63577 causing an error to be indicated.

63578 **RETURN VALUE**

63579 Since *sigsuspend()* suspends thread execution indefinitely, there is no successful completion  
63580 return value. If a return occurs, -1 shall be returned and *errno* set to indicate the error.

63581 **ERRORS**

63582 The *sigsuspend()* function shall fail if:

|       |         |                                                                            |
|-------|---------|----------------------------------------------------------------------------|
| 63583 | [EINTR] | A signal is caught by the calling process and control is returned from the |
| 63584 |         | signal-catching function.                                                  |

63585 **EXAMPLES**

63586 None.

63587 **APPLICATION USAGE**

63588 Normally, at the beginning of a critical code section, a specified set of signals is blocked using  
63589 the *sigprocmask()* function. When the thread has completed the critical section and needs to wait  
63590 for the previously blocked signal(s), it pauses by calling *sigsuspend()* with the mask that was  
63591 returned by the *sigprocmask()* call.

63592 **RATIONALE**

63593 Code which wants to avoid the ambiguity of the signal mask for thread cancellation handlers  
63594 can install an additional cancellation handler which resets the signal mask to the expected value.

```
63595 void cleanup(void *arg)
63596 {
63597     sigset_t *ss = (sigset_t *) arg;
63598     pthread_sigmask(SIG_SETMASK, ss, NULL);
63599 }
63600 int call_sigsuspend(const sigset_t *mask)
63601 {
63602     sigset_t oldmask;
63603     int result;
63604     pthread_sigmask(SIG_SETMASK, NULL, &oldmask);
63605     pthread_cleanup_push(cleanup, &oldmask);
```

```
63606         result = sigsuspend(sigmask);
63607         pthread_cleanup_pop(0);
63608         return result;
63609     }
```

#### 63610 FUTURE DIRECTIONS

63611 None.

#### 63612 SEE ALSO

63613 [Section 2.4](#) (on page 488), [pause\(\)](#), [sigaction\(\)](#), [sigaddset\(\)](#), [sigdelset\(\)](#), [sigemptyset\(\)](#), [sigfillset\(\)](#)

63614 XBD [<signal.h>](#)

#### 63615 CHANGE HISTORY

63616 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

##### 63617 Issue 5

63618 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

##### 63619 Issue 6

63620 The text in the RETURN VALUE section has been changed from “suspends process execution”  
63621 to “suspends thread execution”. This reflects IEEE PASC Interpretation 1003.1c #40.

63622 Text in the APPLICATION USAGE section has been replaced.

63623 The SYNOPSIS is marked CX since the presence of this function in the [<signal.h>](#) header is an  
63624 extension over the ISO C standard.

##### 63625 Issue 7

63626 SD5-XSH-ERN-122 is applied, adding the example code in the RATIONALE.

**NAME**

sigtimedwait, sigwaitinfo — wait for queued signals

**SYNOPSIS**

```
CX      #include <signal.h>

        int sigtimedwait(const sigset_t *restrict set,
        siginfo_t *restrict info,
        const struct timespec *restrict timeout);
        int sigwaitinfo(const sigset_t *restrict set,
        siginfo_t *restrict info);
```

**DESCRIPTION**

The *sigtimedwait()* function shall be equivalent to *sigwaitinfo()* except that if none of the signals specified by *set* are pending, *sigtimedwait()* shall wait for the time interval specified in the **timespec** structure referenced by *timeout*. If the **timespec** structure pointed to by *timeout* is zero-valued and if none of the signals specified by *set* are pending, then *sigtimedwait()* shall return immediately with an error. If *timeout* is the null pointer, the behavior is unspecified. If the Monotonic Clock option is supported, the **CLOCK\_MONOTONIC** clock shall be used to measure the time interval specified by the *timeout* argument.

The *sigwaitinfo()* function selects the pending signal from the set specified by *set*. Should any of multiple pending signals in the range SIGRTMIN to SIGRTMAX be selected, it shall be the lowest numbered one. The selection order between realtime and non-realtime signals, or between multiple pending non-realtime signals, is unspecified. If no signal in *set* is pending at the time of the call, the calling thread shall be suspended until one or more signals in *set* become pending or until it is interrupted by an unblocked, caught signal.

The *sigwaitinfo()* function shall be equivalent to the *sigwait()* function, except that the return value and the error reporting method are different (see RETURN VALUE), and that if the *info* argument is non-NULL, the selected signal number shall be stored in the *si\_signo* member, and the cause of the signal shall be stored in the *si\_code* member. If any value is queued to the selected signal, the first such queued value shall be dequeued and, if the *info* argument is non-NULL, the value shall be stored in the *si\_value* member of *info*. The system resource used to queue the signal shall be released and returned to the system for other use. If no value is queued, the content of the *si\_value* member is undefined. If no further signals are queued for the selected signal, the pending indication for that signal shall be reset.

**RETURN VALUE**

Upon successful completion (that is, one of the signals specified by *set* is pending or is generated) *sigwaitinfo()* and *sigtimedwait()* shall return the selected signal number. Otherwise, the function shall return a value of -1 and set *errno* to indicate the error.

**ERRORS**

The *sigtimedwait()* function shall fail if:

[EAGAIN] No signal specified by *set* was generated within the specified timeout period.

The *sigtimedwait()* and *sigwaitinfo()* functions may fail if:

[EINTR] The wait was interrupted by an unblocked, caught signal. It shall be documented in system documentation whether this error causes these functions to fail.

63670 The *sigtimedwait()* function may also fail if:  
 63671 [EINVAL] The *timeout* argument specified a *tv\_nsec* value less than zero or greater than  
 63672 or equal to 1 000 million.

63673 An implementation should only check for this error if no signal is pending in *set* and it is  
 63674 necessary to wait.

#### 63675 EXAMPLES

63676 None.

#### 63677 APPLICATION USAGE

63678 The *sigtimedwait()* function times out and returns an [EAGAIN] error. Application developers  
 63679 should note that this is inconsistent with other functions such as *pthread\_cond\_timedwait()* that  
 63680 return [ETIMEDOUT].

63681 Note that in order to ensure that generated signals are queued and signal values passed to  
 63682 *sigqueue()* are available in *si\_value*, applications which use *sigwaitinfo()* or *sigtimedwait()* need to  
 63683 set the SA\_SIGINFO flag for each signal in the set (see [Section 2.4](#), on page 488). This means  
 63684 setting each signal to be handled by a three-argument signal-catching function, even if the  
 63685 handler will never be called. It is not possible (portably) to set a signal handler to SIG\_DFL  
 63686 while setting the SA\_SIGINFO flag, because assigning to the *sa\_handler* member of **struct**  
 63687 **sigaction** instead of the *sa\_sigaction* member would result in undefined behavior, and SIG\_DFL  
 63688 need not be assignment-compatible with *sa\_sigaction*. Even if an assignment of SIG\_DFL to  
 63689 *sa\_sigaction* is accepted by the compiler, the implementation need not treat this value as special—  
 63690 it could just be taken as the address of a signal-catching function.

#### 63691 RATIONALE

63692 Existing programming practice on realtime systems uses the ability to pause waiting for a  
 63693 selected set of events and handle the first event that occurs in-line instead of in a signal-handling  
 63694 function. This allows applications to be written in an event-directed style similar to a state  
 63695 machine. This style of programming is useful for largescale transaction processing in which the  
 63696 overall throughput of an application and the ability to clearly track states are more important  
 63697 than the ability to minimize the response time of individual event handling.

63698 It is possible to construct a signal-waiting macro function out of the realtime signal function  
 63699 mechanism defined in this volume of POSIX.1-2008. However, such a macro has to include the  
 63700 definition of a generalized handler for all signals to be waited on. A significant portion of the  
 63701 overhead of handler processing can be avoided if the signal-waiting function is provided by the  
 63702 kernel. This volume of POSIX.1-2008 therefore provides two signal-waiting functions—one that  
 63703 waits indefinitely and one with a timeout—as part of the overall realtime signal function  
 63704 specification.

63705 The specification of a function with a timeout allows an application to be written that can be  
 63706 broken out of a wait after a set period of time if no event has occurred. It was argued that setting  
 63707 a timer event before the wait and recognizing the timer event in the wait would also implement  
 63708 the same functionality, but at a lower performance level. Because of the performance  
 63709 degradation associated with the user-level specification of a timer event and the subsequent  
 63710 cancellation of that timer event after the wait completes for a valid event, and the complexity  
 63711 associated with handling potential race conditions associated with the user-level method, the  
 63712 separate function has been included.

63713 Note that the semantics of the *sigwaitinfo()* function are nearly identical to that of the *sigwait()*  
 63714 function defined by this volume of POSIX.1-2008. The only difference is that *sigwaitinfo()* returns  
 63715 the queued signal value in the *value* argument. The return of the queued value is required so that  
 63716 applications can differentiate between multiple events queued to the same signal number.

The two distinct functions are being maintained because some implementations may choose to implement the POSIX Threads Extension functions and not implement the queued signals extensions. Note, though, that *sigwaitinfo()* does not return the queued value if the *value* argument is NULL, so the POSIX Threads Extension *sigwait()* function can be implemented as a macro on *sigwaitinfo()*.

The *sigtimedwait()* function was separated from the *sigwaitinfo()* function to address concerns regarding the overloading of the *timeout* pointer to indicate indefinite wait (no timeout), timed wait, and immediate return, and concerns regarding consistency with other functions where the conditional and timed waits were separate functions from the pure blocking function. The semantics of *sigtimedwait()* are specified such that *sigwaitinfo()* could be implemented as a macro with a null pointer for *timeout*.

The *sigwait* functions provide a synchronous mechanism for threads to wait for asynchronously-generated signals. One important question was how many threads that are suspended in a call to a *sigwait()* function for a signal should return from the call when the signal is sent. Four choices were considered:

1. Return an error for multiple simultaneous calls to *sigwait* functions for the same signal.
2. One or more threads return.
3. All waiting threads return.
4. Exactly one thread returns.

Prohibiting multiple calls to *sigwait()* for the same signal was felt to be overly restrictive. The “one or more” behavior made implementation of conforming packages easy at the expense of forcing POSIX threads clients to protect against multiple simultaneous calls to *sigwait()* in application code in order to achieve predictable behavior. There was concern that the “all waiting threads” behavior would result in “signal broadcast storms”, consuming excessive CPU resources by replicating the signals in the general case. Furthermore, no convincing examples could be presented that delivery to all was either simpler or more powerful than delivery to one.

Thus, the consensus was that exactly one thread that was suspended in a call to a *sigwait* function for a signal should return when that signal occurs. This is not an onerous restriction as:

- A multi-way signal wait can be built from the single-way wait.
- Signals should only be handled by application-level code, as library routines cannot guess what the application wants to do with signals generated for the entire process.
- Applications can thus arrange for a single thread to wait for any given signal and call any needed routines upon its arrival.

In an application that is using signals for interprocess communication, signal processing is typically done in one place. Alternatively, if the signal is being caught so that process cleanup can be done, the signal handler thread can call separate process cleanup routines for each portion of the application. Since the application main line started each portion of the application, it is at the right abstraction level to tell each portion of the application to clean up.

Certainly, there exist programming styles where it is logical to consider waiting for a single signal in multiple threads. A simple *sigwait\_multiple()* routine can be constructed to achieve this goal. A possible implementation would be to have each *sigwait\_multiple()* caller registered as having expressed interest in a set of signals. The caller then waits on a thread-specific condition variable. A single server thread calls a *sigwait()* function on the union of all registered signals. When the *sigwait()* function returns, the appropriate state is set and condition variables are broadcast. New *sigwait\_multiple()* callers may cause the pending *sigwait()* call to be canceled



63762 and reissued in order to update the set of signals being waited for.

#### 63763 **FUTURE DIRECTIONS**

63764 None.

#### 63765 **SEE ALSO**

63766 [Section 2.4](#) (on page 488), [Section 2.8.1](#) (on page 503), [pause\(\)](#), [pthread\\_sigmask\(\)](#), [sigaction\(\)](#),  
63767 [sigpending\(\)](#), [sigsuspend\(\)](#), [sigwait\(\)](#)

63768 XBD [<signal.h>](#), [<time.h>](#)

#### 63769 **CHANGE HISTORY**

63770 First released in Issue 5. Included for alignment with the POSIX Realtime Extension and the  
63771 POSIX Threads Extension.

#### 63772 **Issue 6**

63773 These functions are marked as part of the Realtime Signals Extension option.

63774 The Open Group Corrigendum U035/3 is applied. The SYNOPSIS of the *sigwaitinfo()* function  
63775 has been corrected so that the second argument is of type **siginfo\_t** \*.

63776 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
63777 implementation does not support the Realtime Signals Extension option.

63778 The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that the  
63779 CLOCK\_MONOTONIC clock, if supported, is used to measure timeout intervals.

63780 The **restrict** keyword is added to the *sigtimedwait()* and *sigwaitinfo()* prototypes for alignment  
63781 with the ISO/IEC 9899:1999 standard.

63782 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/130 is applied, restoring wording in the  
63783 RETURN VALUE section to that in the original base document (“An implementation should  
63784 only check for this error if no signal is pending in *set* and it is necessary to wait”).

#### 63785 **Issue 7**

63786 The *sigtimedwait()* and *sigwaitinfo()* functions are moved from the Realtime Signals Extension  
63787 option to the Base.

63788 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0583 [392] is applied.



63789 **NAME**

63790 sigwait — wait for queued signals

63791 **SYNOPSIS**

```
63792 CX      #include <signal.h>
63793      int sigwait(const sigset_t *restrict set, int *restrict sig);
```

63794 **DESCRIPTION**

63795 The *sigwait()* function shall select a pending signal from *set*, atomically clear it from the system's  
 63796 set of pending signals, and return that signal number in the location referenced by *sig*. If prior to  
 63797 the call to *sigwait()* there are multiple pending instances of a single signal number, it is  
 63798 implementation-defined whether upon successful return there are any remaining pending  
 63799 signals for that signal number. If the implementation supports queued signals and there are  
 63800 multiple signals queued for the signal number selected, the first such queued signal shall cause a  
 63801 return from *sigwait()* and the remainder shall remain queued. If no signal in *set* is pending at the  
 63802 time of the call, the thread shall be suspended until one or more becomes pending. The signals  
 63803 defined by *set* shall have been blocked at the time of the call to *sigwait()*; otherwise, the behavior  
 63804 is undefined. The effect of *sigwait()* on the signal actions for the signals in *set* is unspecified.

63805 If more than one thread is using *sigwait()* to wait for the same signal, no more than one of these  
 63806 threads shall return from *sigwait()* with the signal number. If more than a single thread is  
 63807 blocked in *sigwait()* for a signal when that signal is generated for the process, it is unspecified  
 63808 which of the waiting threads returns from *sigwait()*. If the signal is generated for a specific  
 63809 thread, as by *pthread\_kill()*, only that thread shall return.

63810 Should any of the multiple pending signals in the range SIGRTMIN to SIGRTMAX be selected, it  
 63811 shall be the lowest numbered one. The selection order between realtime and non-realtime  
 63812 signals, or between multiple pending non-realtime signals, is unspecified.

63813 **RETURN VALUE**

63814 Upon successful completion, *sigwait()* shall store the signal number of the received signal at the  
 63815 location referenced by *sig* and return zero. Otherwise, an error number shall be returned to  
 63816 indicate the error.

63817 **ERRORS**

63818 The *sigwait()* function may fail if:

63819 [EINVAL] The *set* argument contains an invalid or unsupported signal number.

63820 **EXAMPLES**

63821 None.

63822 **APPLICATION USAGE**

63823 None.

63824 **RATIONALE**

63825 To provide a convenient way for a thread to wait for a signal, this volume of POSIX.1-2008  
 63826 provides the *sigwait()* function. For most cases where a thread has to wait for a signal, the  
 63827 *sigwait()* function should be quite convenient, efficient, and adequate.

63828 However, requests were made for a lower-level primitive than *sigwait()* and for semaphores that  
 63829 could be used by threads. After some consideration, threads were allowed to use semaphores  
 63830 and *sem\_post()* was defined to be async-signal-safe.

63831 In summary, when it is necessary for code run in response to an asynchronous signal to notify a  
 63832 thread, *sigwait()* should be used to handle the signal. Alternatively, if the implementation  
 63833 provides semaphores, they also can be used, either following *sigwait()* or from within a signal

63834 handling routine previously registered with *sigaction()*.

#### 63835 FUTURE DIRECTIONS

63836 None.

#### 63837 SEE ALSO

63838 Section 2.4 (on page 488), Section 2.8.1 (on page 503), *pause()*, *pthread\_sigmask()*, *sigaction()*,  
63839 *sigpending()*, *sigsuspend()*, *sigtimedwait()*

63840 XBD <signal.h>, <time.h>

#### 63841 CHANGE HISTORY

63842 First released in Issue 5. Included for alignment with the POSIX Realtime Extension and the  
63843 POSIX Threads Extension.

#### 63844 Issue 6

63845 The **restrict** keyword is added to the *sigwait()* prototype for alignment with the  
63846 ISO/IEC 9899:1999 standard.

63847 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/131 is applied, updating the  
63848 DESCRIPTION to state that if more than a single thread is blocked in *sigwait()*, it is unspecified  
63849 which of the waiting threads returns, and that if a signal is generated for a specific thread only  
63850 that thread shall return.

#### 63851 Issue 7

63852 Functionality relating to the Realtime Signals Extension option is moved to the Base.

63853 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0584 [76] is applied.

63854 **NAME**

63855 sigwaitinfo — wait for queued signals

63856 **SYNOPSIS**

```
63857 #include <signal.h>  
63858 int sigwaitinfo(const sigset_t *restrict set, siginfo_t *restrict info);
```

63859 **DESCRIPTION**63860 Refer to *sigtimedwait()*.

63861 **NAME**

63862 sin, sinf, sinl — sine function

63863 **SYNOPSIS**

```
63864 #include <math.h>
63865 double sin(double x);
63866 float sinf(float x);
63867 long double sinl(long double x);
```

63868 **DESCRIPTION**

63869 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 63870 conflict between the requirements described here and the ISO C standard is unintentional. This  
 63871 volume of POSIX.1-2008 defers to the ISO C standard.

63872 These functions shall compute the sine of their argument  $x$ , measured in radians.

63873 An application wishing to check for error situations should set *errno* to zero and call  
 63874 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 63875 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 63876 zero, an error has occurred.

63877 **RETURN VALUE**

63878 Upon successful completion, these functions shall return the sine of  $x$ .

63879 MX If  $x$  is NaN, a NaN shall be returned.

63880 If  $x$  is  $\pm 0$ ,  $x$  shall be returned.

63881 If  $x$  is subnormal, a range error may occur

63882 MXX and  $x$  should be returned.

63883 MX If  $x$  is not returned, *sin()*, *sinf()*, and *sinl()* shall return an implementation-defined value no  
 63884 greater in magnitude than DBL\_MIN, FLT\_MIN, and LDBL\_MIN, respectively.

63885 If  $x$  is  $\pm\text{Inf}$ , a domain error shall occur, and a NaN shall be returned.

63886 **ERRORS**

63887 These functions shall fail if:

63888 MX **Domain Error** The  $x$  argument is  $\pm\text{Inf}$ .

63889 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 63890 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 63891 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 63892 shall be raised.

63893 These functions may fail if:

63894 MX **Range Error** The value of  $x$  is subnormal

63895 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 63896 then *errno* shall be set to [ERANGE]. If the integer expression  
 63897 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 63898 floating-point exception shall be raised.

63899 **EXAMPLES**63900 **Taking the Sine of a 45-Degree Angle**

```

63901 #include <math.h>
63902 ...
63903 double radians = 45.0 * M_PI / 180;
63904 double result;
63905 ...
63906 result = sin(radians);

```

63907 **APPLICATION USAGE**

63908 These functions may lose accuracy when their argument is near a multiple of  $\pi$  or is far from 0.0.

63909 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 63910 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

63911 **RATIONALE**

63912 None.

63913 **FUTURE DIRECTIONS**

63914 None.

63915 **SEE ALSO**

63916 *asin()*, *feclearexcept()*, *fetestexcept()*, *isnan()*

63917 XBD Section 4.20 (on page 117), *<math.h>*

63918 **CHANGE HISTORY**

63919 First released in Issue 1. Derived from Issue 1 of the SVID.

63920 **Issue 5**

63921 The last two paragraphs of the DESCRIPTION were included as APPLICATION USAGE notes  
 63922 in previous issues.

63923 **Issue 6**

63924 The *sinf()* and *sinl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

63925 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
 63926 revised to align with the ISO/IEC 9899:1999 standard.

63927 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
 63928 marked.

63929 **Issue 7**

63930 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0585 [68] and XSH/TC1-2008/0586  
 63931 [320] are applied.

63932 **NAME**

63933       sinh, sinh, sinhl — hyperbolic sine functions

63934 **SYNOPSIS**

```
63935     #include <math.h>
63936     double sinh(double x);
63937     float  sinhf(float x);
63938     long double sinhl(long double x);
```

63939 **DESCRIPTION**

63940 **CX**       The functionality described on this reference page is aligned with the ISO C standard. Any  
 63941 conflict between the requirements described here and the ISO C standard is unintentional. This  
 63942 volume of POSIX.1-2008 defers to the ISO C standard.

63943       These functions shall compute the hyperbolic sine of their argument *x*.

63944       An application wishing to check for error situations should set *errno* to zero and call  
 63945 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 63946 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 63947 zero, an error has occurred.

63948 **RETURN VALUE**

63949       Upon successful completion, these functions shall return the hyperbolic sine of *x*.

63950       If the result would cause an overflow, a range error shall occur and  $\pm\text{HUGE\_VAL}$ ,  
 63951  $\pm\text{HUGE\_VALF}$ , and  $\pm\text{HUGE\_VALL}$  (with the same sign as *x*) shall be returned as appropriate for  
 63952 the type of the function.

63953 **MX**       If *x* is NaN, a NaN shall be returned.

63954       If *x* is  $\pm 0$  or  $\pm\text{Inf}$ , *x* shall be returned.

63955       If *x* is subnormal, a range error may occur

63956 **MXX**       and *x* should be returned.

63957 **MX**       If *x* is not returned, *sinh()*, *sinhf()*, and *sinhl()* shall return an implementation-defined value no  
 63958 greater in magnitude than DBL\_MIN, FLT\_MIN, and LDBL\_MIN, respectively.

63959 **ERRORS**

63960       These functions shall fail if:

63961       Range Error       The result would cause an overflow.

63962       If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 63963 then *errno* shall be set to [ERANGE]. If the integer expression  
 63964 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the overflow  
 63965 floating-point exception shall be raised.

63966       These functions may fail if:

63967 **MX**       Range Error       The value *x* is subnormal.

63968       If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 63969 then *errno* shall be set to [ERANGE]. If the integer expression  
 63970 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 63971 floating-point exception shall be raised.

63972 **EXAMPLES**

63973 None.

63974 **APPLICATION USAGE**

63975 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 63976 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

63977 **RATIONALE**

63978 None.

63979 **FUTURE DIRECTIONS**

63980 None.

63981 **SEE ALSO**63982 *asinh()*, *cosh()*, *feclearexcept()*, *fetestexcept()*, *isnan()*, *tanh()*63983 XBD Section 4.20 (on page 117), **<math.h>**63984 **CHANGE HISTORY**

63985 First released in Issue 1. Derived from Issue 1 of the SVID.

63986 **Issue 5**

63987 The DESCRIPTION is updated to indicate how an application should check for an error. This  
 63988 text was previously published in the APPLICATION USAGE section.

63989 **Issue 6**63990 The *sinhf()* and *sinhl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

63991 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
 63992 revised to align with the ISO/IEC 9899:1999 standard.

63993 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
 63994 marked.

63995 **Issue 7**

63996 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0587 [68] is applied.

**63997 NAME**

63998       sinl — sine function

**63999 SYNOPSIS**

64000       #include <math.h>

64001       long double sinl(long double x);

**64002 DESCRIPTION**

64003       Refer to *sin()*.



64004 **NAME**

64005 sleep — suspend execution for an interval of time

64006 **SYNOPSIS**

64007 #include &lt;unistd.h&gt;

64008 unsigned sleep(unsigned *seconds*);64009 **DESCRIPTION**

64010 The *sleep()* function shall cause the calling thread to be suspended from execution until either  
 64011 the number of realtime seconds specified by the argument *seconds* has elapsed or a signal is  
 64012 delivered to the calling thread and its action is to invoke a signal-catching function or to  
 64013 terminate the process. The suspension time may be longer than requested due to the scheduling  
 64014 of other activity by the system.

64015 In single-threaded programs, *sleep()* may make use of SIGALRM. In multi-threaded programs, +  
 64016 *sleep()* shall not make use of SIGALRM and the remainder of this DESCRIPTION does not apply. +

64017 If a SIGALRM signal is generated for the calling process during execution of *sleep()* and if the  
 64018 SIGALRM signal is being ignored or blocked from delivery, it is unspecified whether *sleep()*  
 64019 returns when the SIGALRM signal is scheduled. If the signal is being blocked, it is also  
 64020 unspecified whether it remains pending after *sleep()* returns or it is discarded.

64021 If a SIGALRM signal is generated for the calling process during execution of *sleep()*, except as a  
 64022 result of a prior call to *alarm()*, and if the SIGALRM signal is not being ignored or blocked from  
 64023 delivery, it is unspecified whether that signal has any effect other than causing *sleep()* to return.

64024 If a signal-catching function interrupts *sleep()* and examines or changes either the time a  
 64025 SIGALRM is scheduled to be generated, the action associated with the SIGALRM signal, or  
 64026 whether the SIGALRM signal is blocked from delivery, the results are unspecified.

64027 If a signal-catching function interrupts *sleep()* and calls *siglongjmp()* or *longjmp()* to restore an  
 64028 environment saved prior to the *sleep()* call, the action associated with the SIGALRM signal and  
 64029 the time at which a SIGALRM signal is scheduled to be generated are unspecified. It is also  
 64030 unspecified whether the SIGALRM signal is blocked, unless the signal mask of the process is  
 64031 restored as part of the environment.

64032 XSI Interactions between *sleep()* and *setitimer()* are unspecified.

64033 **RETURN VALUE**

64034 If *sleep()* returns because the requested time has elapsed, the value returned shall be 0. If *sleep()*  
 64035 returns due to delivery of a signal, the return value shall be the “unslept” amount (the requested  
 64036 time minus the time actually slept) in seconds.

64037 **ERRORS**

64038 No errors are defined.

64039 **EXAMPLES**

64040 None.

64041 **APPLICATION USAGE**

64042 None.

64043 **RATIONALE**

64044 There are two general approaches to the implementation of the *sleep()* function. One is to use the  
 64045 *alarm()* function to schedule a SIGALRM signal and then suspend the calling thread waiting for  
 64046 that signal. The other is to implement an independent facility. This volume of POSIX.1-2008  
 64047 permits either approach in single-threaded programs, but the simple alarm/suspend  
 64048 implementation is not appropriate for multi-threaded programs.

In order to comply with the requirement that no primitive shall change a process attribute unless explicitly described by this volume of POSIX.1-2008, an implementation using SIGALRM must carefully take into account any SIGALRM signal scheduled by previous *alarm()* calls, the action previously established for SIGALRM, and whether SIGALRM was blocked. If a SIGALRM has been scheduled before the *sleep()* would ordinarily complete, the *sleep()* must be shortened to that time and a SIGALRM generated (possibly simulated by direct invocation of the signal-catching function) before *sleep()* returns. If a SIGALRM has been scheduled after the *sleep()* would ordinarily complete, it must be rescheduled for the same time before *sleep()* returns. The action and blocking for SIGALRM must be saved and restored.

Historical implementations often implement the SIGALRM-based version using *alarm()* and *pause()*. One such implementation is prone to infinite hangups, as described in *pause()*. Another such implementation uses the C-language *setjmp()* and *longjmp()* functions to avoid that window. That implementation introduces a different problem: when the SIGALRM signal interrupts a signal-catching function installed by the user to catch a different signal, the *longjmp()* aborts that signal-catching function. An implementation based on *sigprocmask()*, *alarm()*, and *sigsuspend()* can avoid these problems.

Despite all reasonable care, there are several very subtle, but detectable and unavoidable, differences between the two types of implementations. These are the cases mentioned in this volume of POSIX.1-2008 where some other activity relating to SIGALRM takes place, and the results are stated to be unspecified. All of these cases are sufficiently unusual as not to be of concern to most applications.

See also the discussion of the term *realtime* in *alarm()*.

Since *sleep()* can be implemented using *alarm()*, the discussion about alarms occurring early under *alarm()* applies to *sleep()* as well.

Application developers should note that the type of the argument *seconds* and the return value of *sleep()* is **unsigned**. That means that a Strictly Conforming POSIX System Interfaces Application cannot pass a value greater than the minimum guaranteed value for {UINT\_MAX}, which the ISO C standard sets as 65535, and any application passing a larger value is restricting its portability. A different type was considered, but historical implementations, including those with a 16-bit **int** type, consistently use either **unsigned** or **int**.

Scheduling delays may cause the process to return from the *sleep()* function significantly after the requested time. In such cases, the return value should be set to zero, since the formula (requested time minus the time actually spent) yields a negative number and *sleep()* returns an **unsigned**.

## FUTURE DIRECTIONS

A future version of this standard may require that *sleep()* does not make use of SIGALRM in all programs, not just multi-threaded programs.

## SEE ALSO

*alarm()*, *getitimer()*, *nanosleep()*, *pause()*, *sigaction()*, *sigsetjmp()*

XBD <**unistd.h**>

## CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

## Issue 5

The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

|       |                |                                                                                             |
|-------|----------------|---------------------------------------------------------------------------------------------|
| 64093 | <b>Issue 6</b> |                                                                                             |
| 64094 |                | IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/132 is applied, making a correction in the |
| 64095 |                | RATIONALE section.                                                                          |

64096 **NAME**

64097       snprintf — print formatted output

64098 **SYNOPSIS**

64099       #include &lt;stdio.h&gt;

64100       int snprintf(char \*restrict *s*, size\_t *n*,64101           const char \*restrict *format*, ...);64102 **DESCRIPTION**64103       Refer to *fprintf()*.

**64104 NAME**

64105 sockatmark — determine whether a socket is at the out-of-band mark

**64106 SYNOPSIS**

64107 #include <sys/socket.h>

64108 int sockatmark(int s);

**64109 DESCRIPTION**

64110 The *sockatmark()* function shall determine whether the socket specified by the descriptor *s* is at  
64111 the out-of-band data mark (see [Section 2.10.12](#), on page 527). If the protocol for the socket  
64112 supports out-of-band data by marking the stream with an out-of-band data mark, the  
64113 *sockatmark()* function shall return 1 when all data preceding the mark has been read and the out-  
64114 of-band data mark is the first element in the receive queue. The *sockatmark()* function shall not  
64115 remove the mark from the stream.

**64116 RETURN VALUE**

64117 Upon successful completion, the *sockatmark()* function shall return a value indicating whether  
64118 the socket is at an out-of-band data mark. If the protocol has marked the data stream and all data  
64119 preceding the mark has been read, the return value shall be 1; if there is no mark, or if data  
64120 precedes the mark in the receive queue, the *sockatmark()* function shall return 0. Otherwise, it  
64121 shall return a value of -1 and set *errno* to indicate the error.

**64122 ERRORS**

64123 The *sockatmark()* function shall fail if:

64124 [EBADF] The *s* argument is not a valid file descriptor.

64125 [ENOTTY] The file associated with the *s* argument is not a socket.

**64126 EXAMPLES**

64127 None.

**64128 APPLICATION USAGE**

64129 The use of this function between receive operations allows an application to determine which  
64130 received data precedes the out-of-band data and which follows the out-of-band data.

64131 There is an inherent race condition in the use of this function. On an empty receive queue, the  
64132 current read of the location might well be at the “mark”, but the system has no way of knowing  
64133 that the next data segment that will arrive from the network will carry the mark, and  
64134 *sockatmark()* will return false, and the next read operation will silently consume the mark.

64135 Hence, this function can only be used reliably when the application already knows that the out-  
64136 of-band data has been seen by the system or that it is known that there is data waiting to be read  
64137 at the socket (via SIGURG or *select()*). See [Section 2.10.11](#) (on page 526), [Section 2.10.12](#) (on page  
64138 527), [Section 2.10.14](#) (on page 527), and *pselect()* for details.

**64139 RATIONALE**

64140 The *sockatmark()* function replaces the historical SIOCATMARK command to *ioctl()* which  
64141 implemented the same functionality on many implementations. Using a wrapper function  
64142 follows the adopted conventions to avoid specifying commands to the *ioctl()* function, other  
64143 than those now included to support XSI STREAMS. The *sockatmark()* function could be  
64144 implemented as follows:

64145 #include <sys/ioctl.h>

64146 int sockatmark(int s)

64147 {  
64148 int val;

```
64149         if (ioctl(s,SIOCATMARK,&val)==-1)
64150             return(-1);
64151         return(val);
64152     }
```

64153 The use of [ENOTTY] to indicate an incorrect descriptor type matches the historical behavior of  
64154 SIOCATMARK.

64155 **FUTURE DIRECTIONS**

64156 None.

64157 **SEE ALSO**

64158 [Section 2.10.12](#) (on page 527), [pselect\(\)](#), [recv\(\)](#), [recvmsg\(\)](#)

64159 XBD [<sys/socket.h>](#)

64160 **CHANGE HISTORY**

64161 First released in Issue 6. Derived from IEEE Std 1003.1g-2000.

64162 **Issue 7**

64163 SD5-XSH-ERN-100 is applied, correcting the definition of the [ENOTTY] error condition.

**NAME**

socket — create an endpoint for communication

**SYNOPSIS**

```
#include <sys/socket.h>

int socket(int domain, int type, int protocol);
```

**DESCRIPTION**

The *socket()* function shall create an unbound socket in a communications domain, and return a file descriptor that can be used in later function calls that operate on sockets. The file descriptor shall be allocated as described in [Section 2.14](#) (on page 549).

The *socket()* function takes the following arguments:

|                 |                                                                                                                                                                                                        |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>domain</i>   | Specifies the communications domain in which a socket is to be created.                                                                                                                                |
| <i>type</i>     | Specifies the type of socket to be created.                                                                                                                                                            |
| <i>protocol</i> | Specifies a particular protocol to be used with the socket. Specifying a <i>protocol</i> of 0 causes <i>socket()</i> to use an unspecified default protocol appropriate for the requested socket type. |

The *domain* argument specifies the address family used in the communications domain. The address families supported by the system are implementation-defined.

Symbolic constants that can be used for the domain argument are defined in the `<sys/socket.h>` header.

The *type* argument specifies the socket type, which determines the semantics of communication over the socket. The following socket types are defined; implementations may specify additional socket types:

**SOCK\_STREAM** Provides sequenced, reliable, bidirectional, connection-mode byte streams, and may provide a transmission mechanism for out-of-band data.

**SOCK\_DGRAM** Provides datagrams, which are connectionless-mode, unreliable messages of fixed maximum length.

**SOCK\_SEQPACKET**

Provides sequenced, reliable, bidirectional, connection-mode transmission paths for records. A record can be sent using one or more output operations and received using one or more input operations, but a single operation never transfers part of more than one record. Record boundaries are visible to the receiver via the MSG\_EOR flag.

If the *protocol* argument is non-zero, it shall specify a protocol that is supported by the address family. If the *protocol* argument is zero, the default protocol for this address family and type shall be used. The protocols supported by the system are implementation-defined.

The process may need to have appropriate privileges to use the *socket()* function or to create some sockets.

**RETURN VALUE**

Upon successful completion, *socket()* shall return a non-negative integer, the socket file descriptor. Otherwise, a value of `-1` shall be returned and *errno* set to indicate the error.

**ERRORS**

The *socket()* function shall fail if:

[EAFNOSUPPORT]

The implementation does not support the specified address family.

[EMFILE]

All file descriptors available to the process are currently open.

[ENFILE]

No more file descriptors are available for the system.

[EPROTONOSUPPORT]

The protocol is not supported by the address family, or the protocol is not supported by the implementation.

[EPROTOTYPE] The socket type is not supported by the protocol.

The *socket()* function may fail if:

[EACCES]

The process does not have appropriate privileges.

[ENOBUFS]

Insufficient resources were available in the system to perform the operation.

[ENOMEM]

Insufficient memory was available to fulfill the request.

**EXAMPLES**

None.

**APPLICATION USAGE**

The documentation for specific address families specifies which protocols each address family supports. The documentation for specific protocols specifies which socket types each protocol supports.

The application can determine whether an address family is supported by trying to create a socket with *domain* set to the protocol in question.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

Section 2.14 (on page 549), *accept()*, *bind()*, *connect()*, *getsockname()*, *getsockopt()*, *listen()*, *recv()*, *recvfrom()*, *recvmsg()*, *send()*, *sendmsg()*, *setsockopt()*, *shutdown()*, *socketpair()*

XBD <netinet/in.h>, <sys/socket.h>

**CHANGE HISTORY**

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.



**NAME**

socketpair — create a pair of connected sockets

**SYNOPSIS**

```
#include <sys/socket.h>

int socketpair(int domain, int type, int protocol,
               int socket_vector[2]);
```

**DESCRIPTION**

The *socketpair()* function shall create an unbound pair of connected sockets in a specified *domain*, of a specified *type*, under the protocol optionally specified by the *protocol* argument. The two sockets shall be identical. The file descriptors used in referencing the created sockets shall be returned in *socket\_vector*[0] and *socket\_vector*[1]. The file descriptors shall be allocated as described in [Section 2.14](#) (on page 549).

The *socketpair()* function takes the following arguments:

|                      |                                                                                                                                                                                                             |
|----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>domain</i>        | Specifies the communications domain in which the sockets are to be created.                                                                                                                                 |
| <i>type</i>          | Specifies the type of sockets to be created.                                                                                                                                                                |
| <i>protocol</i>      | Specifies a particular protocol to be used with the sockets. Specifying a <i>protocol</i> of 0 causes <i>socketpair()</i> to use an unspecified default protocol appropriate for the requested socket type. |
| <i>socket_vector</i> | Specifies a 2-integer array to hold the file descriptors of the created socket pair.                                                                                                                        |

The *type* argument specifies the socket type, which determines the semantics of communications over the socket. The following socket types are defined; implementations may specify additional socket types:

|                |                                                                                                                                                                                                                                                                                                                                                 |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SOCK_STREAM    | Provides sequenced, reliable, bidirectional, connection-mode byte streams, and may provide a transmission mechanism for out-of-band data.                                                                                                                                                                                                       |
| SOCK_DGRAM     | Provides datagrams, which are connectionless-mode, unreliable messages of fixed maximum length.                                                                                                                                                                                                                                                 |
| SOCK_SEQPACKET | Provides sequenced, reliable, bidirectional, connection-mode transmission paths for records. A record can be sent using one or more output operations and received using one or more input operations, but a single operation never transfers part of more than one record. Record boundaries are visible to the receiver via the MSG_EOR flag. |

If the *protocol* argument is non-zero, it shall specify a protocol that is supported by the address family. If the *protocol* argument is zero, the default protocol for this address family and type shall be used. The protocols supported by the system are implementation-defined.

The process may need to have appropriate privileges to use the *socketpair()* function or to create some sockets.

**RETURN VALUE**

Upon successful completion, this function shall return 0; otherwise,  $-1$  shall be returned and *errno* set to indicate the error, no file descriptors shall be allocated, and the contents of *socket\_vector* shall be left unmodified.

**ERRORS**

The *socketpair*( ) function shall fail if:

[EAFNOSUPPORT]

The implementation does not support the specified address family.

[EMFILE]

All, or all but one, of the file descriptors available to the process are currently open.

[ENFILE]

No more file descriptors are available for the system.

[EOPNOTSUPP] The specified protocol does not permit creation of socket pairs.

[EPROTONOSUPPORT]

The protocol is not supported by the address family, or the protocol is not supported by the implementation.

[EPROTOTYPE] The socket type is not supported by the protocol.

The *socketpair*( ) function may fail if:

[EACCES]

The process does not have appropriate privileges.

[ENOBUFS]

Insufficient resources were available in the system to perform the operation.

[ENOMEM]

Insufficient memory was available to fulfill the request.

**EXAMPLES**

None.

**APPLICATION USAGE**

The documentation for specific address families specifies which protocols each address family supports. The documentation for specific protocols specifies which socket types each protocol supports.

The *socketpair*( ) function is used primarily with UNIX domain sockets and need not be supported for other domains.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 2.14](#) (on page 549), *socket*( )

XBD [<sys/socket.h>](#)

**CHANGE HISTORY**

First released in Issue 6. Derived from the XNS, Issue 5.2 specification.

**Issue 7**

The description of the [EMFILE] error condition is aligned with the *pipe*( ) function.

64312 **NAME**

64313        sprintf — print formatted output

64314 **SYNOPSIS**

64315        #include &lt;stdio.h&gt;

64316        int sprintf(char \*restrict *s*, const char \*restrict *format*, ...);64317 **DESCRIPTION**64318        Refer to *fprintf()*.

64319 **NAME**

64320 sqrt, sqrtf, sqrtl — square root function

64321 **SYNOPSIS**

```
64322 #include <math.h>
64323 double sqrt(double x);
64324 float sqrtf(float x);
64325 long double sqrtl(long double x);
```

64326 **DESCRIPTION**

64327 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 64328 conflict between the requirements described here and the ISO C standard is unintentional. This  
 64329 volume of POSIX.1-2008 defers to the ISO C standard.

64330 These functions shall compute the square root of their argument  $x$ ,  $\sqrt{x}$ .

64331 An application wishing to check for error situations should set *errno* to zero and call  
 64332 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 64333 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 64334 zero, an error has occurred.

64335 **RETURN VALUE**

64336 Upon successful completion, these functions shall return the square root of  $x$ .

64337 MX For finite values of  $x < -0$ , a domain error shall occur, and either a NaN (if supported), or an  
 64338 implementation-defined value shall be returned.

64339 MX If  $x$  is NaN, a NaN shall be returned.

64340 If  $x$  is  $\pm 0$  or  $+\text{Inf}$ ,  $x$  shall be returned.

64341 If  $x$  is  $-\text{Inf}$ , a domain error shall occur, and a NaN shall be returned.

64342 **ERRORS**

64343 These functions shall fail if:

64344 MX Domain Error The finite value of  $x$  is  $< -0$ , or  $x$  is  $-\text{Inf}$ .

64345 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 64346 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 64347 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 64348 shall be raised.

64349 **EXAMPLES**64350 **Taking the Square Root of 9.0**

```
64351 #include <math.h>
64352 ...
64353 double x = 9.0;
64354 double result;
64355 ...
64356 result = sqrt(x);
```

64357 **APPLICATION USAGE**

64358 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 64359 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

64360 **RATIONALE**

64361 None.

64362 **FUTURE DIRECTIONS**

64363 None.

64364 **SEE ALSO**64365 *feclearexcept()*, *fetestexcept()*, *isnan()*64366 XBD Section 4.20 (on page 117), *<math.h>*, *<stdio.h>*64367 **CHANGE HISTORY**

64368 First released in Issue 1. Derived from Issue 1 of the SVID.

64369 **Issue 5**64370 The DESCRIPTION is updated to indicate how an application should check for an error. This  
64371 text was previously published in the APPLICATION USAGE section.64372 **Issue 6**64373 The *sqrtrf()* and *sqrtrl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.64374 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
64375 revised to align with the ISO/IEC 9899:1999 standard.64376 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
64377 marked.64378 **Issue 7**

64379 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0588 [320] is applied.

64380 **NAME**

64381       srand — pseudo-random number generator

64382 **SYNOPSIS**

64383       #include &lt;stdlib.h&gt;

64384       void srand(unsigned seed);

64385 **DESCRIPTION**64386       Refer to *rand()*.

64387 **NAME**

64388       srand48 — seed the uniformly distributed double-precision pseudo-random number generator

64389 **SYNOPSIS**

```
64390 XSI      #include <stdlib.h>
64391          void srand48(long seedval);
```

64392 **DESCRIPTION**64393       Refer to *drand48()*.

## 64394 NAME

64395 srandom — seed pseudo-random number generator

## 64396 SYNOPSIS

64397 XSI `#include <stdlib.h>`

64398 `void srandom(unsigned seed);`

## 64399 DESCRIPTION

64400 Refer to *initstate()*.



64401 **NAME**

64402       scanf — convert formatted input

64403 **SYNOPSIS**

64404       #include &lt;stdio.h&gt;

64405       int sscanf(const char \*restrict *s*, const char \*restrict *format*, ...);64406 **DESCRIPTION**64407       Refer to *fscanf()*.

64408 **NAME**

64409       stat — get file status

64410 **SYNOPSIS**

64411       #include &lt;sys/stat.h&gt;

64412       int stat(const char \*restrict *path*, struct stat \*restrict *buf*);64413 **DESCRIPTION**64414       Refer to *fstatat()*.

64415 **NAME**

64416 statvfs — get file system information

64417 **SYNOPSIS**

64418 #include &lt;sys/statvfs.h&gt;

64419 int statvfs(const char \*restrict *path*, struct statvfs \*restrict *buf*);64420 **DESCRIPTION**64421 Refer to *fstatvfs()*.

## 64422 NAME

64423 stderr, stdin, stdout — standard I/O streams

## 64424 SYNOPSIS

```
64425 #include <stdio.h>
64426 extern FILE *stderr, *stdin, *stdout;
```

## 64427 DESCRIPTION

64428 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
64429 conflict between the requirements described here and the ISO C standard is unintentional. This  
64430 volume of POSIX.1-2008 defers to the ISO C standard.

64431 A file with associated buffering is called a *stream* and is declared to be a pointer to a defined type  
64432 **FILE**. The *fopen()* function shall create certain descriptive data for a stream and return a pointer  
64433 to designate the stream in all further transactions. Normally, there are three open streams with  
64434 constant pointers declared in the **<stdio.h>** header and associated with the standard open files.

64435 At program start-up, three streams shall be predefined and need not be opened explicitly:  
64436 *standard input* (for reading conventional input), *standard output* (for writing conventional output),  
64437 and *standard error* (for writing diagnostic output). When opened, the standard error stream is not  
64438 fully buffered; the standard input and standard output streams are fully buffered if and only if  
64439 the stream can be determined not to refer to an interactive device.

64440 CX The following symbolic values in **<unistd.h>** define the file descriptors that shall be associated  
64441 with the C-language *stdin*, *stdout*, and *stderr* when the application is started:

|       |                                                                          |                                                        |
|-------|--------------------------------------------------------------------------|--------------------------------------------------------|
| 64442 | STDIN_FILENO                                                             | Standard input value, <i>stdin</i> . Its value is 0.   |
| 64443 | STDOUT_FILENO                                                            | Standard output value, <i>stdout</i> . Its value is 1. |
| 64444 | STDERR_FILENO                                                            | Standard error value, <i>stderr</i> . Its value is 2.  |
| 64445 | The <i>stderr</i> stream is expected to be open for reading and writing. |                                                        |

## 64446 RETURN VALUE

64447 None.

## 64448 ERRORS

64449 No errors are defined.

## 64450 EXAMPLES

64451 None.

## 64452 APPLICATION USAGE

64453 None.

## 64454 RATIONALE

64455 None.

## 64456 FUTURE DIRECTIONS

64457 None.

## 64458 SEE ALSO

64459 *fclose()*, *feof()*, *ferror()*, *fileno()*, *fopen()*, *fprintf()*, *fread()*, *fscanf()*, *fseek()*, *getc()*, *gets()*, *popen()*,  
64460 *putc()*, *puts()*, *read()*, *setbuf()*, *setvbuf()*, *tmpfile()*, *ungetc()*, *vfprintf()*

64461 XBD **<stdio.h>**, **<unistd.h>**

64462 **CHANGE HISTORY**

64463 First released in Issue 1.

64464 **Issue 6**

64465 Extensions beyond the ISO C standard are marked.

64466 A note that *stderr* is expected to be open for reading and writing is added to the DESCRIPTION.

64467 **NAME**

64468       strcpy — copy a string and return a pointer to the end of the result

64469 **SYNOPSIS**

```
64470 CX       #include <string.h>
64471       char *strcpy(char *restrict s1, const char *restrict s2);
```

64472 **DESCRIPTION**64473       Refer to *strcpy()*.

64474 **NAME**

64475 stpncpy — copy fixed length string, returning a pointer to the array end

64476 **SYNOPSIS**64477 CX `#include <string.h>`64478 `char *stpncpy(char *restrict s1, const char *restrict s2, size_t size);`64479 **DESCRIPTION**64480 Refer to *strncpy()*.

**NAME**

strcasemp, strcasemp\_l, strncasemp, strncasemp\_l — case-insensitive string comparisons

**SYNOPSIS**

```
#include <strings.h>

int strcasemp(const char *s1, const char *s2);
int strcasemp_l(const char *s1, const char *s2,
    locale_t locale);
int strncasemp(const char *s1, const char *s2, size_t n);
int strncasemp_l(const char *s1, const char *s2,
    size_t n, locale_t locale);
```

**DESCRIPTION**

The *strcasemp()* and *strcasemp\_l()* functions shall compare, while ignoring differences in case, the string pointed to by *s1* to the string pointed to by *s2*. The *strncasemp()* and *strncasemp\_l()* functions shall compare, while ignoring differences in case, not more than *n* bytes from the string pointed to by *s1* to the string pointed to by *s2*.

The *strcasemp()* and *strncasemp()* functions use the current locale to determine the case of the characters.

The *strcasemp\_l()* and *strncasemp\_l()* functions use the locale represented by *locale* to determine the case of the characters.

When the *LC\_CTYPE* category of the locale being used is from the POSIX locale, these functions shall behave as if the strings had been converted to lowercase and then a byte comparison performed. Otherwise, the results are unspecified.

The behavior is undefined if the *locale* argument to *strcasemp\_l()* or *strncasemp\_l()* is the special locale object *LC\_GLOBAL\_LOCALE* or is not a valid locale object handle.

**RETURN VALUE**

Upon completion, *strcasemp()* and *strcasemp\_l()* shall return an integer greater than, equal to, or less than 0, if the string pointed to by *s1* is, ignoring case, greater than, equal to, or less than the string pointed to by *s2*, respectively.

Upon successful completion, *strncasemp()* and *strncasemp\_l()* shall return an integer greater than, equal to, or less than 0, if the possibly null-terminated array pointed to by *s1* is, ignoring case, greater than, equal to, or less than the possibly null-terminated array pointed to by *s2*, respectively.

**ERRORS**

No errors are defined.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.



64523 **SEE ALSO**64524 [\*wscasecmp\(\)\*](#)64525 XBD [\*\*<strings.h>\*\*](#)64526 **CHANGE HISTORY**

64527 First released in Issue 4, Version 2.

64528 **Issue 5**

64529 Moved from X/OPEN UNIX extension to BASE.

64530 **Issue 7**64531 The *strcasecmp()* and *strncasecmp()* functions are moved from the XSI option to the Base.64532 The *strcasecmp\_l()* and *strncasecmp\_l()* functions are added from The Open Group Technical  
64533 Standard, 2006, Extended API Set Part 4.64534 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0589 [302], XSH/TC1-2008/0590 [294],  
64535 XSH/TC1-2008/0591 [283], and XSH/TC1-2008/0592 [283] are applied.

64536 **NAME**

64537       strcat — concatenate two strings

64538 **SYNOPSIS**

64539       #include &lt;string.h&gt;

64540       char \*strcat(char \*restrict s1, const char \*restrict s2);

64541 **DESCRIPTION**

64542 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
64543 conflict between the requirements described here and the ISO C standard is unintentional. This  
64544 volume of POSIX.1-2008 defers to the ISO C standard.

64545       The *strcat()* function shall append a copy of the string pointed to by *s2* (including the  
64546 terminating NUL character) to the end of the string pointed to by *s1*. The initial byte of *s2*  
64547 overwrites the NUL character at the end of *s1*. If copying takes place between objects that  
64548 overlap, the behavior is undefined.

64549 **RETURN VALUE**64550       The *strcat()* function shall return *s1*; no return value is reserved to indicate an error.64551 **ERRORS**

64552       No errors are defined.

64553 **EXAMPLES**

64554       None.

64555 **APPLICATION USAGE**

64556       This version is aligned with the ISO C standard; this does not affect compatibility with XPG3  
64557 applications. Reliable error detection by this function was never guaranteed.

64558 **RATIONALE**

64559       None.

64560 **FUTURE DIRECTIONS**

64561       None.

64562 **SEE ALSO**64563       [\*strncat\(\)\*](#)64564       XBD [\*<string.h>\*](#)64565 **CHANGE HISTORY**

64566       First released in Issue 1. Derived from Issue 1 of the SVID.

64567 **Issue 6**64568       The *strcat()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

64569 **NAME**

64570       strchr — string scanning operation

64571 **SYNOPSIS**

64572       #include &lt;string.h&gt;

64573       char \*strchr(const char \*s, int c);

64574 **DESCRIPTION**

64575 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
64576       conflict between the requirements described here and the ISO C standard is unintentional. This  
64577       volume of POSIX.1-2008 defers to the ISO C standard.

64578       The *strchr()* function shall locate the first occurrence of *c* (converted to a **char**) in the string  
64579       pointed to by *s*. The terminating NUL character is considered to be part of the string.

64580 **RETURN VALUE**

64581       Upon completion, *strchr()* shall return a pointer to the byte, or a null pointer if the byte was not  
64582       found.

64583 **ERRORS**

64584       No errors are defined.

64585 **EXAMPLES**

64586       None.

64587 **APPLICATION USAGE**

64588       None.

64589 **RATIONALE**

64590       None.

64591 **FUTURE DIRECTIONS**

64592       None.

64593 **SEE ALSO**64594       [strrchr\(\)](#)64595       XBD [<string.h>](#)64596 **CHANGE HISTORY**

64597       First released in Issue 1. Derived from Issue 1 of the SVID.

64598 **Issue 6**

64599       Extensions beyond the ISO C standard are marked.

## 64600 NAME

64601 strcmp — compare two strings

## 64602 SYNOPSIS

64603 #include &lt;string.h&gt;

64604 int strcmp(const char \*s1, const char \*s2);

## 64605 DESCRIPTION

64606 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
64607 conflict between the requirements described here and the ISO C standard is unintentional. This  
64608 volume of POSIX.1-2008 defers to the ISO C standard.

64609 The *strcmp()* function shall compare the string pointed to by *s1* to the string pointed to by *s2*.

64610 The sign of a non-zero return value shall be determined by the sign of the difference between the  
64611 values of the first pair of bytes (both interpreted as type **unsigned char**) that differ in the strings  
64612 being compared.

## 64613 RETURN VALUE

64614 Upon completion, *strcmp()* shall return an integer greater than, equal to, or less than 0, if the  
64615 string pointed to by *s1* is greater than, equal to, or less than the string pointed to by *s2*,  
64616 respectively.

## 64617 ERRORS

64618 No errors are defined.

## 64619 EXAMPLES

## 64620 Checking a Password Entry

64621 The following example compares the information read from standard input to the value of the  
64622 name of the user entry. If the *strcmp()* function returns 0 (indicating a match), a further check  
64623 will be made to see if the user entered the proper old password. The *crypt()* function shall  
64624 encrypt the old password entered by the user, using the value of the encrypted password in the  
64625 **passwd** structure as the salt. If this value matches the value of the encrypted **passwd** in the  
64626 structure, the entered password *oldpasswd* is the correct user's password. Finally, the program  
64627 encrypts the new password so that it can store the information in the **passwd** structure.

```
64628 #include <string.h>
64629 #include <unistd.h>
64630 #include <stdio.h>
64631 ...
64632 int valid_change;
64633 struct passwd *p;
64634 char user[100];
64635 char oldpasswd[100];
64636 char newpasswd[100];
64637 char savepasswd[100];
64638 ...
64639 if (strcmp(p->pw_name, user) == 0) {
64640     if (strcmp(p->pw_passwd, crypt(oldpasswd, p->pw_passwd)) == 0) {
64641         strcpy(savepasswd, crypt(newpasswd, user));
64642         p->pw_passwd = savepasswd;
64643         valid_change = 1;
64644     }
64645     else {
```

```
64646         fprintf(stderr, "Old password is not valid\n");
64647     }
64648 }
64649 ...
```

**64650 APPLICATION USAGE**

64651 None.

**64652 RATIONALE**

64653 None.

**64654 FUTURE DIRECTIONS**

64655 None.

**64656 SEE ALSO**

64657 [strncmp\(\)](#)

64658 XBD [<string.h>](#)

**64659 CHANGE HISTORY**

64660 First released in Issue 1. Derived from Issue 1 of the SVID.

**64661 Issue 6**

64662 Extensions beyond the ISO C standard are marked.

## 64663 NAME

64664 strcoll, strcoll\_l — string comparison using collating information

## 64665 SYNOPSIS

```
64666 #include <string.h>
64667 int strcoll(const char *s1, const char *s2);
64668 CX int strcoll_l(const char *s1, const char *s2,
64669 locale_t locale);
```

## 64670 DESCRIPTION

64671 CX For *strcoll()*: The functionality described on this reference page is aligned with the ISO C  
 64672 standard. Any conflict between the requirements described here and the ISO C standard is  
 64673 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

64674 CX The *strcoll()* and *strcoll\_l()* functions shall compare the string pointed to by *s1* to the string  
 64675 pointed to by *s2*, both interpreted as appropriate to the *LC\_COLLATE* category of the current  
 64676 CX locale, or of the locale represented by *locale*, respectively.

64677 CX The *strcoll()* and *strcoll\_l()* functions shall not change the setting of *errno* if successful.

64678 Since no return value is reserved to indicate an error, an application wishing to check for error  
 64679 CX situations should set *errno* to 0, then call *strcoll()*, or *strcoll\_l()* then check *errno*.

64680 CX The behavior is undefined if the *locale* argument to *strcoll\_l()* is the special locale object  
 64681 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

## 64682 RETURN VALUE

64683 Upon successful completion, *strcoll()* shall return an integer greater than, equal to, or less than 0,  
 64684 according to whether the string pointed to by *s1* is greater than, equal to, or less than the string  
 64685 CX pointed to by *s2* when both are interpreted as appropriate to the current locale. On error,  
 64686 *strcoll()* may set *errno*, but no return value is reserved to indicate an error.

64687 Upon successful completion, *strcoll\_l()* shall return an integer greater than, equal to, or less than  
 64688 0, according to whether the string pointed to by *s1* is greater than, equal to, or less than the  
 64689 string pointed to by *s2* when both are interpreted as appropriate to the locale represented by  
 64690 *locale*. On error, *strcoll\_l()* may set *errno*, but no return value is reserved to indicate an error.

## 64691 ERRORS

64692 These functions may fail if:

64693 CX [EINVAL] The *s1* or *s2* arguments contain characters outside the domain of the collating  
 64694 sequence.

## 64695 EXAMPLES

## 64696 Comparing Nodes

64697 The following example uses an application-defined function, *node\_compare()*, to compare two  
 64698 nodes based on an alphabetical ordering of the *string* field.

```
64699 #include <string.h>
64700 ...
64701 struct node { /* These are stored in the table. */
64702     char *string;
64703     int length;
64704 };
64705 ...
```

```

64706     int node_compare(const void *node1, const void *node2)
64707     {
64708         return strcoll(((const struct node *)node1)->string,
64709                     ((const struct node *)node2)->string);
64710     }
64711     ...

```

#### 64712 APPLICATION USAGE

64713 The *strxfrm()* and *strcmp()* functions should be used for sorting large lists.

#### 64714 RATIONALE

64715 None.

#### 64716 FUTURE DIRECTIONS

64717 None.

#### 64718 SEE ALSO

64719 *alphasort()*, *strcmp()*, *strxfrm()*

64720 XBD <string.h>

#### 64721 CHANGE HISTORY

64722 First released in Issue 3.

#### 64723 Issue 5

64724 The DESCRIPTION is updated to indicate that *errno* does not change if the function is successful.

#### 64725 Issue 6

64726 Extensions beyond the ISO C standard are marked.

64727 The following new requirements on POSIX implementations derive from alignment with the  
64728 Single UNIX Specification:

- 64729 • The [EINVAL] optional error condition is added.

64730 An example is added.

#### 64731 Issue 7

64732 The *strcoll\_l()* function is added from The Open Group Technical Standard, 2006, Extended API  
64733 Set Part 4.

64734 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0593 [283] and XSH/TC1-2008/0594  
64735 [283] are applied.

**NAME**

strcpy, strcpy — copy a string and return a pointer to the end of the result

**SYNOPSIS**

```
#include <string.h>
```

```
char *stpcpy(char *restrict s1, const char *restrict s2);
char *strcpy(char *restrict s1, const char *restrict s2);
```

**DESCRIPTION**

For *stpcpy()*: The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *stpcpy()* and *strcpy()* functions shall copy the string pointed to by *s2* (including the terminating NUL character) into the array pointed to by *s1*.

If copying takes place between objects that overlap, the behavior is undefined.

**RETURN VALUE**

The *stpcpy()* function shall return a pointer to the terminating NUL character copied into the *s1* buffer.

The *strcpy()* function shall return *s1*.

No return values are reserved to indicate an error.

**ERRORS**

No errors are defined.

**EXAMPLES****Construction of a Multi-Part Message in a Single Buffer**

```
#include <string.h>
#include <stdio.h>

int
main (void)
{
    char buffer [10];
    char *name = buffer;

    name = stpcpy (stpcpy (stpcpy (name, "ice"), "-"), "cream");
    puts (buffer);
    return 0;
}
```

**Initializing a String**

The following example copies the string "-----" into the *permstring* variable.

```
#include <string.h>
...
static char permstring[11];
...
strcpy(permstring, "-----");
...
```



**Storing a Key and Data**

The following example allocates space for a key using *malloc()* then uses *strcpy()* to place the key there. Then it allocates space for data using *malloc()*, and uses *strcpy()* to place data there. (The user-defined function *dbfree()* frees memory previously allocated to an array of type **struct element** \*.)

```
#include <string.h>
#include <stdlib.h>
#include <stdio.h>
...
/* Structure used to read data and store it. */
struct element {
    char *key;
    char *data;
};

struct element *tbl, *curtbl;
char *key, *data;
int count;
...
void dbfree(struct element *, int);
...
if ((curtbl->key = malloc(strlen(key) + 1)) == NULL) {
    perror("malloc"); dbfree(tbl, count); return NULL;
}
strcpy(curtbl->key, key);

if ((curtbl->data = malloc(strlen(data) + 1)) == NULL) {
    perror("malloc"); free(curtbl->key); dbfree(tbl, count); return NULL;
}
strcpy(curtbl->data, data);
...
```

**APPLICATION USAGE**

Character movement is performed differently in different implementations. Thus, overlapping moves may yield surprises.

This version is aligned with the ISO C standard; this does not affect compatibility with XPG3 applications. Reliable error detection by this function was never guaranteed.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*strncpy()*, *wcscpy()*

XBD **<string.h>**

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

64820 **Issue 6**

64821 The *strcpy()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

64822 **Issue 7**

64823 The *strcpy()* function is added from The Open Group Technical Standard, 2006, Extended API  
64824 Set Part 1.

64825 **NAME**

64826 strcspn — get the length of a complementary substring

64827 **SYNOPSIS**

64828 #include &lt;string.h&gt;

64829 size\_t strcspn(const char \*s1, const char \*s2);

64830 **DESCRIPTION**

64831 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 64832 conflict between the requirements described here and the ISO C standard is unintentional. This  
 64833 volume of POSIX.1-2008 defers to the ISO C standard.

64834 The *strcspn()* function shall compute the length (in bytes) of the maximum initial segment of the  
 64835 string pointed to by *s1* which consists entirely of bytes *not* from the string pointed to by *s2*.

64836 **RETURN VALUE**

64837 The *strcspn()* function shall return the length of the computed segment of the string pointed to  
 64838 by *s1*; no return value is reserved to indicate an error.

64839 **ERRORS**

64840 No errors are defined.

64841 **EXAMPLES**

64842 None.

64843 **APPLICATION USAGE**

64844 None.

64845 **RATIONALE**

64846 None.

64847 **FUTURE DIRECTIONS**

64848 None.

64849 **SEE ALSO**64850 *strspn()*

64851 XBD &lt;string.h&gt;

64852 **CHANGE HISTORY**

64853 First released in Issue 1. Derived from Issue 1 of the SVID.

64854 **Issue 5**

64855 The RETURN VALUE section is updated to indicate that *strcspn()* returns the length of *s1*, and  
 64856 not *s1* itself as was previously stated.

64857 **Issue 6**

64858 The Open Group Corrigendum U030/1 is applied. The text of the RETURN VALUE section is  
 64859 updated to indicate that the computed segment length is returned, not the *s1* length.

**NAME**

strdup, strndup — duplicate a specific number of bytes from a string

**SYNOPSIS**

```
CX      #include <string.h>
char *strdup(const char *s);
char *strndup(const char *s, size_t size);
```

**DESCRIPTION**

The *strdup()* function shall return a pointer to a new string, which is a duplicate of the string pointed to by *s*. The returned pointer can be passed to *free()*. A null pointer is returned if the new string cannot be created.

The *strndup()* function shall be equivalent to the *strdup()* function, duplicating the provided *s* in a new block of memory allocated as if by using *malloc()*, with the exception being that *strndup()* copies at most *size* plus one bytes into the newly allocated memory, terminating the new string with a NUL character. If the length of *s* is larger than *size*, only *size* bytes shall be duplicated. If *size* is larger than the length of *s*, all bytes in *s* shall be copied into the new memory buffer, including the terminating NUL character. The newly created string shall always be properly terminated.

**RETURN VALUE**

The *strdup()* function shall return a pointer to a new string on success. Otherwise, it shall return a null pointer and set *errno* to indicate the error.

Upon successful completion, the *strndup()* function shall return a pointer to the newly allocated memory containing the duplicated string. Otherwise, it shall return a null pointer and set *errno* to indicate the error.

**ERRORS**

These functions shall fail if:

[ENOMEM] Storage space available is insufficient.

**EXAMPLES**

None.

**APPLICATION USAGE**

For functions that allocate memory as if by *malloc()*, the application should release such memory when it is no longer required by a call to *free()*. For *strdup()* and *strndup()*, this is the return value.

Implementations are free to *malloc()* a buffer containing either (*size* + 1) bytes or (*strlen( s, size)* + 1) bytes. Applications should not assume that *strndup()* will allocate (*size* + 1) bytes when *strlen( s)* is smaller than *size*.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*free()*, *wcsdup()*

XBD <string.h>

64902 **CHANGE HISTORY**

64903 First released in Issue 4, Version 2.

64904 **Issue 5**

64905 Moved from X/OPEN UNIX extension to BASE.

64906 **Issue 7**64907 Austin Group Interpretation 1003.1-2001 #044 is applied, changing the “may fail” [ENOMEM]  
64908 error to become a “shall fail” error.64909 The *strdup()* function is moved from the XSI option to the Base.64910 The *strndup()* function is added from The Open Group Technical Standard, 2006, Extended API  
64911 Set Part 1.64912 The APPLICATION USAGE section is updated to clarify that memory is allocated as if by  
64913 *malloc()*.

## 64914 NAME

64915       strerror, strerror\_l, strerror\_r — get error message string

## 64916 SYNOPSIS

64917       #include &lt;string.h&gt;

64918       char \*strerror(int errnum);

64919 CX       char \*strerror\_l(int errnum, locale\_t locale);

64920       int strerror\_r(int errnum, char \*strerrbuf, size\_t buflen);

## 64921 DESCRIPTION

64922 CX       For *strerror()*: The functionality described on this reference page is aligned with the ISO C  
 64923 standard. Any conflict between the requirements described here and the ISO C standard is  
 64924 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

64925       The *strerror()* function shall map the error number in *errnum* to a locale-dependent error  
 64926 message string and shall return a pointer to it. Typically, the values for *errnum* come from *errno*,  
 64927 but *strerror()* shall map any value of type *int* to a message.

64928 CX       The application shall not modify the string returned. The returned string pointer might be  
 64929 CX       invalidated or the string content might be overwritten by a subsequent call to *strerror()*, or by a  
 64930 subsequent call to *strerror\_l()* in the same thread. The returned pointer and the string content  
 64931 might also be invalidated if the calling thread is terminated.

64932 CX       The string may be overwritten by a subsequent call to *strerror\_l()* in the same thread.

64933       The contents of the error message strings returned by *strerror()* should be determined by the  
 64934 setting of the *LC\_MESSAGES* category in the current locale.

64935       The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls  
 64936 *strerror()*.

64937 CX       The *strerror()* and *strerror\_l()* functions shall not change the setting of *errno* if successful.

64938       Since no return value is reserved to indicate an error of *strerror()*, an application wishing to  
 64939 check for error situations should set *errno* to 0, then call *strerror()*, then check *errno*. Similarly,  
 64940 since *strerror\_l()* is required to return a string for some errors, an application wishing to check  
 64941 for all error situations should set *errno* to 0, then call *strerror\_l()*, then check *errno*.

64942       The *strerror()* function need not be thread-safe.

64943       The *strerror\_l()* function shall map the error number in *errnum* to a locale-dependent error  
 64944 message string in the locale represented by *locale* and shall return a pointer to it.

64945       The *strerror\_r()* function shall map the error number in *errnum* to a locale-dependent error  
 64946 message string and shall return the string in the buffer pointed to by *strerrbuf*, with length  
 64947 *buflen*.

64948 CX       If the value of *errnum* is a valid error number, the message string shall indicate what error  
 64949 occurred; if the value of *errnum* is zero, the message string shall either be an empty string or  
 64950 indicate that no error occurred; otherwise, if these functions complete successfully, the message  
 64951 string shall indicate that an unknown error occurred.

64952 CX       The behavior is undefined if the *locale* argument to *strerror\_l()* is the special locale object  
 64953 *LC\_GLOBAL\_LOCALE* or is not a valid locale object handle.

**RETURN VALUE**

Upon completion, whether successful or not, *strerror()* shall return a pointer to the generated message string. On error *errno* may be set, but no return value is reserved to indicate an error.

Upon successful completion, *strerror\_l()* shall return a pointer to the generated message string. If *errnum* is not a valid error number, *errno* may be set to [EINVAL], but a pointer to a message string shall still be returned. If any other error occurs, *errno* shall be set to indicate the error and a null pointer shall be returned.

Upon successful completion, *strerror\_r()* shall return 0. Otherwise, an error number shall be returned to indicate the error.

**ERRORS**

These functions may fail if:

[EINVAL] The value of *errnum* is neither a valid error number nor zero.

The *strerror\_r()* function may fail if:

[ERANGE] Insufficient storage was supplied via *strerrbuf* and *buflen* to contain the generated message string.

**EXAMPLES**

None.

**APPLICATION USAGE**

Historically in some implementations, calls to *perror()* would overwrite the string that the pointer returned by *strerror()* points to. Such implementations did not conform to the ISO C standard; however, application developers should be aware of this behavior if they wish their applications to be portable to such implementations.

**RATIONALE**

The *strerror\_l()* function is required to be thread-safe, thereby eliminating the need for an equivalent to the *strerror\_r()* function.

Earlier versions of this standard did not explicitly require that the error message strings returned by *strerror()* and *strerror\_r()* provide any information about the error. This version of the standard requires a meaningful message for any successful completion.

Since no return value is reserved to indicate a *strerror()* error, but all calls (whether successful or not) must return a pointer to a message string, on error *strerror()* can return a pointer to an empty string or a pointer to a meaningful string that can be printed.

Note that the [EINVAL] error condition is a may fail error. If an invalid error number is supplied as the value of *errnum*, applications should be prepared to handle any of the following:

1. Error (with no meaningful message): *errno* is set to [EINVAL], the return value is a pointer to an empty string.
2. Successful completion: *errno* is unchanged and the return value points to a string like "unknown error" or "error number xxx" (where xxx is the value of *errnum*).
3. Combination of #1 and #2: *errno* is set to [EINVAL] and the return value points to a string like "unknown error" or "error number xxx" (where xxx is the value of *errnum*). Since applications frequently use the return value of *strerror()* as an argument to functions like *fprintf()* (without checking the return value) and since applications have no way to parse an error message string to determine whether *errnum* represents a valid error number, implementations are encouraged to implement #3. Similarly, implementations are encouraged to have *strerror\_r()* return [EINVAL] and put a string

64998 like "unknown error" or "error number xxx" in the buffer pointed to by *strerrbuf*  
64999 when the value of *errnum* is not a valid error number.

65000 Some applications rely on being able to set *errno* to 0 before calling a function with no reserved  
65001 value to indicate an error, then call *strerror(errno)* afterwards to detect whether an error occurred  
65002 (because *errno* changed) or to indicate success (because *errno* remained zero). This usage pattern  
65003 requires that *strerror(0)* succeed with useful results. Previous versions of the standard did not  
65004 specify the behavior when *errnum* is zero.

#### 65005 FUTURE DIRECTIONS

65006 None.

#### 65007 SEE ALSO

65008 *perror()*

65009 XBD <string.h>

#### 65010 CHANGE HISTORY

65011 First released in Issue 3.

#### 65012 Issue 5

65013 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

65014 A note indicating that the *strerror()* function need not be reentrant is added to the  
65015 DESCRIPTION.

#### 65016 Issue 6

65017 Extensions beyond the ISO C standard are marked.

65018 The following new requirements on POSIX implementations derive from alignment with the  
65019 Single UNIX Specification:

- 65020 • In the RETURN VALUE section, the fact that *errno* may be set is added.
- 65021 • The [EINVAL] optional error condition is added.

65022 The normative text is updated to avoid use of the term “must” for application requirements.

65023 The *strerror\_r()* function is added in response to IEEE PASC Interpretation 1003.1c #39.

65024 The *strerror\_r()* function is marked as part of the Thread-Safe Functions option.

#### 65025 Issue 7

65026 Austin Group Interpretation 1003.1-2001 #072 is applied, updating the ERRORS section.

65027 Austin Group Interpretation 1003.1-2001 #156 is applied.

65028 Austin Group Interpretation 1003.1-2001 #187 is applied, clarifying the behavior when the  
65029 generated error message is an empty string.

65030 SD5-XSH-ERN-191 is applied, updating the APPLICATION USAGE section.

65031 The *strerror\_l()* function is added from The Open Group Technical Standard, 2006, Extended API  
65032 Set Part 4.

65033 The *strerror\_r()* function is moved from the Thread-Safe Functions option to the Base.

65034 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0595 [75], XSH/TC1-2008/0596 [447],  
65035 XSH/TC1-2008/0597 [382,428], XSH/TC1-2008/0598 [283], XSH/TC1-2008/0599 [382,428],  
65036 XSH/TC1-2008/0600 [283], and XSH/TC1-2008/0601 [382,428] are applied.



**NAME**

strfmon, strfmon\_l — convert monetary value to a string

**SYNOPSIS**

```
#include <monetary.h>

ssize_t strfmon(char *restrict s, size_t maxsize,
               const char *restrict format, ...);
ssize_t strfmon_l(char *restrict s, size_t maxsize,
                 locale_t locale, const char *restrict format, ...);
```

**DESCRIPTION**

The *strfmon()* function shall place characters into the array pointed to by *s* as controlled by the string pointed to by *format*. No more than *maxsize* bytes are placed into the array.

The format is a character string, beginning and ending in its initial state, if any, that contains two types of objects: *plain characters*, which are simply copied to the output stream, and *conversion specifications*, each of which shall result in the fetching of zero or more arguments which are converted and formatted. The results are undefined if there are insufficient arguments for the format. If the format is exhausted while arguments remain, the excess arguments are simply ignored.

The application shall ensure that a conversion specification consists of the following sequence:

- A '%' character
- Optional flags
- Optional field width
- Optional left precision
- Optional right precision
- A required conversion specifier character that determines the conversion to be performed

The *strfmon\_l()* function shall be equivalent to the *strfmon()* function, except that the locale data used is from the locale represented by *locale*.

**Flags**

One or more of the following optional flags can be specified to control the conversion:

- =f** An '=' followed by a single character *f* which is used as the numeric fill character. In order to work with precision or width counts, the fill character shall be a single byte character; if not, the behavior is undefined. The default numeric fill character is the <space>. This flag does not affect field width filling which always uses the <space>. This flag is ignored unless a left precision (see below) is specified.
- ^** Do not format the currency amount with grouping characters. The default is to insert the grouping characters if defined for the current locale.
- + or (** Specify the style of representing positive and negative currency amounts. Only one of '+' or '(' may be specified. If '+' is specified, the locale's equivalent of '+' and '-' are used (for example, in many locales, the empty string if positive and '-' if negative). If '(' is specified, negative amounts are enclosed within parentheses. If neither flag is specified, the '+' style is used.
- !** Suppress the currency symbol from the output conversion.

65078 – Specify the alignment. If this flag is present the result of the conversion is left-justified  
 65079 (padded to the right) rather than right-justified. This flag shall be ignored unless a field  
 65080 width (see below) is specified.

#### 65081 Field Width

65082 *w* A decimal digit string *w* specifying a minimum field width in bytes in which the result  
 65083 of the conversion is right-justified (or left-justified if the flag '*-*' is specified). The  
 65084 default is 0.

#### 65085 Left Precision

65086 *#n* A '*#*' followed by a decimal digit string *n* specifying a maximum number of digits  
 65087 expected to be formatted to the left of the radix character. This option can be used to  
 65088 keep the formatted output from multiple calls to the *strfmon()* function aligned in the  
 65089 same columns. It can also be used to fill unused positions with a special character as in  
 65090 "\$\*\*\*123.45". This option causes an amount to be formatted as if it has the number  
 65091 of digits specified by *n*. If more than *n* digit positions are required, this conversion  
 65092 specification is ignored. Digit positions in excess of those actually required are filled  
 65093 with the numeric fill character (see the *=f* flag above).

65094 If grouping has not been suppressed with the '*^*' flag, and it is defined for the current  
 65095 locale, grouping separators are inserted before the fill characters (if any) are added.  
 65096 Grouping separators are not applied to fill characters even if the fill character is a digit.

65097 To ensure alignment, any characters appearing before or after the number in the  
 65098 formatted output such as currency or sign symbols are padded as necessary with  
 65099 <space> characters to make their positive and negative formats an equal length.

#### 65100 Right Precision

65101 *.p* A <period> followed by a decimal digit string *p* specifying the number of digits after  
 65102 the radix character. If the value of the right precision *p* is 0, no radix character appears.  
 65103 If a right precision is not included, a default specified by the current locale is used. The  
 65104 amount being formatted is rounded to the specified number of digits prior to  
 65105 formatting.

#### 65106 Conversion Specifier Characters

65107 The conversion specifier characters and their meanings are:

65108 *i* The **double** argument is formatted according to the locale's international currency  
 65109 format (for example, in the US: USD 1,234.56). If the argument is  $\pm\text{Inf}$  or NaN, the result  
 65110 of the conversion is unspecified.

65111 *n* The **double** argument is formatted according to the locale's national currency format  
 65112 (for example, in the US: \$1,234.56). If the argument is  $\pm\text{Inf}$  or NaN, the result of the  
 65113 conversion is unspecified.

65114 *%* Convert to a '*%*'; no argument is converted. The entire conversion specification shall  
 65115 be *%%*.

**Locale Information**

The *LC\_MONETARY* category of the current locale affects the behavior of this function including the monetary radix character (which may be different from the numeric radix character affected by the *LC\_NUMERIC* category), the grouping separator, the currency symbols, and formats. The international currency symbol should be conformant with the ISO 4217:2001 standard.

If the value of *maxsize* is greater than `{SSIZE_MAX}`, the result is implementation-defined.

The behavior is undefined if the *locale* argument to *strfmon\_l()* is the special locale object *LC\_GLOBAL\_LOCALE* or is not a valid locale object handle.

**RETURN VALUE**

If the total number of resulting bytes including the terminating null byte is not more than *maxsize*, these functions shall return the number of bytes placed into the array pointed to by *s*, not including the terminating NUL character. Otherwise, `-1` shall be returned, the contents of the array are unspecified, and *errno* shall be set to indicate the error.

**ERRORS**

These functions shall fail if:

[E2BIG] Conversion stopped due to lack of space in the buffer.

**EXAMPLES**

Given a locale for the US and the values 123.45, `-123.45`, and `3456.781`, the following output might be produced. Square brackets ("`[ ]`") are used in this example to delimit the output.

|                      |                                                                                                         |                                                                           |
|----------------------|---------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------|
| <code>%n</code>      | <code>[ \$123.45 ]</code><br><code>[ -\$123.45 ]</code><br><code>[ \$3,456.78 ]</code>                  | Default formatting                                                        |
| <code>%11n</code>    | <code>[     \$123.45 ]</code><br><code>[    -\$123.45 ]</code><br><code>[   \$3,456.78 ]</code>         | Right align within an 11-character field                                  |
| <code>%#5n</code>    | <code>[ \$    123.45 ]</code><br><code>[ -\$    123.45 ]</code><br><code>[ \$ 3,456.78 ]</code>         | Aligned columns for values up to 99 999                                   |
| <code>%=*#5n</code>  | <code>[ \$***123.45 ]</code><br><code>[ -\$***123.45 ]</code><br><code>[ \$*3,456.78 ]</code>           | Specify a fill character                                                  |
| <code>%=0#5n</code>  | <code>[ \$000123.45 ]</code><br><code>[ -\$000123.45 ]</code><br><code>[ \$03,456.78 ]</code>           | Fill characters do not use grouping even if the fill character is a digit |
| <code>%^#5n</code>   | <code>[ \$    123.45 ]</code><br><code>[ -\$    123.45 ]</code><br><code>[ \$ 3456.78 ]</code>          | Disable the grouping separator                                            |
| <code>%^#5.0n</code> | <code>[ \$    123 ]</code><br><code>[ -\$    123 ]</code><br><code>[ \$    3457 ]</code>                | Round off to whole units                                                  |
| <code>%^#5.4n</code> | <code>[ \$    123.4500 ]</code><br><code>[ -\$    123.4500 ]</code><br><code>[ \$    3456.7810 ]</code> | Increase the precision                                                    |
| <code>%(#5n</code>   | <code>[ \$    123.45 ]</code>                                                                           | Use an alternative pos/neg style                                          |

```

65160          [ ( $    123.45 ) ]
65161          [ $ 3,456.78 ]

65162      %! ( #5n    [    123.45 ]      Disable the currency symbol
65163          [ (    123.45 ) ]
65164          [ 3,456.78 ]

65165      %-14#5.4n [ $    123.4500 ]      Left-justify the output
65166          [ -$    123.4500 ]
65167          [ $ 3,456.7810 ]

65168      %14#5.4n  [ $    123.4500 ]      Corresponding right-justified output
65169          [ -$    123.4500 ]
65170          [ $ 3,456.7810 ]

```

65171 See also the EXAMPLES section in *fprintf()*.

## 65172 APPLICATION USAGE

65173 None.

## 65174 RATIONALE

65175 None.

## 65176 FUTURE DIRECTIONS

65177 Lowercase conversion characters are reserved for future standards use and uppercase for  
65178 implementation-defined use.

## 65179 SEE ALSO

65180 *fprintf()*, *localeconv()*

65181 XBD <monetary.h>

## 65182 CHANGE HISTORY

65183 First released in Issue 4.

### 65184 Issue 5

65185 Moved from ENHANCED I18N to BASE.

65186 The [ENOSYS] error is removed.

65187 Text is added to the DESCRIPTION warning about values of *maxsize* that are greater than  
65188 {SSIZE\_MAX}.

### 65189 Issue 6

65190 The normative text is updated to avoid use of the term “must” for application requirements.

65191 The **restrict** keyword is added to the *strfmon()* prototype for alignment with the  
65192 ISO/IEC 9899:1999 standard.

65193 The EXAMPLES section is reworked, clarifying the output format.

### 65194 Issue 7

65195 SD5-XSH-ERN-29 is applied, updating the examples for % ( #5n and % ! ( #5n.

65196 SD5-XSH-ERN-233 is applied, changing the definition of the '+' or '(' flags to refer to  
65197 multiple locales.

65198 The *strfmon()* function is moved from the XSI option to the Base.

65199  
65200

The *strfmon\_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

65201  
65202

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0602 [302], XSH/TC1-2008/0603 [283], and XSH/TC1-2008/0604 [283] are applied.

65203 **NAME**65204 `strptime`, `strptime_l` — convert date and time to a string65205 **SYNOPSIS**

```

65206     #include <time.h>
65207     size_t strptime(char *restrict s, size_t maxsize,
65208                   const char *restrict format, const struct tm *restrict timeptr);
65209 CX    size_t strptime_l(char *restrict s, size_t maxsize,
65210                   const char *restrict format, const struct tm *restrict timeptr,
65211                   locale_t locale);

```

65212 **DESCRIPTION**

65213 CX For `strptime()`: The functionality described on this reference page is aligned with the ISO C  
 65214 standard. Any conflict between the requirements described here and the ISO C standard is  
 65215 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

65216 The `strptime()` function shall place bytes into the array pointed to by `s` as controlled by the string  
 65217 pointed to by `format`. The format is a character string, beginning and ending in its initial shift  
 65218 state, if any. The format string consists of zero or more conversion specifications and ordinary  
 65219 characters.

65220 Each conversion specification is introduced by the ‘%’ character after which the following  
 65221 appear in sequence:

- 65222 CX
- An optional flag:
    - 65223 0 The zero character (‘0’), which specifies that the character used as the padding  
 65224 character is ‘0’.
    - 65225 + The <plus-sign> character (‘+’), which specifies that the character used as the  
 65226 padding character is ‘0’, and that if and only if the field being produced consumes  
 65227 more than four bytes to represent a year (for %F, %G, or %Y) or more than two bytes to  
 65228 represent the year divided by 100 (for %C) then a leading <plus-sign> character shall  
 65229 be included if the year being processed is greater than or equal to zero or a leading  
 65230 <hyphen-minus> character (‘-’) shall be included if the year is less than zero.
  - 65231 The default padding character is unspecified.
  - An optional minimum field width. If the converted value, including any leading ‘+’ or  
 65232 ‘-’ sign, has fewer bytes than the minimum field width and the padding character is not  
 65233 the NUL character, the output shall be padded on the left (after any leading ‘+’ or ‘-’  
 65234 sign) with the padding character.
  - 65235
  - An optional E or O modifier.
  - 65236
  - A terminating conversion specifier character that indicates the type of conversion to be  
 65237 applied.
  - 65238

65239 CX The results are unspecified if more than one flag character is specified, a flag character is  
 65240 specified without a minimum field width; a minimum field width is specified without a flag  
 65241 character; a modifier is specified with a flag or with a minimum field width; or if a minimum  
 65242 field width is specified for any conversion specifier other than C, F, G, or Y.

65243 All ordinary characters (including the terminating NUL character) are copied unchanged into  
 65244 the array. If copying takes place between objects that overlap, the behavior is undefined. No  
 65245 more than `maxsize` bytes are placed into the array. Each conversion specifier is replaced by  
 65246 appropriate characters as described in the following list. The appropriate characters are

determined using the *LC\_TIME* category of the current locale and by the values of zero or more members of the broken-down time structure pointed to by *timeptr*, as specified in brackets in the description. If any of the specified values are outside the normal range, the characters stored are unspecified.

65251 CX The *strptime\_l()* function shall be equivalent to the *strptime()* function, except that the locale data used is from the locale represented by *locale*.

65252

65253 Local timezone information is used as though *strptime()* called *tzset()*.

65254 The following conversion specifiers shall be supported:

65255 a Replaced by the locale's abbreviated weekday name. [*tm\_wday*]

65256 A Replaced by the locale's full weekday name. [*tm\_wday*]

65257 b Replaced by the locale's abbreviated month name. [*tm\_mon*]

65258 B Replaced by the locale's full month name. [*tm\_mon*]

65259 c Replaced by the locale's appropriate date and time representation. (See the Base Definitions volume of POSIX.1-2008, <**time.h**>.)

65260

65261 C Replaced by the year divided by 100 and truncated to an integer, as a decimal number.

65262 [*tm\_year*]

65263 If a minimum field width is not specified, the number of characters placed into the

65264 array pointed to by *s* will be the number of digits in the year divided by 100 or two,

65265 CX whichever is greater. If a minimum field width is specified, the number of characters

65266 placed into the array pointed to by *s* will be the number of digits in the year divided by

65267 100 or the minimum field width, whichever is greater.

65268 d Replaced by the day of the month as a decimal number [01,31]. [*tm\_mday*]

65269 D Equivalent to %m/%d/%Y. [*tm\_mon*, *tm\_mday*, *tm\_year*]

65270 e Replaced by the day of the month as a decimal number [1,31]; a single digit is preceded

65271 by a space. [*tm\_mday*]

65272 CX F Equivalent to %<sup>+4</sup>Y-%m-%d if no flag and no minimum field width are specified.

65273 [*tm\_year*, *tm\_mon*, *tm\_mday*]

65274 CX If a minimum field width of *x* is specified, the year shall be output as if by the Y

65275 specifier (described below) with whatever flag was given and a minimum field width

65276 of *x*−6. If *x* is less than 6, the behavior shall be as if *x* equalled 6.

65277 If the minimum field width is specified to be 10, and the year is four digits long, then

65278 the output string produced will match the ISO 8601:2004 standard subclause 4.1.2.2

65279 complete representation, extended format date representation of a specific day. If a +

65280 flag is specified, a minimum field width of *x* is specified, and *x*−7 bytes are sufficient to

65281 hold the digits of the year (not including any needed sign character), then the output

65282 will match the ISO 8601:2004 standard subclause 4.1.2.4 complete representation,

65283 expanded format date representation of a specific day.

65284 g Replaced by the last 2 digits of the week-based year (see below) as a decimal number

65285 [00,99]. [*tm\_year*, *tm\_wday*, *tm\_yday*]

65286 G Replaced by the week-based year (see below) as a decimal number (for example, 1977).

65287 [*tm\_year*, *tm\_wday*, *tm\_yday*]



|       |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|-------|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 65288 | CX | If a minimum field width is specified, the number of characters placed into the array pointed to by <i>s</i> will be the number of digits and leading sign characters (if any) in the year, or the minimum field width, whichever is greater.                                                                                                                                                                                    |
| 65289 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65290 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65291 | h  | Equivalent to %b. [ <i>tm_mon</i> ]                                                                                                                                                                                                                                                                                                                                                                                              |
| 65292 | H  | Replaced by the hour (24-hour clock) as a decimal number [00,23]. [ <i>tm_hour</i> ]                                                                                                                                                                                                                                                                                                                                             |
| 65293 | I  | Replaced by the hour (12-hour clock) as a decimal number [01,12]. [ <i>tm_hour</i> ]                                                                                                                                                                                                                                                                                                                                             |
| 65294 | j  | Replaced by the day of the year as a decimal number [001,366]. [ <i>tm_yday</i> ]                                                                                                                                                                                                                                                                                                                                                |
| 65295 | m  | Replaced by the month as a decimal number [01,12]. [ <i>tm_mon</i> ]                                                                                                                                                                                                                                                                                                                                                             |
| 65296 | M  | Replaced by the minute as a decimal number [00,59]. [ <i>tm_min</i> ]                                                                                                                                                                                                                                                                                                                                                            |
| 65297 | n  | Replaced by a <newline>.                                                                                                                                                                                                                                                                                                                                                                                                         |
| 65298 | p  | Replaced by the locale's equivalent of either a.m. or p.m. [ <i>tm_hour</i> ]                                                                                                                                                                                                                                                                                                                                                    |
| 65299 | CX | Replaced by the time in a.m. and p.m. notation; in the POSIX locale this shall be equivalent to %I:%M:%S %p. [ <i>tm_hour</i> , <i>tm_min</i> , <i>tm_sec</i> ]                                                                                                                                                                                                                                                                  |
| 65300 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65301 | R  | Replaced by the time in 24-hour notation (%H:%M). [ <i>tm_hour</i> , <i>tm_min</i> ]                                                                                                                                                                                                                                                                                                                                             |
| 65302 | S  | Replaced by the second as a decimal number [00,60]. [ <i>tm_sec</i> ]                                                                                                                                                                                                                                                                                                                                                            |
| 65303 | t  | Replaced by a <tab>.                                                                                                                                                                                                                                                                                                                                                                                                             |
| 65304 | T  | Replaced by the time (%H:%M:%S). [ <i>tm_hour</i> , <i>tm_min</i> , <i>tm_sec</i> ]                                                                                                                                                                                                                                                                                                                                              |
| 65305 | u  | Replaced by the weekday as a decimal number [1,7], with 1 representing Monday. [ <i>tm_wday</i> ]                                                                                                                                                                                                                                                                                                                                |
| 65306 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65307 | U  | Replaced by the week number of the year as a decimal number [00,53]. The first Sunday of January is the first day of week 1; days in the new year before this are in week 0. [ <i>tm_year</i> , <i>tm_wday</i> , <i>tm_yday</i> ]                                                                                                                                                                                                |
| 65308 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65309 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65310 | V  | Replaced by the week number of the year (Monday as the first day of the week) as a decimal number [01,53]. If the week containing 1 January has four or more days in the new year, then it is considered week 1. Otherwise, it is the last week of the previous year, and the next week is week 1. Both January 4th and the first Thursday of January are always in week 1. [ <i>tm_year</i> , <i>tm_wday</i> , <i>tm_yday</i> ] |
| 65311 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65312 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65313 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65314 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65315 | w  | Replaced by the weekday as a decimal number [0,6], with 0 representing Sunday. [ <i>tm_wday</i> ]                                                                                                                                                                                                                                                                                                                                |
| 65316 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65317 | W  | Replaced by the week number of the year as a decimal number [00,53]. The first Monday of January is the first day of week 1; days in the new year before this are in week 0. [ <i>tm_year</i> , <i>tm_wday</i> , <i>tm_yday</i> ]                                                                                                                                                                                                |
| 65318 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65319 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65320 | x  | Replaced by the locale's appropriate date representation. (See the Base Definitions volume of POSIX.1-2008, <time.h>.)                                                                                                                                                                                                                                                                                                           |
| 65321 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65322 | X  | Replaced by the locale's appropriate time representation. (See the Base Definitions volume of POSIX.1-2008, <time.h>.)                                                                                                                                                                                                                                                                                                           |
| 65323 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 65324 | y  | Replaced by the last two digits of the year as a decimal number [00,99]. [ <i>tm_year</i> ]                                                                                                                                                                                                                                                                                                                                      |
| 65325 | Y  | Replaced by the year as a decimal number (for example, 1997). [ <i>tm_year</i> ]                                                                                                                                                                                                                                                                                                                                                 |
| 65326 | CX | If a minimum field width is specified, the number of characters placed into the array pointed to by <i>s</i> will be the number of digits and leading sign characters (if any) in the                                                                                                                                                                                                                                            |
| 65327 |    |                                                                                                                                                                                                                                                                                                                                                                                                                                  |



|       |     |                                                                                                                                                                 |
|-------|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 65328 |     | year, or the minimum field width, whichever is greater.                                                                                                         |
| 65329 | z   | Replaced by the offset from UTC in the ISO 8601:2004 standard format (+hhmm or -hhmm), or by no characters if no timezone is determinable. For example, "-0430" |
| 65330 |     | means 4 hours 30 minutes behind UTC (west of Greenwich). If <i>tm_isdst</i> is zero, the                                                                        |
| 65331 | CX  | standard time offset is used. If <i>tm_isdst</i> is greater than zero, the daylight savings time                                                                |
| 65332 |     | offset is used. If <i>tm_isdst</i> is negative, no characters are returned. [ <i>tm_isdst</i> ]                                                                 |
| 65333 |     |                                                                                                                                                                 |
| 65334 | Z   | Replaced by the timezone name or abbreviation, or by no bytes if no timezone                                                                                    |
| 65335 |     | information exists. [ <i>tm_isdst</i> ]                                                                                                                         |
| 65336 | %   | Replaced by %.                                                                                                                                                  |
| 65337 |     | If a conversion specification does not correspond to any of the above, the behavior is undefined.                                                               |
| 65338 | CX  | If a <b>struct tm</b> broken-down time structure is created by <i>localtime()</i> or <i>localtime_r()</i> , or modified                                         |
| 65339 |     | by <i>mktime()</i> , and the value of <i>TZ</i> is subsequently modified, the results of the %Z and %z                                                          |
| 65340 |     | <i>strptime()</i> conversion specifiers are undefined, when <i>strptime()</i> is called with such a broken-down                                                 |
| 65341 |     | time structure.                                                                                                                                                 |
| 65342 |     | If a <b>struct tm</b> broken-down time structure is created or modified by <i>gmtime()</i> or <i>gmtime_r()</i> , it is                                         |
| 65343 |     | unspecified whether the result of the %Z and %z conversion specifiers shall refer to UTC or the                                                                 |
| 65344 |     | current local timezone, when <i>strptime()</i> is called with such a broken-down time structure.                                                                |
| 65345 |     | <b>Modified Conversion Specifiers</b>                                                                                                                           |
| 65346 |     | Some conversion specifiers can be modified by the E or O modifier characters to indicate that an                                                                |
| 65347 |     | alternative format or specification should be used rather than the one normally used by the                                                                     |
| 65348 |     | unmodified conversion specifier. If the alternative format or specification does not exist for the                                                              |
| 65349 |     | current locale (see ERA in XBD <a href="#">Section 7.3.5</a> , on page 159), the behavior shall be as if the                                                    |
| 65350 |     | unmodified conversion specification were used.                                                                                                                  |
| 65351 | %Ec | Replaced by the locale's alternative appropriate date and time representation.                                                                                  |
| 65352 | %EC | Replaced by the name of the base year (period) in the locale's alternative                                                                                      |
| 65353 |     | representation.                                                                                                                                                 |
| 65354 | %Ex | Replaced by the locale's alternative date representation.                                                                                                       |
| 65355 | %EX | Replaced by the locale's alternative time representation.                                                                                                       |
| 65356 | %Ey | Replaced by the offset from %EC (year only) in the locale's alternative representation.                                                                         |
| 65357 | %EY | Replaced by the full alternative year representation.                                                                                                           |
| 65358 | %Od | Replaced by the day of the month, using the locale's alternative numeric symbols, filled                                                                        |
| 65359 |     | as needed with leading zeros if there is any alternative symbol for zero; otherwise, with                                                                       |
| 65360 |     | leading <space> characters.                                                                                                                                     |
| 65361 | %Oe | Replaced by the day of the month, using the locale's alternative numeric symbols, filled                                                                        |
| 65362 |     | as needed with leading <space> characters.                                                                                                                      |
| 65363 | %OH | Replaced by the hour (24-hour clock) using the locale's alternative numeric symbols.                                                                            |
| 65364 | %OI | Replaced by the hour (12-hour clock) using the locale's alternative numeric symbols.                                                                            |
| 65365 | %Om | Replaced by the month using the locale's alternative numeric symbols.                                                                                           |
| 65366 | %OM | Replaced by the minutes using the locale's alternative numeric symbols.                                                                                         |

|       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                      |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 65367 | %OS                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Replaced by the seconds using the locale's alternative numeric symbols.                                                                                              |
| 65368 | %Ou                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Replaced by the weekday as a number in the locale's alternative representation (Monday=1).                                                                           |
| 65369 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                      |
| 65370 | %OU                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Replaced by the week number of the year (Sunday as the first day of the week, rules corresponding to %U) using the locale's alternative numeric symbols.             |
| 65371 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                      |
| 65372 | %OV                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Replaced by the week number of the year (Monday as the first day of the week, rules corresponding to %V) using the locale's alternative numeric symbols.             |
| 65373 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                      |
| 65374 | %Ow                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Replaced by the number of the weekday (Sunday=0) using the locale's alternative numeric symbols.                                                                     |
| 65375 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                      |
| 65376 | %OW                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Replaced by the week number of the year (Monday as the first day of the week) using the locale's alternative numeric symbols.                                        |
| 65377 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                      |
| 65378 | %Oy                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Replaced by the year (offset from %C) using the locale's alternative numeric symbols.                                                                                |
| 65379 | %g, %G, and %V give values according to the ISO 8601:2004 standard week-based year. In this system, weeks begin on a Monday and week 1 of the year is the week that includes January 4th, which is also the week that includes the first Thursday of the year, and is also the first week that contains at least four days in the year. If the first Monday of January is the 2nd, 3rd, or 4th, the preceding days are part of the last week of the preceding year; thus, for Saturday 2nd January 1999, %G is replaced by 1998 and %V is replaced by 53. If December 29th, 30th, or 31st is a Monday, it and any following days are part of week 1 of the following year. Thus, for Tuesday 30th December 1997, %G is replaced by 1998 and %V is replaced by 01. |                                                                                                                                                                      |
| 65380 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                      |
| 65381 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                      |
| 65382 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                      |
| 65383 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                      |
| 65384 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                      |
| 65385 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                      |
| 65386 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                      |
| 65387 | If a conversion specifier is not one of the above, the behavior is undefined.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |                                                                                                                                                                      |
| 65388 | CX                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | The behavior is undefined if the <i>locale</i> argument to <i>strptime_l()</i> is the special locale object LC_GLOBAL_LOCALE or is not a valid locale object handle. |
| 65389 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                      |

## 65390 RETURN VALUE

65391 If the total number of resulting bytes including the terminating null byte is not more than  
 65392 *maxsize*, these functions shall return the number of bytes placed into the array pointed to by *s*,  
 65393 not including the terminating NUL character. Otherwise, 0 shall be returned and the contents of  
 65394 the array are unspecified.

## 65395 ERRORS

65396 No errors are defined.

## 65397 EXAMPLES

### 65398 Getting a Localized Date String

65399 The following example first sets the locale to the user's default. The locale information will be  
 65400 used in the *nl\_langinfo()* and *strptime()* functions. The *nl\_langinfo()* function returns the localized  
 65401 date string which specifies how the date is laid out. The *strptime()* function takes this  
 65402 information and, using the **tm** structure for values, places the date and time information into  
 65403 *datestring*.

```
65404 #include <time.h>
65405 #include <locale.h>
65406 #include <langinfo.h>
65407 ...
65408 struct tm *tm;
65409 char datestring[256];
65410 ...
```

```

65411     setlocale (LC_ALL, "");
65412     ...
65413     strptime (datestring, sizeof(datestring), nl_langinfo (D_T_FMT), tm);
65414     ...

```

## APPLICATION USAGE

The range of values for %S is [00,60] rather than [00,59] to allow for the occasional leap second.

Some of the conversion specifications are duplicates of others. They are included for compatibility with *nl\_cxtime()* and *nl\_ascxtime()*, which were published in Issue 2.

The %C, %F, %G, and %Y format specifiers in *strptime()* always print full values, but the *strptime()* %C, %F, and %Y format specifiers only scan two digits (assumed to be the first two digits of a four-digit year) for %C and four digits (assumed to be the entire (four-digit) year) for %F and %Y. This mimics the behavior of *printf()* and *scanf()*; that is:

```
printf("%2d", x = 1000);
```

prints "1000", but:

```
scanf("%2d", &x);
```

when given "1000" as input will only store 10 in x). Applications using extended ranges of years must be sure that the number of digits specified for scanning years with *strptime()* matches the number of digits that will actually be present in the input stream. Historic implementations of the %Y conversion specification (with no flags and no minimum field width) produced different output formats. Some always produced at least four digits (with 0 fill for years from 0 through 999) while others only produced the number of digits present in the year (with no fill and no padding). These two forms can be produced with the '0' flag and a minimum field width options using the conversions specifications %04Y and %01Y, respectively.

In the past, the C and POSIX standards specified that %F produced an ISO 8601:2004 standard date format, but didn't specify which one. For years in the range [0001,9999], POSIX.1-2008 requires that the output produced match the ISO 8601:2004 standard complete representation extended format (YYYY-MM-DD) and for years outside of this range produce output that matches the ISO 8601:2004 standard expanded representation extended format (<+/-><Underline>Y</Underline>YYYY-MM-DD). To fully meet ISO 8601:2004 standard requirements, the producer and consumer must agree on a date format that has a specific number of bytes reserved to hold the characters used to represent the years that is sufficiently large to hold all values that will be shared. For example, the %+13F conversion specification will produce output matching the format "<+/->YYYYYY-MM-DD" (a leading '+' or '-' sign; a six-digit, 0-filled year; a '-' ; a two-digit, leading 0-filled month; another '-' ; and the two-digit, leading 0-filled day within the month).

Note that if the year being printed is greater than 9999, the resulting string from the unadorned %F conversion specifications will not conform to the ISO 8601:2004 standard extended format, complete representation for a date and will instead be an extended format, expanded representation (presumably without the required agreement between the date's producer and consumer).

In the C or POSIX locale, the E and O modifiers are ignored and the replacement strings for the following specifiers are:

- %a      The first three characters of %A.
- %A      One of Sunday, Monday, ..., Saturday.

|       |    |                                          |
|-------|----|------------------------------------------|
| 65455 | %b | The first three characters of %B.        |
| 65456 | %B | One of January, February, ..., December. |
| 65457 | %c | Equivalent to %a %b %e %T %Y.            |
| 65458 | %p | One of AM or PM.                         |
| 65459 | %r | Equivalent to %I:%M:%S %p.               |
| 65460 | %x | Equivalent to %m/%d/%Y.                  |
| 65461 | %X | Equivalent to %T.                        |
| 65462 | %Z | Implementation-defined.                  |

#### 65463 RATIONALE

65464 The %Y conversion specification to *strptime()* was frequently assumed to be a four-digit year, but  
 65465 the ISO C standard does not specify that %Y is restricted to any subset of allowed values from the  
 65466 *tm\_year* field. Similarly, the %C conversion specification was assumed to be a two-digit field and  
 65467 the first part of the output from the %F conversion specification was assumed to be a four-digit  
 65468 field. With *tm\_year* being a signed 32 or more-bit **int** and with many current implementations  
 65469 supporting 64-bit **time\_t** types in one or more programming environments, these assumptions  
 65470 are clearly wrong.

65471 POSIX.1-2008 now allows the format specifications %0xC, %0xF, %0xG, and %0xY (where 'x' is  
 65472 a string of decimal digits used to specify printing and scanning of a string of *x* decimal digits)  
 65473 with leading zero fill characters. Allowing applications to set the field width enables them to  
 65474 agree on the number of digits to be printed and scanned in the ISO 8601:2004 standard  
 65475 expanded representation of a year (for %F, %G, and %Y) or all but the last two digits of the year  
 65476 (for %C). This is based on a feature in some versions of GNU **libc**'s *strptime()*. The GNU version  
 65477 allows specifying space, zero, or no-fill characters in *strptime()* format strings, but does not allow  
 65478 any flags to be specified in *strptime()* format strings. These implementations also allow these  
 65479 flags to be specified for any numeric field. POSIX.1-2008 only requires the zero fill flag ('0') and  
 65480 only requires that it be recognized when processing %C, %F, %G, and %Y specifications when a  
 65481 minimum field width is also specified. The '0' flag is the only flag needed to produce and scan  
 65482 the ISO 8601:2004 standard year fields using the extended format forms. POSIX.1-2008 also  
 65483 allows applications to specify the same flag and field width specifiers to be used in both  
 65484 *strptime()* and *strptime()* format strings for symmetry. Systems may provide other flag characters  
 65485 and may accept flags in conjunction with conversion specifiers other than %C, %F, %G, and %Y;  
 65486 but portable applications cannot depend on such extensions.

65487 POSIX.1-2008 now also allows the format specifications %+xC, %+xF, %+xG, and %+xY (where  
 65488 'x' is a string of decimal digits used to specify printing and scanning of a string of 'x' decimal  
 65489 digits) with leading zero fill characters and a leading '+' sign character if the year being  
 65490 converted is more than four digits or a minimum field width is specified that allows room for  
 65491 more than four digits for the year. This allows date providers and consumers to agree on a  
 65492 specific number of digits to represent a year as required by the ISO 8601:2004 standard  
 65493 expanded representation formats. The expanded representation formats all require the year to  
 65494 begin with a leading '+' or '-' sign. (All of these specifiers can also provide a leading '-'  
 65495 sign for negative years. Since negative years and the year 0 don't fit well with the Gregorian or  
 65496 Julian calendars, the normal ranges of dates start with year 1. The ISO C standard allows *tm\_year*  
 65497 to assume values corresponding to years before year 1, but the use of such years provided  
 65498 unspecified results.)

65499 Some earlier version of this standard specified that applications wanting to use *strptime()* to scan  
 65500 dates and times printed by *strptime()* should provide non-digit characters between fields to

separate years from months and days. It also supported %F to print and scan the ISO 8601:2004 standard extended format, complete representation date for years 1 through 9999 (i.e., YYYY-MM-DD). However, many applications were written to print (using *strptime()*) and scan (using *strptime()*) dates written using the basic format complete representation (four-digit years) and truncated representation (two-digit years) specified by the ISO 8601:2004 standard representation of dates and times which do not have any separation characters between fields. The ISO 8601:2004 standard also specifies basic format expanded representation where the creator and consumer of these fields agree beforehand to represent years as leading zero-filled strings of an agreed length of more than four digits to represent a year (again with no separation characters when year, month, and day are all displayed). Applications producing and consuming expanded representations are encouraged to use the '+' flag and an appropriate maximum field width to scan the year including the leading sign. Note that even without the '+' flag, years less than zero may be represented with a leading <hyphen-minus> for %F, %G, and %Y conversion specifications. Using negative years results in unspecified behavior.

If a format specification %+xF with the field width *x* greater than 11 is specified and the width is large enough to display the full year, the output string produced will match the ISO 8601:2004 standard subclause 4.1.2.4 expanded representation, extended format date representation for a specific day. (For years in the range [1,99999], %+12F is sufficient for an agreed five-digit year with a leading sign using the ISO 8601:2004 standard expanded representation, extended format for a specific day "<+/->YYYY-MM-DD".) Note also that years less than 0 may produce a leading <hyphen-minus> character ('-') when using %Y or %C whether or not the '0' or '+' flags are used.

The difference between the '0' flag and the '+' flag is whether the leading '+' character will be provided for years >9999 as required for the ISO 8601:2004 standard extended representation format containing a year. For example:

| Year   | Conversion Specification | <i>strptime()</i><br>Output | <i>strptime()</i><br>Scan Back |
|--------|--------------------------|-----------------------------|--------------------------------|
| 1970   | %Y                       | 1970                        | 1970                           |
| 1970   | %+4Y                     | 1970                        | 1970                           |
| 27     | %Y                       | 27 or 0027                  | 27                             |
| 270    | %Y                       | 270 or 0270                 | 270                            |
| 270    | %+4Y                     | 0270                        | 270                            |
| 17     | %C%Y                     | 0017                        | 17                             |
| 270    | %C%Y                     | 0270                        | 270                            |
| 12345  | %Y                       | 12345                       | 1234*                          |
| 12345  | %+4Y                     | +12345                      | 123*                           |
| 12345  | %05Y                     | 12345                       | 12345                          |
| 270    | %+5Y or %+3C%Y           | +0270                       | 270                            |
| 12345  | %+5Y or %+3C%Y           | +12345                      | 1234*                          |
| 12345  | %06Y or %04C%Y           | 012345                      | 12345                          |
| 12345  | %+6Y or %+4C%Y           | +12345                      | 12345                          |
| 123456 | %08Y or %06C%Y           | 00123456                    | 123456                         |
| 123456 | %+8Y or %+6C%Y           | +0123456                    | 123456                         |

In the cases above marked with a \* in the *strptime()* scan back field, the implied or specified number of characters scanned by *strptime()* was less than the number of characters output by *strptime()* using the same format; so the remaining digits of the year were dropped when the output date produced by *strptime()* was scanned back in by *strptime()*.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*asctime()*, *clock()*, *ctime()*, *difftime()*, *getdate()*, *gmtime()*, *localtime()*, *mktime()*, *strptime()*, *time()*, *tzset()*, *uselocale()*, *utime()*

XBD Section 7.3.5 (on page 159), **<time.h>**

**CHANGE HISTORY**

First released in Issue 3.

**Issue 5**

The description of %OV is changed to be consistent with %V and defines Monday as the first day of the week.

The description of %Oy is clarified.

**Issue 6**

Extensions beyond the ISO C standard are marked.

The Open Group Corrigendum U033/8 is applied. The %V conversion specifier is changed from “Otherwise, it is week 53 of the previous year, and the next week is week 1” to “Otherwise, it is the last week of the previous year, and the next week is week 1”.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The %C, %D, %e, %h, %n, %r, %R, %t, and %T conversion specifiers are added.
- The modified conversion specifiers are added for consistency with the ISO POSIX-2 standard *date* utility.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *strptime()* prototype is updated.
- The DESCRIPTION is extensively revised.
- The %z conversion specifier is added.

An example is added.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/60 is applied.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #163 is applied.

The *strptime\_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0605 [283], XSH/TC1-2008/0606 [283], XSH/TC1-2008/0607 [193], and XSH/TC1-2008/0608 [193] are applied.



65582 **NAME**

65583        strlen, strlen — get length of fixed size string

65584 **SYNOPSIS**

65585        #include &lt;string.h&gt;

65586        size\_t strlen(const char \*s);

65587 CX     size\_t strnlen(const char \*s, size\_t maxlen);

65588 **DESCRIPTION**

65589 CX     For *strlen()*: The functionality described on this reference page is aligned with the ISO C  
 65590 standard. Any conflict between the requirements described here and the ISO C standard is  
 65591 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

65592        The *strlen()* function shall compute the number of bytes in the string to which *s* points, not  
 65593 including the terminating NUL character.

65594 CX     The *strnlen()* function shall compute the smaller of the number of bytes in the array to which *s*  
 65595 points, not including any terminating NUL character, or the value of the *maxlen* argument. The  
 65596 *strnlen()* function shall never examine more than *maxlen* bytes of the array pointed to by *s*.

65597 **RETURN VALUE**

65598        The *strlen()* function shall return the length of *s*; no return value shall be reserved to indicate an  
 65599 error.

65600 CX     The *strnlen()* function shall return the number of bytes preceding the first null byte in the array  
 65601 to which *s* points, if *s* contains a null byte within the first *maxlen* bytes; otherwise, it shall return  
 65602 *maxlen*.

65603 **ERRORS**

65604        No errors are defined.

65605 **EXAMPLES**65606        **Getting String Lengths**

65607        The following example sets the maximum length of *key* and *data* by using *strlen()* to get the  
 65608 lengths of those strings.

```

65609     #include <string.h>
65610     ...
65611     struct element {
65612         char *key;
65613         char *data;
65614     };
65615     ...
65616     char *key, *data;
65617     int len;

65618     *keylength = *datalength = 0;
65619     ...
65620     if ((len = strlen(key)) > *keylength)
65621         *keylength = len;
65622     if ((len = strlen(data)) > *datalength)
65623         *datalength = len;
65624     ...

```

65625 **APPLICATION USAGE**

65626 None.

65627 **RATIONALE**

65628 None.

65629 **FUTURE DIRECTIONS**

65630 None.

65631 **SEE ALSO**

65632 *wcslen()*

65633 XBD <string.h>

65634 **CHANGE HISTORY**

65635 First released in Issue 1. Derived from Issue 1 of the SVID.

65636 **Issue 5**

65637 The RETURN VALUE section is updated to indicate that *strlen()* returns the length of *s*, and not  
65638 *s* itself as was previously stated.

65639 **Issue 7**

65640 The *strnlen()* function is added from The Open Group Technical Standard, 2006, Extended API  
65641 Set Part 1.



65642 **NAME**

65643 strncasecmp, strncasecmp\_l — case-insensitive string comparisons

65644 **SYNOPSIS**

65645 #include &lt;strings.h&gt;

65646 int strncasecmp(const char \*s1, const char \*s2, size\_t n);

65647 int strncasecmp\_l(const char \*s1, const char \*s2,

65648 size\_t n, locale\_t locale);

65649 **DESCRIPTION**65650 Refer to *strcasecmp()*.

65651 **NAME**

65652 strncat — concatenate a string with part of another

65653 **SYNOPSIS**

65654 #include &lt;string.h&gt;

65655 char \*strncat(char \*restrict s1, const char \*restrict s2, size\_t n);

65656 **DESCRIPTION**

65657 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
65658 conflict between the requirements described here and the ISO C standard is unintentional. This  
65659 volume of POSIX.1-2008 defers to the ISO C standard.

65660 The *strncat()* function shall append not more than *n* bytes (a NUL character and bytes that  
65661 follow it are not appended) from the array pointed to by *s2* to the end of the string pointed to by  
65662 *s1*. The initial byte of *s2* overwrites the NUL character at the end of *s1*. A terminating NUL  
65663 character is always appended to the result. If copying takes place between objects that overlap,  
65664 the behavior is undefined.

65665 **RETURN VALUE**65666 The *strncat()* function shall return *s1*; no return value shall be reserved to indicate an error.65667 **ERRORS**

65668 No errors are defined.

65669 **EXAMPLES**

65670 None.

65671 **APPLICATION USAGE**

65672 None.

65673 **RATIONALE**

65674 None.

65675 **FUTURE DIRECTIONS**

65676 None.

65677 **SEE ALSO**65678 [strcat\(\)](#)65679 XBD [<string.h>](#)65680 **CHANGE HISTORY**

65681 First released in Issue 1. Derived from Issue 1 of the SVID.

65682 **Issue 6**65683 The *strncat()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

65684 **NAME**

65685 strncmp — compare part of two strings

65686 **SYNOPSIS**

65687 #include &lt;string.h&gt;

65688 int strncmp(const char \*s1, const char \*s2, size\_t n);

65689 **DESCRIPTION**

65690 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 65691 conflict between the requirements described here and the ISO C standard is unintentional. This  
 65692 volume of POSIX.1-2008 defers to the ISO C standard.

65693 The *strncmp()* function shall compare not more than *n* bytes (bytes that follow a NUL character  
 65694 are not compared) from the array pointed to by *s1* to the array pointed to by *s2*.

65695 The sign of a non-zero return value is determined by the sign of the difference between the  
 65696 values of the first pair of bytes (both interpreted as type **unsigned char**) that differ in the strings  
 65697 being compared.

65698 **RETURN VALUE**

65699 Upon successful completion, *strncmp()* shall return an integer greater than, equal to, or less than  
 65700 0, if the possibly null-terminated array pointed to by *s1* is greater than, equal to, or less than the  
 65701 possibly null-terminated array pointed to by *s2* respectively.

65702 **ERRORS**

65703 No errors are defined.

65704 **EXAMPLES**

65705 None.

65706 **APPLICATION USAGE**

65707 None.

65708 **RATIONALE**

65709 None.

65710 **FUTURE DIRECTIONS**

65711 None.

65712 **SEE ALSO**65713 *strcmp()*

65714 XBD &lt;string.h&gt;

65715 **CHANGE HISTORY**

65716 First released in Issue 1. Derived from Issue 1 of the SVID.

65717 **Issue 6**

65718 Extensions beyond the ISO C standard are marked.

65719 **NAME**

65720       stpnpcy, strncpy — copy fixed length string, returning a pointer to the array end

65721 **SYNOPSIS**

65722       #include &lt;string.h&gt;

```
65723 CX     char *stpnpcy(char *restrict s1, const char *restrict s2, size_t n);
65724     char *strncpy(char *restrict s1, const char *restrict s2, size_t n);
```

65725 **DESCRIPTION**

65726 CX     For *strncpy()*: The functionality described on this reference page is aligned with the ISO C  
65727 standard. Any conflict between the requirements described here and the ISO C standard is  
65728 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

65729 CX     The *stpnpcy()* and *strncpy()* functions shall copy not more than *n* bytes (bytes that follow a NUL  
65730 character are not copied) from the array pointed to by *s2* to the array pointed to by *s1*.

65731       If the array pointed to by *s2* is a string that is shorter than *n* bytes, NUL characters shall be  
65732 appended to the copy in the array pointed to by *s1*, until *n* bytes in all are written.

65733       If copying takes place between objects that overlap, the behavior is undefined.

65734 **RETURN VALUE**

65735 CX     If a NUL character is written to the destination, the *stpnpcy()* function shall return the address of  
65736 the first such NUL character. Otherwise, it shall return *&s1[n]*.

65737       The *strncpy()* function shall return *s1*.

65738       No return values are reserved to indicate an error.

65739 **ERRORS**

65740       No errors are defined.

65741 **EXAMPLES**

65742       None.

65743 **APPLICATION USAGE**

65744       Applications must provide the space in *s1* for the *n* bytes to be transferred, as well as ensure that  
65745 the *s2* and *s1* arrays do not overlap.

65746       Character movement is performed differently in different implementations. Thus, overlapping  
65747 moves may yield surprises.

65748       If there is no NUL character byte in the first *n* bytes of the array pointed to by *s2*, the result is not  
65749 null-terminated.

65750 **RATIONALE**

65751       None.

65752 **FUTURE DIRECTIONS**

65753       None.

65754 **SEE ALSO**

65755       *strcpy()*, *wcsncpy()*

65756       XBD <string.h>

65757 **CHANGE HISTORY**

65758       First released in Issue 1. Derived from Issue 1 of the SVID.

65759 **Issue 6**

65760 The *strncpy()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

65761 **Issue 7**

65762 The *stpncpy()* function is added from The Open Group Technical Standard, 2006, Extended API  
65763 Set Part 1.

65764 **NAME**

65765       strndup — duplicate a specific number of bytes from a string

65766 **SYNOPSIS**

65767 CX       #include &lt;string.h&gt;

65768       char \*strndup(const char \*s, size\_t size);

65769 **DESCRIPTION**65770       Refer to *strdup()*.

65771 **NAME**

65772 strnlen — get length of fixed size string

65773 **SYNOPSIS**

```
65774 CX #include <string.h>
65775 size_t strnlen(const char *s, size_t maxlen);
```

65776 **DESCRIPTION**65777 Refer to *strlen()*.

65778 **NAME**

65779 strpbrk — scan a string for a byte

65780 **SYNOPSIS**

65781 #include &lt;string.h&gt;

65782 char \*strpbrk(const char \*s1, const char \*s2);

65783 **DESCRIPTION**

65784 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
65785 conflict between the requirements described here and the ISO C standard is unintentional. This  
65786 volume of POSIX.1-2008 defers to the ISO C standard.

65787 The *strpbrk()* function shall locate the first occurrence in the string pointed to by *s1* of any byte  
65788 from the string pointed to by *s2*.

65789 **RETURN VALUE**

65790 Upon successful completion, *strpbrk()* shall return a pointer to the byte or a null pointer if no  
65791 byte from *s2* occurs in *s1*.

65792 **ERRORS**

65793 No errors are defined.

65794 **EXAMPLES**

65795 None.

65796 **APPLICATION USAGE**

65797 None.

65798 **RATIONALE**

65799 None.

65800 **FUTURE DIRECTIONS**

65801 None.

65802 **SEE ALSO**65803 *strchr()*, *strrchr()*

65804 XBD &lt;string.h&gt;

65805 **CHANGE HISTORY**

65806 First released in Issue 1. Derived from Issue 1 of the SVID.



65807 **NAME**

65808 strptime — date and time conversion

65809 **SYNOPSIS**

```
65810 XSI    #include <time.h>
65811
65811    char *strptime(const char *restrict buf, const char *restrict format,
65812                  struct tm *restrict tm);
```

65813 **DESCRIPTION**

65814 The *strptime()* function shall convert the character string pointed to by *buf* to values which are  
 65815 stored in the **tm** structure pointed to by *tm*, using the format specified by *format*.

65816 The format is composed of zero or more directives. Each directive is composed of one of the  
 65817 following: one or more white-space characters (as specified by *isspace()*); an ordinary character  
 65818 (neither '%' nor a white-space character); or a conversion specification.

65819 Each conversion specification is introduced by the '%' character after which the following  
 65820 appear in sequence:

- 65821 • An optional flag, the zero character ('0') or the <plus-sign> character ('+'), which is  
 65822 ignored.
- 65823 • An optional field width. If a field width is specified, it shall be interpreted as a string of  
 65824 decimal digits that will determine the maximum number of bytes converted for the  
 65825 conversion rather than the number of bytes specified below in the description of the  
 65826 conversion specifiers.
- 65827 • An optional E or O modifier.
- 65828 • A terminating conversion specifier character that indicates the type of conversion to be  
 65829 applied.

65830 The conversions are determined using the *LC\_TIME* category of the current locale. The  
 65831 application shall ensure that there is white-space or other non-alphanumeric characters between  
 65832 any two conversion specifications unless all of the adjacent conversion specifications convert a  
 65833 known, fixed number of characters. In the following list, the maximum number of characters  
 65834 scanned (excluding the one matching the next directive) is as follows:

- 65835 • If a maximum field width is specified, then that number
- 65836 • Otherwise, the pattern "{x}" indicates that the maximum is *x*
- 65837 • Otherwise, the pattern "[x,y]" indicates that the value shall fall within the range given  
 65838 (both bounds being inclusive), and the maximum number of characters scanned shall be  
 65839 the maximum required to represent any value in the range without leading zeros and  
 65840 without a leading <plus-sign>

65841 The following conversion specifiers are supported.

65842 The results are unspecified if a modifier is specified with a flag or with a minimum field width,  
 65843 or if a field width is specified for any conversion specifier other than C or Y.

- |       |   |                                                                                       |
|-------|---|---------------------------------------------------------------------------------------|
| 65844 | a | The day of the week, using the locale's weekday names; either the abbreviated or full |
| 65845 |   | name may be specified.                                                                |
| 65846 | A | Equivalent to %a.                                                                     |

|       |   |                                                                                                                                                                                                                                                                                                                                                                   |
|-------|---|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 65847 | b | The month, using the locale's month names; either the abbreviated or full name may be specified.                                                                                                                                                                                                                                                                  |
| 65848 |   |                                                                                                                                                                                                                                                                                                                                                                   |
| 65849 | B | Equivalent to %b.                                                                                                                                                                                                                                                                                                                                                 |
| 65850 | c | Replaced by the locale's appropriate date and time representation.                                                                                                                                                                                                                                                                                                |
| 65851 | C | All but the last two digits of the year {2}; leading zeros shall be permitted but shall not be required. A leading '+' or '-' character shall be permitted before any leading zeros but shall not be required.                                                                                                                                                    |
| 65852 |   |                                                                                                                                                                                                                                                                                                                                                                   |
| 65853 |   |                                                                                                                                                                                                                                                                                                                                                                   |
| 65854 | d | The day of the month [01,31]; leading zeros shall be permitted but shall not be required.                                                                                                                                                                                                                                                                         |
| 65855 | D | The date as %m/%d/%Y.                                                                                                                                                                                                                                                                                                                                             |
| 65856 | e | Equivalent to %d.                                                                                                                                                                                                                                                                                                                                                 |
| 65857 | h | Equivalent to %b.                                                                                                                                                                                                                                                                                                                                                 |
| 65858 | H | The hour (24-hour clock) [00,23]; leading zeros shall be permitted but shall not be required.                                                                                                                                                                                                                                                                     |
| 65859 |   |                                                                                                                                                                                                                                                                                                                                                                   |
| 65860 | I | The hour (12-hour clock) [01,12]; leading zeros shall be permitted but shall not be required.                                                                                                                                                                                                                                                                     |
| 65861 |   |                                                                                                                                                                                                                                                                                                                                                                   |
| 65862 | j | The day number of the year [001,366]; leading zeros shall be permitted but shall not be required.                                                                                                                                                                                                                                                                 |
| 65863 |   |                                                                                                                                                                                                                                                                                                                                                                   |
| 65864 | m | The month number [01,12]; leading zeros shall be permitted but shall not be required.                                                                                                                                                                                                                                                                             |
| 65865 | M | The minute [00,59]; leading zeros shall be permitted but shall not be required.                                                                                                                                                                                                                                                                                   |
| 65866 | n | Any white space.                                                                                                                                                                                                                                                                                                                                                  |
| 65867 | p | The locale's equivalent of a.m. or p.m.                                                                                                                                                                                                                                                                                                                           |
| 65868 | r | 12-hour clock time using the AM/PM notation if <b>t_fmt_ampm</b> is not an empty string in the <b>LC_TIME</b> portion of the current locale; in the POSIX locale, this shall be equivalent to %I:%M:%S %p.                                                                                                                                                        |
| 65869 |   |                                                                                                                                                                                                                                                                                                                                                                   |
| 65870 |   |                                                                                                                                                                                                                                                                                                                                                                   |
| 65871 | R | The time as %H:%M.                                                                                                                                                                                                                                                                                                                                                |
| 65872 | S | The seconds [00,60]; leading zeros shall be permitted but shall not be required.                                                                                                                                                                                                                                                                                  |
| 65873 | t | Any white space.                                                                                                                                                                                                                                                                                                                                                  |
| 65874 | T | The time as %H:%M:%S.                                                                                                                                                                                                                                                                                                                                             |
| 65875 | U | The week number of the year (Sunday as the first day of the week) as a decimal number [00,53]; leading zeros shall be permitted but shall not be required.                                                                                                                                                                                                        |
| 65876 |   |                                                                                                                                                                                                                                                                                                                                                                   |
| 65877 | w | The weekday as a decimal number [0,6], with 0 representing Sunday.                                                                                                                                                                                                                                                                                                |
| 65878 | W | The week number of the year (Monday as the first day of the week) as a decimal number [00,53]; leading zeros shall be permitted but shall not be required.                                                                                                                                                                                                        |
| 65879 |   |                                                                                                                                                                                                                                                                                                                                                                   |
| 65880 | x | The date, using the locale's date format.                                                                                                                                                                                                                                                                                                                         |
| 65881 | X | The time, using the locale's time format.                                                                                                                                                                                                                                                                                                                         |
| 65882 | y | The last two digits of the year. When <i>format</i> contains neither a C conversion specifier nor a Y conversion specifier, values in the range [69,99] shall refer to years 1969 to 1999 inclusive and values in the range [00,68] shall refer to years 2000 to 2068 inclusive; leading zeros shall be permitted but shall not be required. A leading '+' or '-' |
| 65883 |   |                                                                                                                                                                                                                                                                                                                                                                   |
| 65884 |   |                                                                                                                                                                                                                                                                                                                                                                   |
| 65885 |   |                                                                                                                                                                                                                                                                                                                                                                   |

|       |              |                                                                                          |
|-------|--------------|------------------------------------------------------------------------------------------|
| 65886 |              | character shall be permitted before any leading zeros but shall not be required.         |
| 65887 | <b>Note:</b> | It is expected that in a future version of this standard the default century inferred    |
| 65888 |              | from a 2-digit year will change. (This would apply to all commands accepting a           |
| 65889 |              | 2-digit year as input.)                                                                  |
| 65890 | Y            | The full year {4}; leading zeros shall be permitted but shall not be required. A leading |
| 65891 |              | '+' or '-' character shall be permitted before any leading zeros but shall not be        |
| 65892 |              | required.                                                                                |
| 65893 | %            | Replaced by %.                                                                           |

#### 65894 Modified Conversion Specifiers

65895 Some conversion specifiers can be modified by the E and O modifier characters to indicate that  
 65896 an alternative format or specification should be used rather than the one normally used by the  
 65897 unmodified conversion specifier. If the alternative format or specification does not exist in the  
 65898 current locale, the behavior shall be as if the unmodified conversion specification were used.

|       |     |                                                                                      |
|-------|-----|--------------------------------------------------------------------------------------|
| 65899 | %Ec | The locale's alternative appropriate date and time representation.                   |
| 65900 | %EC | The name of the base year (period) in the locale's alternative representation.       |
| 65901 | %Ex | The locale's alternative date representation.                                        |
| 65902 | %EX | The locale's alternative time representation.                                        |
| 65903 | %Ey | The offset from %EC (year only) in the locale's alternative representation.          |
| 65904 | %EY | The full alternative year representation.                                            |
| 65905 | %Od | The day of the month using the locale's alternative numeric symbols; leading zeros   |
| 65906 |     | shall be permitted but shall not be required.                                        |
| 65907 | %Oe | Equivalent to %Od.                                                                   |
| 65908 | %OH | The hour (24-hour clock) using the locale's alternative numeric symbols.             |
| 65909 | %OI | The hour (12-hour clock) using the locale's alternative numeric symbols.             |
| 65910 | %Om | The month using the locale's alternative numeric symbols.                            |
| 65911 | %OM | The minutes using the locale's alternative numeric symbols.                          |
| 65912 | %OS | The seconds using the locale's alternative numeric symbols.                          |
| 65913 | %OU | The week number of the year (Sunday as the first day of the week) using the locale's |
| 65914 |     | alternative numeric symbols.                                                         |
| 65915 | %Ow | The number of the weekday (Sunday=0) using the locale's alternative numeric          |
| 65916 |     | symbols.                                                                             |
| 65917 | %OW | The week number of the year (Monday as the first day of the week) using the locale's |
| 65918 |     | alternative numeric symbols.                                                         |
| 65919 | %Oy | The year (offset from %C) using the locale's alternative numeric symbols.            |

65920 A conversion specification composed of white-space characters is executed by scanning input up  
 65921 to the first character that is not white-space (which remains unscanned), or until no more  
 65922 characters can be scanned.

65923 A conversion specification that is an ordinary character is executed by scanning the next  
 65924 character from the buffer. If the character scanned from the buffer differs from the one  
 65925 comprising the directive, the directive fails, and the differing and subsequent characters remain

65926 unscanned.

65927 A series of conversion specifications composed of %n, %t, white-space characters, or any  
65928 combination is executed by scanning up to the first character that is not white space (which  
65929 remains unscanned), or until no more characters can be scanned.

65930 Any other conversion specification is executed by scanning characters until a character matching  
65931 the next directive is scanned, or until no more characters can be scanned. These characters,  
65932 except the one matching the next directive, are then compared to the locale values associated  
65933 with the conversion specifier. If a match is found, values for the appropriate **tm** structure  
65934 members are set to values corresponding to the locale information. Case is ignored when  
65935 matching items in *buf* such as month or weekday names. If no match is found, *strptime()* fails  
65936 and no more characters are scanned.

#### 65937 RETURN VALUE

65938 Upon successful completion, *strptime()* shall return a pointer to the character following the last  
65939 character parsed. Otherwise, a null pointer shall be returned.

#### 65940 ERRORS

65941 No errors are defined.

#### 65942 EXAMPLES

##### 65943 Convert a Date-Plus-Time String to Broken-Down Time and Then into Seconds

65944 The following example demonstrates the use of *strptime()* to convert a string into broken-down  
65945 time. The broken-down time is then converted into seconds since the Epoch using *mktime()*.

```
65946 #include <time.h>
65947 ...
65948 struct tm tm;
65949 time_t t;
65950 if (strptime("6 Dec 2001 12:33:45", "%d %b %Y %H:%M:%S", &tm) == NULL)
65951     /* Handle error */;
65952 printf("year: %d; month: %d; day: %d;\n",
65953        tm.tm_year, tm.tm_mon, tm.tm_mday);
65954 printf("hour: %d; minute: %d; second: %d\n",
65955        tm.tm_hour, tm.tm_min, tm.tm_sec);
65956 printf("week day: %d; year day: %d\n", tm.tm_wday, tm.tm_yday);
65957 tm.tm_isdst = -1;          /* Not set by strptime(); tells mktime()
65958                            to determine whether daylight saving time
65959                            is in effect */
65960 t = mktime(&tm);
65961 if (t == -1)
65962     /* Handle error */;
65963 printf("seconds since the Epoch: %ld\n", (long) t);"
```

#### 65964 APPLICATION USAGE

65965 Several “equivalent to” formats and the special processing of white-space characters are  
65966 provided in order to ease the use of identical *format* strings for *strptime()* and *strptime()*.

65967 It should be noted that dates constructed by the *strptime()* function with the %Y or %C%y  
65968 conversion specifiers may have values larger than 9999. If the *strptime()* function is used to read  
65969 such values using %C%y or %Y, the year values will be truncated to four digits. Applications

65970 should use `%+w%y` or `%+xY` with *w* and *x* set large enough to contain the full value of any years  
65971 that will be printed or scanned.

65972 See also the APPLICATION USAGE section in *strptime()*.

65973 It is unspecified whether multiple calls to *strptime()* using the same **tm** structure will update the  
65974 current contents of the structure or overwrite all contents of the structure. Conforming  
65975 applications should make a single call to *strptime()* with a format and all data needed to  
65976 completely specify the date and time being converted.

#### 65977 RATIONALE

65978 See the RATIONALE section for *strptime()*.

#### 65979 FUTURE DIRECTIONS

65980 None.

#### 65981 SEE ALSO

65982 *fprintf()*, *fscanf()*, *strptime()*, *time()*

65983 XBD **<time.h>**

#### 65984 CHANGE HISTORY

65985 First released in Issue 4.

#### 65986 Issue 5

65987 Moved from ENHANCED I18N to BASE.

65988 The [ENOSYS] error is removed.

65989 The exact meaning of the `%y` and `%Oy` specifiers is clarified in the DESCRIPTION.

#### 65990 Issue 6

65991 The Open Group Corrigendum U033/5 is applied. The `%r` specifier description is reworded.

65992 The normative text is updated to avoid use of the term “must” for application requirements.

65993 The **restrict** keyword is added to the *strptime()* prototype for alignment with the  
65994 ISO/IEC 9899:1999 standard.

65995 The Open Group Corrigendum U047/2 is applied.

65996 The DESCRIPTION is updated to use the terms “conversion specifier” and “conversion  
65997 specification” for consistency with *strptime()*.

65998 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/133 is applied, adding the example to the  
65999 EXAMPLES section.

#### 66000 Issue 7

66001 Austin Group Interpretation 1003.1-2001 #041 is applied, updating the DESCRIPTION and  
66002 APPLICATION USAGE sections.

66003 Austin Group Interpretation 1003.1-2001 #163 is applied.

66004 SD5-XSH-ERN-67 is applied, correcting the APPLICATION USAGE to remove the impression  
66005 that `%Y` is 4-digit years.

66006 **NAME**

66007       strrchr — string scanning operation

66008 **SYNOPSIS**

66009       #include &lt;string.h&gt;

66010       char \*strrchr(const char \*s, int c);

66011 **DESCRIPTION**

66012 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
66013       conflict between the requirements described here and the ISO C standard is unintentional. This  
66014       volume of POSIX.1-2008 defers to the ISO C standard.

66015       The *strrchr()* function shall locate the last occurrence of *c* (converted to a **char**) in the string  
66016       pointed to by *s*. The terminating NUL character is considered to be part of the string.

66017 **RETURN VALUE**

66018       Upon successful completion, *strrchr()* shall return a pointer to the byte or a null pointer if *c* does  
66019       not occur in the string.

66020 **ERRORS**

66021       No errors are defined.

66022 **EXAMPLES**66023       **Finding the Base Name of a File**

66024       The following example uses *strrchr()* to get a pointer to the base name of a file. The *strrchr()*  
66025       function searches backwards through the name of the file to find the last '/' character in *name*.  
66026       This pointer (plus one) will point to the base name of the file.

```
66027       #include <string.h>
66028       ...
66029       const char *name;
66030       char *basename;
66031       ...
66032       basename = strrchr(name, '/') + 1;
66033       ...
```

66034 **APPLICATION USAGE**

66035       None.

66036 **RATIONALE**

66037       None.

66038 **FUTURE DIRECTIONS**

66039       None.

66040 **SEE ALSO**66041       [\*strchr\(\)\*](#)66042       XBD [\*<string.h>\*](#)66043 **CHANGE HISTORY**

66044       First released in Issue 1. Derived from Issue 1 of the SVID.

66045 **NAME**

66046 strsignal — get name of signal

66047 **SYNOPSIS**

```
66048 CX    #include <string.h>
66049      char *strsignal(int signum);
```

66050 **DESCRIPTION**

66051 The *strsignal()* function shall map the signal number in *signum* to an implementation-defined  
 66052 string and shall return a pointer to it. It shall use the same set of messages as the *psignal()*  
 66053 function.

66054 The application shall not modify the string returned. The returned pointer might be invalidated  
 66055 or the string content might be overwritten by a subsequent call to *strsignal()* or *setlocale()*. The  
 66056 returned pointer might also be invalidated if the calling thread is terminated. +

66057 The contents of the message strings returned by *strsignal()* should be determined by the setting  
 66058 of the *LC\_MESSAGES* category in the current locale.

66059 The implementation shall behave as if no function defined in this standard calls *strsignal()*.

66060 Since no return value is reserved to indicate an error, an application wishing to check for error  
 66061 situations should set *errno* to 0, then call *strsignal()*, then check *errno*.

66062 The *strsignal()* function need not be thread-safe.

66063 **RETURN VALUE**

66064 Upon successful completion, *strsignal()* shall return a pointer to a string. Otherwise, if *signum* is  
 66065 not a valid signal number, the return value is unspecified.

66066 **ERRORS**

66067 No errors are defined.

66068 **EXAMPLES**

66069 None.

66070 **APPLICATION USAGE**

66071 None.

66072 **RATIONALE**

66073 If *signum* is not a valid signal number, some implementations return NULL, while for others the  
 66074 *strsignal()* function returns a pointer to a string containing an unspecified message denoting an  
 66075 unknown signal. POSIX.1-2008 leaves this return value unspecified.

66076 **FUTURE DIRECTIONS**

66077 None.

66078 **SEE ALSO**

66079 *psiginfo()*, *setlocale()*

66080 XBD *<string.h>*

66081 **CHANGE HISTORY**

66082 First released in Issue 7.

66083 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0609 [75] is applied.

66084 **NAME**

66085 strspn — get length of a substring

66086 **SYNOPSIS**

66087 #include &lt;string.h&gt;

66088 size\_t strspn(const char \*s1, const char \*s2);

66089 **DESCRIPTION**

66090 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
66091 conflict between the requirements described here and the ISO C standard is unintentional. This  
66092 volume of POSIX.1-2008 defers to the ISO C standard.

66093 The *strspn()* function shall compute the length (in bytes) of the maximum initial segment of the  
66094 string pointed to by *s1* which consists entirely of bytes from the string pointed to by *s2*.

66095 **RETURN VALUE**

66096 The *strspn()* function shall return the computed length; no return value is reserved to indicate an  
66097 error.

66098 **ERRORS**

66099 No errors are defined.

66100 **EXAMPLES**

66101 None.

66102 **APPLICATION USAGE**

66103 None.

66104 **RATIONALE**

66105 None.

66106 **FUTURE DIRECTIONS**

66107 None.

66108 **SEE ALSO**66109 [strcspn\(\)](#)66110 XBD [<string.h>](#)66111 **CHANGE HISTORY**

66112 First released in Issue 1. Derived from Issue 1 of the SVID.

66113 **Issue 5**

66114 The RETURN VALUE section is updated to indicate that *strspn()* returns the length of *s*, and not  
66115 *s* itself as was previously stated.

66116 **Issue 7**

66117 SD5-XSH-ERN-182 is applied.



66118 **NAME**

66119        strstr — find a substring

66120 **SYNOPSIS**

66121        #include &lt;string.h&gt;

66122        char \*strstr(const char \*s1, const char \*s2);

66123 **DESCRIPTION**

66124 CX        The functionality described on this reference page is aligned with the ISO C standard. Any  
 66125        conflict between the requirements described here and the ISO C standard is unintentional. This  
 66126        volume of POSIX.1-2008 defers to the ISO C standard.

66127        The *strstr()* function shall locate the first occurrence in the string pointed to by *s1* of the  
 66128        sequence of bytes (excluding the terminating NUL character) in the string pointed to by *s2*.

66129 **RETURN VALUE**

66130        Upon successful completion, *strstr()* shall return a pointer to the located string or a null pointer  
 66131        if the string is not found.

66132        If *s2* points to a string with zero length, the function shall return *s1*.

66133 **ERRORS**

66134        No errors are defined.

66135 **EXAMPLES**

66136        None.

66137 **APPLICATION USAGE**

66138        None.

66139 **RATIONALE**

66140        None.

66141 **FUTURE DIRECTIONS**

66142        None.

66143 **SEE ALSO**66144        *strchr()*

66145        XBD &lt;string.h&gt;

66146 **CHANGE HISTORY**

66147        First released in Issue 3. Included for alignment with the ANSI C standard.

**NAME**

strtod, strtodf, strtold — convert a string to a double-precision number

**SYNOPSIS**

```
#include <stdlib.h>
```

```
double strtod(const char *restrict nptr, char **restrict endptr);
```

```
float strtodf(const char *restrict nptr, char **restrict endptr);
```

```
long double strtold(const char *restrict nptr, char **restrict endptr);
```

**DESCRIPTION**

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

These functions shall convert the initial portion of the string pointed to by *nptr* to **double**, **float**, and **long double** representation, respectively. First, they decompose the input string into three parts:

1. An initial, possibly empty, sequence of white-space characters (as specified by *isspace()*)
2. A subject sequence interpreted as a floating-point constant or representing infinity or NaN
3. A final string of one or more unrecognized characters, including the terminating NUL character of the input string

Then they shall attempt to convert the subject sequence to a floating-point number, and return the result.

The expected form of the subject sequence is an optional '+' or '-' sign, then one of the following:

- A non-empty sequence of decimal digits optionally containing a radix character; then an optional exponent part consisting of the character 'e' or the character 'E', optionally followed by a '+' or '-' character, and then followed by one or more decimal digits
- A 0x or 0X, then a non-empty sequence of hexadecimal digits optionally containing a radix character; then an optional binary exponent part consisting of the character 'p' or the character 'P', optionally followed by a '+' or '-' character, and then followed by one or more decimal digits
- One of INF or INFINITY, ignoring case
- One of NAN or NAN(*n-char-sequence<sub>opt</sub>*), ignoring case in the NAN part, where:

n-char-sequence:

digit

nondigit

n-char-sequence digit

n-char-sequence nondigit

The subject sequence is defined as the longest initial subsequence of the input string, starting with the first non-white-space character, that is of the expected form. The subject sequence contains no characters if the input string is not of the expected form.

If the subject sequence has the expected form for a floating-point number, the sequence of characters starting with the first digit or the decimal-point character (whichever occurs first) shall be interpreted as a floating constant of the C language, except that the radix character shall be used in place of a period, and that if neither an exponent part nor a radix character appears in

66192 a decimal floating-point number, or if a binary exponent part does not appear in a hexadecimal  
 66193 floating-point number, an exponent part of the appropriate type with value zero is assumed to  
 66194 follow the last digit in the string. If the subject sequence begins with a <hyphen-minus>, the  
 66195 sequence shall be interpreted as negated. A character sequence INF or INFINITY shall be  
 66196 interpreted as an infinity, if representable in the return type, else as if it were a floating constant  
 66197 that is too large for the range of the return type. A character sequence NAN or NAN(*n-char-*  
 66198 *sequence<sub>opt</sub>*) shall be interpreted as a quiet NaN, if supported in the return type, else as if it were a  
 66199 subject sequence part that does not have the expected form; the meaning of the *n-char* sequences  
 66200 is implementation-defined. A pointer to the final string is stored in the object pointed to by  
 66201 *endptr*, provided that *endptr* is not a null pointer.

66202 If the subject sequence has the hexadecimal form and FLT\_RADIX is a power of 2, the value  
 66203 resulting from the conversion is correctly rounded.

66204 CX The radix character is defined in the current locale (category *LC\_NUMERIC*). In the POSIX  
 66205 locale, or in a locale where the radix character is not defined, the radix character shall default to  
 66206 a <period> ( ' . ' ).

66207 CX In other than the C or POSIX locale, additional locale-specific subject sequence forms may be  
 66208 accepted.

66209 If the subject sequence is empty or does not have the expected form, no conversion shall be  
 66210 performed; the value of *nptr* is stored in the object pointed to by *endptr*, provided that *endptr* is  
 66211 not a null pointer.

66212 These functions shall not change the setting of *errno* if successful.

66213 Since 0 is returned on error and is also a valid return on success, an application wishing to check  
 66214 for error situations should set *errno* to 0, then call *strtod()*, *strtodf()*, or *strtold()*, then check *errno*.

#### 66215 RETURN VALUE

66216 Upon successful completion, these functions shall return the converted value. If no conversion  
 66217 could be performed, 0 shall be returned, and *errno* may be set to [EINVAL].

66218 If the correct value is outside the range of representable values,  $\pm$ HUGE\_VAL,  $\pm$ HUGE\_VALF, or  
 66219  $\pm$ HUGE\_VALL shall be returned (according to the sign of the value), and *errno* shall be set to  
 66220 [ERANGE].

66221 If the correct value would cause an underflow, a value whose magnitude is no greater than the  
 66222 smallest normalized positive number in the return type shall be returned and *errno* set to  
 66223 [ERANGE].

#### 66224 ERRORS

66225 These functions shall fail if:

66226 CX [ERANGE] The value to be returned would cause overflow or underflow.

66227 These functions may fail if:

66228 CX [EINVAL] No conversion could be performed.

**EXAMPLES**

None.

**APPLICATION USAGE**

If the subject sequence has the hexadecimal form and FLT\_RADIX is not a power of 2, and the result is not exactly representable, the result should be one of the two numbers in the appropriate internal format that are adjacent to the hexadecimal floating source value, with the extra stipulation that the error should have a correct sign for the current rounding direction.

If the subject sequence has the decimal form and at most DECIMAL\_DIG (defined in `<float.h>`) significant digits, the result should be correctly rounded. If the subject sequence *D* has the decimal form and more than DECIMAL\_DIG significant digits, consider the two bounding, adjacent decimal strings *L* and *U*, both having DECIMAL\_DIG significant digits, such that the values of *L*, *D*, and *U* satisfy  $L \leq D \leq U$ . The result should be one of the (equal or adjacent) values that would be obtained by correctly rounding *L* and *U* according to the current rounding direction, with the extra stipulation that the error with respect to *D* should have a correct sign for the current rounding direction.

The changes to `strtod()` introduced by the ISO/IEC 9899:1999 standard can alter the behavior of well-formed applications complying with the ISO/IEC 9899:1990 standard and thus earlier versions of this standard. One such example would be:

```
int
what_kind_of_number (char *s)
{
    char *endp;
    double d;
    long l;

    d = strtod(s, &endp);
    if (s != endp && *endp == '\0')
        printf("It's a float with value %g\n", d);
    else
    {
        l = strtol(s, &endp, 0);
        if (s != endp && *endp == '\0')
            printf("It's an integer with value %ld\n", l);
        else
            return 1;
    }
    return 0;
}
```

If the function is called with:

```
what_kind_of_number ("0x10")
```

an ISO/IEC 9899:1990 standard-compliant library will result in the function printing:

```
It's an integer with value 16
```

With the ISO/IEC 9899:1999 standard, the result is:

```
It's a float with value 16
```

The change in behavior is due to the inclusion of floating-point numbers in hexadecimal notation without requiring that either a decimal point or the binary exponent be present.

66274 **RATIONALE**

66275 None.

66276 **FUTURE DIRECTIONS**

66277 None.

66278 **SEE ALSO**66279 *fscanf()*, *isspace()*, *localeconv()*, *setlocale()*, *strtol()*66280 XBD Chapter 7 (on page 135), *<float.h>*, *<stdlib.h>*66281 **CHANGE HISTORY**

66282 First released in Issue 1. Derived from Issue 1 of the SVID.

66283 **Issue 5**66284 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.66285 **Issue 6**

66286 Extensions beyond the ISO C standard are marked.

66287 The following new requirements on POSIX implementations derive from alignment with the  
66288 Single UNIX Specification:

- 66289 • In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is
- 66290 added if no conversion could be performed.

66291 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 66292 • The *strtod()* function is updated.
- 66293 • The *strtof()* and *strtold()* functions are added.
- 66294 • The DESCRIPTION is extensively revised.

66295 ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

66296 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/61 is applied, correcting the second  
66297 paragraph in the RETURN VALUE section. This change clarifies the sign of the return value.66298 **Issue 7**

66299 Austin Group Interpretation 1003.1-2001 #015 is applied.

66300 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0610 [302], XSH/TC1-2008/0611 [94],  
66301 and XSH/TC1-2008/0612 [105] are applied.

66302 **NAME**

66303 strtoimax, strtoumax — convert string to integer type

66304 **SYNOPSIS**

66305 #include &lt;inttypes.h&gt;

66306 intmax\_t strtoimax(const char \*restrict nptr, char \*\*restrict endptr,  
66307 int base);66308 uintmax\_t strtoumax(const char \*restrict nptr, char \*\*restrict endptr,  
66309 int base);66310 **DESCRIPTION**66311 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
66312 conflict between the requirements described here and the ISO C standard is unintentional. This  
66313 volume of POSIX.1-2008 defers to the ISO C standard.66314 These functions shall be equivalent to the *strtol()*, *strtoll()*, *strtoul()*, and *strtoull()* functions,  
66315 except that the initial portion of the string shall be converted to **intmax\_t** and **uintmax\_t**  
66316 representation, respectively.66317 **RETURN VALUE**

66318 These functions shall return the converted value, if any.

66319 CX If no conversion could be performed, zero shall be returned and *errno* may be set to [EINVAL].66320 CX If the value of *base* is not supported, 0 shall be returned and *errno* shall be set to [EINVAL].66321 If the correct value is outside the range of representable values, {INTMAX\_MAX},  
66322 {INTMAX\_MIN}, or {UINTMAX\_MAX} shall be returned (according to the return type and sign  
66323 of the value, if any), and *errno* shall be set to [ERANGE].66324 **ERRORS**

66325 These functions shall fail if:

66326 CX [EINVAL] The value of *base* is not supported.

66327 [ERANGE] The value to be returned is not representable.

66328 These functions may fail if:

66329 [EINVAL] No conversion could be performed.

66330 **EXAMPLES**

66331 None.

66332 **APPLICATION USAGE**66333 Since the value of *\*endptr* is unspecified if the value of *base* is not supported, applications should  
66334 either ensure that *base* has a supported value (0 or between 2 and 36) before the call, or check for  
66335 an [EINVAL] error before examining *\*endptr*.66336 **RATIONALE**

66337 None.

66338 **FUTURE DIRECTIONS**

66339 None.

66340 **SEE ALSO**66341 *strtol()*, *strtoul()*

66342 XBD &lt;inttypes.h&gt;

**CHANGE HISTORY**

66343  
66344 First released in Issue 6. Derived from the ISO/IEC 9899: 1999 standard.

**Issue 7**

66345  
66346 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0613 [453] and XSH/TC1-2008/0614  
66347 [453] are applied.

66348 **NAME**

66349        strtok, strtok\_r — split string into tokens

66350 **SYNOPSIS**

66351        #include &lt;string.h&gt;

66352        char \*strtok(char \*restrict *s*, const char \*restrict *sep*);66353 CX       char \*strtok\_r(char \*restrict *s*, const char \*restrict *sep*,66354           char \*\*restrict *state*);66355 **DESCRIPTION**

66356 CX       For *strtok()*: The functionality described on this reference page is aligned with the ISO C  
 66357 standard. Any conflict between the requirements described here and the ISO C standard is  
 66358 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

66359        A sequence of calls to *strtok()* breaks the string pointed to by *s* into a sequence of tokens, each of  
 66360 which is delimited by a byte from the string pointed to by *sep*. The first call in the sequence has *s*  
 66361 as its first argument, and is followed by calls with a null pointer as their first argument. The  
 66362 separator string pointed to by *sep* may be different from call to call.

66363        The first call in the sequence searches the string pointed to by *s* for the first byte that is *not*  
 66364 contained in the current separator string pointed to by *sep*. If no such byte is found, then there  
 66365 are no tokens in the string pointed to by *s* and *strtok()* shall return a null pointer. If such a byte is  
 66366 found, it is the start of the first token.

66367        The *strtok()* function then searches from there for a byte that *is* contained in the current separator  
 66368 string. If no such byte is found, the current token extends to the end of the string pointed to by *s*,  
 66369 and subsequent searches for a token shall return a null pointer. If such a byte is found, it is  
 66370 overwritten by a NUL character, which terminates the current token. The *strtok()* function saves  
 66371 a pointer to the following byte, from which the next search for a token shall start.

66372        Each subsequent call, with a null pointer as the value of the first argument, starts searching from  
 66373 the saved pointer and behaves as described above.

66374        The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls  
 66375 *strtok()*.

66376 CX       The *strtok()* function need not be thread-safe.

66377        The *strtok\_r()* function shall be equivalent to *strtok()*, except that *strtok\_r()* shall be thread-safe  
 66378 and the argument *state* points to a user-provided pointer that allows *strtok\_r()* to maintain state  
 66379 between calls which scan the same string. The application shall ensure that the pointer pointed  
 66380 to by *state* is unique for each string (*s*) being processed concurrently by *strtok\_r()* calls. The  
 66381 application need not initialize the pointer pointed to by *state* to any particular value. The  
 66382 implementation shall not update the pointer pointed to by *state* to point (directly or indirectly) to  
 66383 resources, other than within the string *s*, that need to be freed or released by the caller.

66384 **RETURN VALUE**

66385        Upon successful completion, *strtok()* shall return a pointer to the first byte of a token. Otherwise,  
 66386 if there is no token, *strtok()* shall return a null pointer.

66387 CX       The *strtok\_r()* function shall return a pointer to the token found, or a null pointer when no token  
 66388 is found.



**ERRORS**

No errors are defined.

**EXAMPLES****Searching for Word Separators**

The following example searches for tokens separated by <space> characters.

```
#include <string.h>
...
char *token;
char line[] = "LINE TO BE SEPARATED";
char *search = " ";

/* Token will point to "LINE". */
token = strtok(line, search);

/* Token will point to "TO". */
token = strtok(NULL, search);
```

**Find First two Fields in a Buffer**

The following example uses *strtok()* to find two character strings (a key and data associated with that key) separated by any combination of <space>, <tab>, or <newline> characters at the start of the array of characters pointed to by *buffer*.

```
#include <string.h>
...
char    *buffer;
...
struct element {
    char *key;
    char *data;
} e;
...
// Load the buffer...
...
// Get the key and its data...
e.key = strtok(buffer, " \t\n");
e.data = strtok(NULL, " \t\n");
// Process the rest of the contents of the buffer...
...
```

**APPLICATION USAGE**

Note that if *sep* is the empty string, *strtok()* and *strtok\_r()* return a pointer to the remainder of the string being tokenized. +

The *strtok\_r()* function is thread-safe and stores its state in a user-supplied buffer instead of possibly using a static data area that may be overwritten by an unrelated call from another thread. +

**RATIONALE**

The *strtok()* function searches for a separator string within a larger string. It returns a pointer to the last substring between separator strings. This function uses static storage to keep track of the current string position between calls. The new function, *strtok\_r()*, takes an additional

66433 argument, *state*, to keep track of the current position in the string.

66434 **FUTURE DIRECTIONS**

66435 None.

66436 **SEE ALSO**

66437 XBD <string.h>

66438 **CHANGE HISTORY**

66439 First released in Issue 1. Derived from Issue 1 of the SVID.

66440 **Issue 5**

66441 The *strtok\_r()* function is included for alignment with the POSIX Threads Extension.

66442 A note indicating that the *strtok()* function need not be reentrant is added to the DESCRIPTION.

66443 **Issue 6**

66444 Extensions beyond the ISO C standard are marked.

66445 The *strtok\_r()* function is marked as part of the Thread-Safe Functions option.

66446 In the DESCRIPTION, the note about reentrancy is expanded to cover thread-safety.

66447 The APPLICATION USAGE section is updated to include a note on the thread-safe function and  
66448 its avoidance of possibly using a static data area.

66449 The **restrict** keyword is added to the *strtok()* and *strtok\_r()* prototypes for alignment with the  
66450 ISO/IEC 9899:1999 standard.

66451 **Issue 7**

66452 Austin Group Interpretation 1003.1-2001 #156 is applied.

66453 SD5-XSH-ERN-235 is applied, correcting an example.

66454 The *strtok\_r()* function is moved from the Thread-Safe Functions option to the Base.

66455 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0615 [177] is applied.

## NAME

strtol, strtoll — convert a string to a long integer

## SYNOPSIS

```
#include <stdlib.h>
```

```
long strtol(const char *restrict nptr, char **restrict endptr, int base);
long long strtoll(const char *restrict nptr, char **restrict endptr,
                  int base)
```

## DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

These functions shall convert the initial portion of the string pointed to by *nptr* to a type **long** and **long long** representation, respectively. First, they decompose the input string into three parts:

1. An initial, possibly empty, sequence of white-space characters (as specified by *isspace()*)
2. A subject sequence interpreted as an integer represented in some radix determined by the value of *base*
3. A final string of one or more unrecognized characters, including the terminating NUL character of the input string.

Then they shall attempt to convert the subject sequence to an integer, and return the result.

If the value of *base* is 0, the expected form of the subject sequence is that of a decimal constant, octal constant, or hexadecimal constant, any of which may be preceded by a '+' or '-' sign. A decimal constant begins with a non-zero digit, and consists of a sequence of decimal digits. An octal constant consists of the prefix '0' optionally followed by a sequence of the digits '0' to '7' only. A hexadecimal constant consists of the prefix 0x or 0X followed by a sequence of the decimal digits and letters 'a' (or 'A') to 'f' (or 'F') with values 10 to 15 respectively.

If the value of *base* is between 2 and 36, the expected form of the subject sequence is a sequence of letters and digits representing an integer with the radix specified by *base*, optionally preceded by a '+' or '-' sign. The letters from 'a' (or 'A') to 'z' (or 'Z') inclusive are ascribed the values 10 to 35; only letters whose ascribed values are less than that of *base* are permitted. If the value of *base* is 16, the characters 0x or 0X may optionally precede the sequence of letters and digits, following the sign if present.

The subject sequence is defined as the longest initial subsequence of the input string, starting with the first non-white-space character that is of the expected form. The subject sequence shall contain no characters if the input string is empty or consists entirely of white-space characters, or if the first non-white-space character is other than a sign or a permissible letter or digit.

If the subject sequence has the expected form and the value of *base* is 0, the sequence of characters starting with the first digit shall be interpreted as an integer constant. If the subject sequence has the expected form and the value of *base* is between 2 and 36, it shall be used as the base for conversion, ascribing to each letter its value as given above. If the subject sequence begins with a <hyphen-minus>, the value resulting from the conversion shall be negated. A pointer to the final string shall be stored in the object pointed to by *endptr*, provided that *endptr* is not a null pointer.

CX In other than the C or POSIX locale, additional locale-specific subject sequence forms may be accepted.

66501 If the subject sequence is empty or does not have the expected form, no conversion is performed;  
 66502 the value of *nptr* shall be stored in the object pointed to by *endptr*, provided that *endptr* is not a  
 66503 null pointer.

66504 These functions shall not change the setting of *errno* if successful.

66505 Since 0, {LONG\_MIN} or {LLONG\_MIN}, and {LONG\_MAX} or {LLONG\_MAX} are returned  
 66506 on error and are also valid returns on success, an application wishing to check for error  
 66507 situations should set *errno* to 0, then call *strtol()* or *strtoll()*, then check *errno*.

#### 66508 RETURN VALUE

66509 Upon successful completion, these functions shall return the converted value, if any. If no  
 66510 CX conversion could be performed, 0 shall be returned and *errno* may be set to [EINVAL].

66511 CX If the value of *base* is not supported, 0 shall be returned and *errno* shall be set to [EINVAL].

66512 If the correct value is outside the range of representable values, {LONG\_MIN}, {LONG\_MAX},  
 66513 {LLONG\_MIN}, or {LLONG\_MAX} shall be returned (according to the sign of the value), and  
 66514 *errno* set to [ERANGE].

#### 66515 ERRORS

66516 These functions shall fail if:

66517 CX [EINVAL] The value of *base* is not supported.

66518 [ERANGE] The value to be returned is not representable.

66519 These functions may fail if:

66520 [EINVAL] No conversion could be performed.

#### 66521 EXAMPLES

66522 None.

#### 66523 APPLICATION USAGE

66524 Since the value of *\*endptr* is unspecified if the value of *base* is not supported, applications should  
 66525 either ensure that *base* has a supported value (0 or between 2 and 36) before the call, or check for  
 66526 an [EINVAL] error before examining *\*endptr*.

#### 66527 RATIONALE

66528 None.

#### 66529 FUTURE DIRECTIONS

66530 None.

#### 66531 SEE ALSO

66532 *fscanf()*, *isalpha()*, *strtod()*

66533 XBD <stdlib.h>

#### 66534 CHANGE HISTORY

66535 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 66536 Issue 5

66537 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

#### 66538 Issue 6

66539 Extensions beyond the ISO C standard are marked.

66540 The following new requirements on POSIX implementations derive from alignment with the  
 66541 Single UNIX Specification:

- In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is added if no conversion could be performed.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *strtol()* prototype is updated.
- The *strtoll()* function is added.

#### Issue 7

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0616 [453], XSH/TC1-2008/0617 [105], XSH/TC1-2008/0618 [453], XSH/TC1-2008/0619 [453], and XSH/TC1-2008/0620 [453] are applied.

66551 **NAME**

66552        strtold — convert a string to a double-precision number

66553 **SYNOPSIS**

66554        #include &lt;stdlib.h&gt;

66555        long double strtold(const char \*restrict *nptr*, char \*\*restrict *endptr*);66556 **DESCRIPTION**66557        Refer to *strtod()*.

66558 **NAME**

66559        strtoll — convert a string to a long integer

66560 **SYNOPSIS**

66561        #include &lt;stdlib.h&gt;

66562        long long strtoll(const char \*restrict *str*, char \*\*restrict *endptr*,  
66563                          int *base*);66564 **DESCRIPTION**66565        Refer to *strtol()*.

## NAME

strtol, strtoull — convert a string to an unsigned long

## SYNOPSIS

```
#include <stdlib.h>

unsigned long strtol(const char *restrict str,
                    char **restrict endptr, int base);
unsigned long long strtoull(const char *restrict str,
                           char **restrict endptr, int base);
```

## DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

These functions shall convert the initial portion of the string pointed to by *str* to a type **unsigned long** and **unsigned long long** representation, respectively. First, they decompose the input string into three parts:

1. An initial, possibly empty, sequence of white-space characters (as specified by *isspace()*)
2. A subject sequence interpreted as an integer represented in some radix determined by the value of *base*
3. A final string of one or more unrecognized characters, including the terminating NUL character of the input string

Then they shall attempt to convert the subject sequence to an unsigned integer, and return the result.

If the value of *base* is 0, the expected form of the subject sequence is that of a decimal constant, octal constant, or hexadecimal constant, any of which may be preceded by a '+' or '-' sign. A decimal constant begins with a non-zero digit, and consists of a sequence of decimal digits. An octal constant consists of the prefix '0' optionally followed by a sequence of the digits '0' to '7' only. A hexadecimal constant consists of the prefix 0x or 0X followed by a sequence of the decimal digits and letters 'a' (or 'A') to 'f' (or 'F') with values 10 to 15 respectively.

If the value of *base* is between 2 and 36, the expected form of the subject sequence is a sequence of letters and digits representing an integer with the radix specified by *base*, optionally preceded by a '+' or '-' sign. The letters from 'a' (or 'A') to 'z' (or 'Z') inclusive are ascribed the values 10 to 35; only letters whose ascribed values are less than that of *base* are permitted. If the value of *base* is 16, the characters 0x or 0X may optionally precede the sequence of letters and digits, following the sign if present.

The subject sequence is defined as the longest initial subsequence of the input string, starting with the first non-white-space character that is of the expected form. The subject sequence shall contain no characters if the input string is empty or consists entirely of white-space characters, or if the first non-white-space character is other than a sign or a permissible letter or digit.

If the subject sequence has the expected form and the value of *base* is 0, the sequence of characters starting with the first digit shall be interpreted as an integer constant. If the subject sequence has the expected form and the value of *base* is between 2 and 36, it shall be used as the base for conversion, ascribing to each letter its value as given above. If the subject sequence begins with a <hyphen-minus>, the value resulting from the conversion shall be negated. A pointer to the final string shall be stored in the object pointed to by *endptr*, provided that *endptr* is not a null pointer.



66611 CX In other than the C or POSIX locale, additional locale-specific subject sequence forms may be  
66612 accepted.

66613 If the subject sequence is empty or does not have the expected form, no conversion shall be  
66614 performed; the value of *str* shall be stored in the object pointed to by *endptr*, provided that *endptr*  
66615 is not a null pointer.

66616 These functions shall not change the setting of *errno* if successful.

66617 Since 0, {ULONG\_MAX}, and {ULLONG\_MAX} are returned on error and are also valid returns  
66618 on success, an application wishing to check for error situations should set *errno* to 0, then call  
66619 *strtoul()* or *strtoull()*, then check *errno*.

#### 66620 RETURN VALUE

66621 Upon successful completion, these functions shall return the converted value, if any. If no  
66622 CX conversion could be performed, 0 shall be returned and *errno* may be set to [EINVAL].

66623 CX If the value of *base* is not supported, 0 shall be returned and *errno* shall be set to [EINVAL].

66624 If the correct value is outside the range of representable values, {ULONG\_MAX} or  
66625 {ULLONG\_MAX} shall be returned and *errno* set to [ERANGE].

#### 66626 ERRORS

66627 These functions shall fail if:

66628 CX [EINVAL] The value of *base* is not supported.

66629 [ERANGE] The value to be returned is not representable.

66630 These functions may fail if:

66631 CX [EINVAL] No conversion could be performed.

#### 66632 EXAMPLES

66633 None.

#### 66634 APPLICATION USAGE

66635 Since the value of *\*endptr* is unspecified if the value of *base* is not supported, applications should  
66636 either ensure that *base* has a supported value (0 or between 2 and 36) before the call, or check for  
66637 an [EINVAL] error before examining *\*endptr*.

#### 66638 RATIONALE

66639 None.

#### 66640 FUTURE DIRECTIONS

66641 None.

#### 66642 SEE ALSO

66643 *fscanf()*, *isalpha()*, *strtod()*, *strtol()*

66644 XBD <stdlib.h>

#### 66645 CHANGE HISTORY

66646 First released in Issue 4. Derived from the ANSI C standard.

#### 66647 Issue 5

66648 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

66649 **Issue 6**

66650 Extensions beyond the ISO C standard are marked.

66651 The following new requirements on POSIX implementations derive from alignment with the  
66652 Single UNIX Specification:

- 66653 • The [EINVAL] error condition is added for when the value of *base* is not supported.
- 66654 • In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is  
66655 added if no conversion could be performed.

66656 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 66657 • The *strtoul()* prototype is updated.
- 66658 • The *strtoull()* function is added.

66659 **Issue 7**66660 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0621 [105], XSH/TC1-2008/0622 [453],  
66661 and XSH/TC1-2008/0623 [453] are applied.

66662 **NAME**

66663        strtoumax — convert a string to an integer type

66664 **SYNOPSIS**

66665        #include &lt;inttypes.h&gt;

66666        uintmax\_t strtoumax(const char \*restrict *nptr*, char \*\*restrict *endptr*,  
66667                            int *base*);66668 **DESCRIPTION**66669        Refer to *strtoimax()*.

## 66670 NAME

66671 strxfrm, strxfrm\_l — string transformation

## 66672 SYNOPSIS

66673 #include &lt;string.h&gt;

66674 size\_t strxfrm(char \*restrict s1, const char \*restrict s2, size\_t n);

66675 CX size\_t strxfrm\_l(char \*restrict s1, const char \*restrict s2,  
66676 size\_t n, locale\_t locale);

## 66677 DESCRIPTION

66678 CX For *strxfrm()*: The functionality described on this reference page is aligned with the ISO C  
66679 standard. Any conflict between the requirements described here and the ISO C standard is  
66680 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.66681 CX The *strxfrm()* and *strxfrm\_l()* functions shall transform the string pointed to by *s2* and place the  
66682 resulting string into the array pointed to by *s1*. The transformation is such that if *strcmp()* is  
66683 applied to two transformed strings, it shall return a value greater than, equal to, or less than 0,  
66684 CX corresponding to the result of *strcoll()* or *strcoll\_l()*, respectively, applied to the same two  
66685 CX original strings with the same locale. No more than *n* bytes are placed into the resulting array  
66686 pointed to by *s1*, including the terminating NUL character. If *n* is 0, *s1* is permitted to be a null  
66687 pointer. If copying takes place between objects that overlap, the behavior is undefined.66688 CX The *strxfrm()* and *strxfrm\_l()* functions shall not change the setting of *errno* if successful.66689 Since no return value is reserved to indicate an error, an application wishing to check for error  
66690 CX situations should set *errno* to 0, then call *strxfrm()* or *strxfrm\_l()*, then check *errno*.66691 CX The behavior is undefined if the *locale* argument to *strxfrm\_l()* is the special locale object  
66692 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

## 66693 RETURN VALUE

66694 CX Upon successful completion, *strxfrm()* and *strxfrm\_l()* shall return the length of the  
66695 transformed string (not including the terminating NUL character). If the value returned is *n* or  
66696 more, the contents of the array pointed to by *s1* are unspecified.66697 CX On error, *strxfrm()* and *strxfrm\_l()* may set *errno* but no return value is reserved to indicate an  
66698 error.

## 66699 ERRORS

66700 These functions may fail if:

66701 CX [EINVAL] The string pointed to by the *s2* argument contains characters outside the  
66702 domain of the collating sequence.

## 66703 EXAMPLES

66704 None.

## 66705 APPLICATION USAGE

66706 The transformation function is such that two transformed strings can be ordered by *strcmp()* as  
66707 appropriate to collating sequence information in the current locale (category *LC\_COLLATE*).66708 The fact that when *n* is 0 *s1* is permitted to be a null pointer is useful to determine the size of the  
66709 *s1* array prior to making the transformation.

66710 **RATIONALE**

66711 None.

66712 **FUTURE DIRECTIONS**

66713 None.

66714 **SEE ALSO**66715 *strcmp()*, *strcoll()*

66716 XBD &lt;string.h&gt;

66717 **CHANGE HISTORY**

66718 First released in Issue 3. Included for alignment with the ISO C standard.

66719 **Issue 5**66720 The DESCRIPTION is updated to indicate that *errno* does not change if the function is successful.66721 **Issue 6**

66722 Extensions beyond the ISO C standard are marked.

66723 The following new requirements on POSIX implementations derive from alignment with the  
66724 Single UNIX Specification:

- 66725
- In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is
- 
- 66726 added if no conversion could be performed.

66727 The *strxfrm()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.66728 **Issue 7**66729 The *strxfrm\_l()* function is added from The Open Group Technical Standard, 2006, Extended API  
66730 Set Part 4.66731 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0624 [283], XSH/TC1-2008/0625 [283],  
66732 and XSH/TC1-2008/0626 [302] are applied.

66733 **NAME**

66734 swab — swap bytes

66735 **SYNOPSIS**

```
66736 XSI #include <unistd.h>
66737 void swab(const void *restrict src, void *restrict dest,
66738          ssize_t nbytes);
```

66739 **DESCRIPTION**

66740 The *swab()* function shall copy *nbytes* bytes, which are pointed to by *src*, to the object pointed to  
66741 by *dest*, exchanging adjacent bytes. The *nbytes* argument should be even. If *nbytes* is odd, *swab()*  
66742 copies and exchanges *nbytes*–1 bytes and the disposition of the last byte is unspecified. If  
66743 copying takes place between objects that overlap, the behavior is undefined. If *nbytes* is  
66744 negative, *swab()* does nothing.

66745 **RETURN VALUE**

66746 None.

66747 **ERRORS**

66748 No errors are defined.

66749 **EXAMPLES**

66750 None.

66751 **APPLICATION USAGE**

66752 None.

66753 **RATIONALE**

66754 None.

66755 **FUTURE DIRECTIONS**

66756 None.

66757 **SEE ALSO**66758 XBD [<unistd.h>](#)66759 **CHANGE HISTORY**

66760 First released in Issue 1. Derived from Issue 1 of the SVID.

66761 **Issue 6**

66762 The **restrict** keyword is added to the *swab()* prototype for alignment with the  
66763 ISO/IEC 9899:1999 standard.

66764 **NAME**

66765 swprintf — print formatted wide-character output

66766 **SYNOPSIS**

66767 #include &lt;stdio.h&gt;

66768 #include &lt;wchar.h&gt;

66769 int swprintf(wchar\_t \*restrict ws, size\_t n,

66770 const wchar\_t \*restrict format, ...);

66771 **DESCRIPTION**66772 Refer to *fwprintf()*.

66773 **NAME**

66774 swscanf — convert formatted wide-character input

66775 **SYNOPSIS**

66776 #include &lt;stdio.h&gt;

66777 #include &lt;wchar.h&gt;

```
66778 int swscanf(const wchar_t *restrict ws,  
66779             const wchar_t *restrict format, ...);
```

66780 **DESCRIPTION**66781 Refer to *fwscanf()*.



66782 **NAME**66783        `symlink, symlinkat` — make a symbolic link |66784 **SYNOPSIS**66785        `#include <unistd.h>`66786        `int symlink(const char *path1, const char *path2);`66787 OH     `#include <fcntl.h>` +

66788 +

66789        `int symlinkat(const char *path1, int fd, const char *path2);`66790 **DESCRIPTION**

66791        The `symlink()` function shall create a symbolic link called *path2* that contains the string pointed to  
 66792        by *path1* (*path2* is the name of the symbolic link created, *path1* is the string contained in the  
 66793        symbolic link).

66794        The string pointed to by *path1* shall be treated only as a string and shall not be validated as a |  
 66795        pathname.

66796        If the `symlink()` function fails for any reason other than [EIO], any file named by *path2* shall be  
 66797        unaffected.

66798        If *path2* names a symbolic link, `symlink()` shall fail and set *errno* to [EEXIST].

66799        The symbolic link's user ID shall be set to the process' effective user ID. The symbolic link's  
 66800        group ID shall be set to the group ID of the parent directory or to the effective group ID of the  
 66801        process. Implementations shall provide a way to initialize the symbolic link's group ID to the  
 66802        group ID of the parent directory. Implementations may, but need not, provide an  
 66803        implementation-defined way to initialize the symbolic link's group ID to the effective group ID  
 66804        of the calling process.

66805        The values of the file mode bits for the created symbolic link are unspecified. All interfaces  
 66806        specified by POSIX.1-2008 shall behave as if the contents of symbolic links can always be read,  
 66807        except that the value of the file mode bits returned in the *st\_mode* field of the **stat** structure is  
 66808        unspecified.

66809        Upon successful completion, `symlink()` shall mark for update the last data access, last data  
 66810        modification, and last file status change timestamps of the symbolic link. Also, the last data  
 66811        modification and last file status change timestamps of the directory that contains the new entry  
 66812        shall be marked for update.

66813        The `symlinkat()` function shall be equivalent to the `symlink()` function except in the case where  
 66814        *path2* specifies a relative path. In this case the symbolic link is created relative to the directory  
 66815        associated with the file descriptor *fd* instead of the current working directory. If the access mode |  
 66816        of the open file description associated with the file descriptor is not O\_SEARCH, the function |  
 66817        shall check whether directory searches are permitted using the current permissions of the |  
 66818        directory underlying the file descriptor. If the access mode is O\_SEARCH, the function shall not |  
 66819        perform the check.

66820        If `symlinkat()` is passed the special value AT\_FDCWD in the *fd* parameter, the current working  
 66821        directory shall be used and the behavior shall be identical to a call to `symlink()`.

66822 **RETURN VALUE**

66823        Upon successful completion, these functions shall return 0. Otherwise, these functions shall  
 66824        return -1 and set *errno* to indicate the error.

**ERRORS**

These functions shall fail if:

[EACCES] Write permission is denied in the directory where the symbolic link is being created, or search permission is denied for a component of the path prefix of *path2*.

[EEXIST] The *path2* argument names an existing file.

[EIO] An I/O error occurs while reading from or writing to the file system.

[ELOOP] A loop exists in symbolic links encountered during resolution of the *path2* argument.

[ENAMETOOLONG]

The length of a component of the pathname specified by the *path2* argument is longer than {NAME\_MAX} or the length of the *path1* argument is longer than {SYMLINK\_MAX}.

[ENOENT] A component of the path prefix of *path2* does not name an existing file or *path2* is an empty string.

[ENOENT] or [ENOTDIR]

The *path2* argument contains at least one non-`<slash>` character and ends with one or more trailing `<slash>` characters. If *path2* without the trailing `<slash>` characters would name an existing file, an [ENOENT] error shall not occur.

[ENOSPC] The directory in which the entry for the new symbolic link is being placed cannot be extended because no space is left on the file system containing the directory, or the new symbolic link cannot be created because no space is left on the file system which shall contain the link, or the file system is out of file-allocation resources.

[ENOTDIR] A component of the path prefix of *path2* names an existing file that is neither a directory nor a symbolic link to a directory.

[EROFS] The new symbolic link would reside on a read-only file system.

The *symlinkat*( ) function shall fail if:

[EACCES] The access mode of the open file description associated with *fd* is not O\_SEARCH and the permissions of the directory underlying *fd* do not permit directory searches.

[EBADF] The *path2* argument does not specify an absolute path and the *fd* argument is neither AT\_FDCWD nor a valid file descriptor open for reading or searching.

[ENOTDIR] The *path2* argument is not an absolute path and *fd* is a file descriptor associated with a non-directory file.

These functions may fail if:

[ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during resolution of the *path2* argument.

[ENAMETOOLONG]

The length of the *path2* argument exceeds {PATH\_MAX} or pathname resolution of a symbolic link in the *path2* argument produced an intermediate result with a length that exceeds {PATH\_MAX}.

**EXAMPLES**

None.

**APPLICATION USAGE**

Like a hard link, a symbolic link allows a file to have multiple logical names. The presence of a hard link guarantees the existence of a file, even after the original name has been removed. A symbolic link provides no such assurance; in fact, the file named by the *path1* argument need not exist when the link is created. A symbolic link can cross file system boundaries.

Normal permission checks are made on each component of the symbolic link pathname during its resolution.

**RATIONALE**

The purpose of the *symlinkat()* function is to create symbolic links in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *symlink()*, resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *symlinkat()* function it can be guaranteed that the created symbolic link is located relative to the desired directory.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*fdopendir()*, *fstatat()*, *lchown()*, *link()*, *open()*, *readlink()*, *rename()*, *unlink()*

XBD <fcntl.h>, <unistd.h>

**CHANGE HISTORY**

First released in Issue 4, Version 2.

**Issue 5**

Moved from X/OPEN UNIX extension to BASE.

**Issue 6**

The following changes were made to align with the IEEE P1003.1a draft standard:

- The DESCRIPTION text is updated.
- The [ELOOP] optional error condition is added.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #143 is applied.

The *symlinkat()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 2.

Additions have been made describing how *symlink()* sets the user and group IDs and file mode of the symbolic link, and its effect on timestamps.

Changes are made to allow a directory to be opened for searching.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0627 [146,428], XSH/TC1-2008/0628 [461], XSH/TC1-2008/0629 [146,428], XSH/TC1-2008/0630 [146,428,436], XSH/TC1-2008/0631 [324], XSH/TC1-2008/0632 [278], XSH/TC1-2008/0633 [278], and XSH/TC1-2008/0634 [151] are applied.

66906 **NAME**

66907 sync — schedule file system updates

66908 **SYNOPSIS**

```
66909 XSI    #include <unistd.h>  
66910      void sync(void);
```

66911 **DESCRIPTION**

66912 The *sync()* function shall cause all information in memory that updates file systems to be  
66913 scheduled for writing out to all file systems.

66914 The writing, although scheduled, is not necessarily complete upon return from *sync()*.

66915 **RETURN VALUE**

66916 The *sync()* function shall not return a value.

66917 **ERRORS**

66918 No errors are defined.

66919 **EXAMPLES**

66920 None.

66921 **APPLICATION USAGE**

66922 None.

66923 **RATIONALE**

66924 None.

66925 **FUTURE DIRECTIONS**

66926 None.

66927 **SEE ALSO**

66928 *fsync()*

66929 XBD <unistd.h>

66930 **CHANGE HISTORY**

66931 First released in Issue 4, Version 2.

66932 **Issue 5**

66933 Moved from X/OPEN UNIX extension to BASE.

66934 **NAME**

66935 sysconf — get configurable system variables

66936 **SYNOPSIS**

66937 #include &lt;unistd.h&gt;

66938 long sysconf(int name);

66939 **DESCRIPTION**

66940 The *sysconf()* function provides a method for the application to determine the current value of a  
 66941 configurable system limit or option (*variable*). The implementation shall support all of the  
 66942 variables listed in the following table and may support others.

66943 The *name* argument represents the system variable to be queried. The following table lists the  
 66944 minimal set of system variables from <limits.h> or <unistd.h> that can be returned by *sysconf()*,  
 66945 and the symbolic constants defined in <unistd.h> that are the corresponding values used for  
 66946 *name*.

| Variable                                                                    | Value of Name                    |
|-----------------------------------------------------------------------------|----------------------------------|
| {AIO_LISTIO_MAX}                                                            | _SC_AIO_LISTIO_MAX               |
| {AIO_MAX}                                                                   | _SC_AIO_MAX                      |
| {AIO_PRIO_DELTA_MAX}                                                        | _SC_AIO_PRIO_DELTA_MAX           |
| {ARG_MAX}                                                                   | _SC_ARG_MAX                      |
| {ATEXIT_MAX}                                                                | _SC_ATEXIT_MAX                   |
| {BC_BASE_MAX}                                                               | _SC_BC_BASE_MAX                  |
| {BC_DIM_MAX}                                                                | _SC_BC_DIM_MAX                   |
| {BC_SCALE_MAX}                                                              | _SC_BC_SCALE_MAX                 |
| {BC_STRING_MAX}                                                             | _SC_BC_STRING_MAX                |
| {CHILD_MAX}                                                                 | _SC_CHILD_MAX                    |
| Clock ticks/second                                                          | _SC_CLK_TCK                      |
| {COLL_WEIGHTS_MAX}                                                          | _SC_COLL_WEIGHTS_MAX             |
| {DELAYTIMER_MAX}                                                            | _SC_DELAYTIMER_MAX               |
| {EXPR_NEST_MAX}                                                             | _SC_EXPR_NEST_MAX                |
| {HOST_NAME_MAX}                                                             | _SC_HOST_NAME_MAX                |
| {IOV_MAX}                                                                   | _SC_IOV_MAX                      |
| {LINE_MAX}                                                                  | _SC_LINE_MAX                     |
| {LOGIN_NAME_MAX}                                                            | _SC_LOGIN_NAME_MAX               |
| {NGROUPS_MAX}                                                               | _SC_NGROUPS_MAX                  |
| Initial size of <i>getgrgid_r()</i> and<br><i>getgrnam_r()</i> data buffers | _SC_GETGR_R_SIZE_MAX             |
| Initial size of <i>getpwuid_r()</i> and<br><i>getpwnam_r()</i> data buffers | _SC_GETPW_R_SIZE_MAX             |
| {MQ_OPEN_MAX}                                                               | _SC_MQ_OPEN_MAX                  |
| {MQ_PRIO_MAX}                                                               | _SC_MQ_PRIO_MAX                  |
| {OPEN_MAX}                                                                  | _SC_OPEN_MAX                     |
| {PAGE_SIZE}                                                                 | _SC_PAGE_SIZE                    |
| {PAGESIZE}                                                                  | _SC_PAGESIZE                     |
| {PTHREAD_DESTRUCTOR_ITERATIONS}                                             | _SC_THREAD_DESTRUCTOR_ITERATIONS |
| {PTHREAD_KEYS_MAX}                                                          | _SC_THREAD_KEYS_MAX              |
| {PTHREAD_STACK_MIN}                                                         | _SC_THREAD_STACK_MIN             |
| {PTHREAD_THREADS_MAX}                                                       | _SC_THREAD_THREADS_MAX           |
| {RE_DUP_MAX}                                                                | _SC_RE_DUP_MAX                   |
| {RTSIG_MAX}                                                                 | _SC_SIGRTMAX                     |

|       | Variable                          | Value of Name                  |
|-------|-----------------------------------|--------------------------------|
| 66982 |                                   |                                |
| 66983 | {SEM_NSEMS_MAX}                   | _SC_SEM_NSEMS_MAX              |
| 66984 | {SEM_VALUE_MAX}                   | _SC_SEM_VALUE_MAX              |
| 66985 | {SIGQUEUE_MAX}                    | _SC_SIGQUEUE_MAX               |
| 66986 | {STREAM_MAX}                      | _SC_STREAM_MAX                 |
| 66987 | {SYMLOOP_MAX}                     | _SC_SYMLOOP_MAX                |
| 66988 | {TIMER_MAX}                       | _SC_TIMER_MAX                  |
| 66989 | {TTY_NAME_MAX}                    | _SC_TTY_NAME_MAX               |
| 66990 | {TZNAME_MAX}                      | _SC_TZNAME_MAX                 |
| 66991 | _POSIX_ADVISORY_INFO              | _SC_ADVISORY_INFO              |
| 66992 | _POSIX_BARRIERS                   | _SC_BARRIERS                   |
| 66993 | _POSIX_ASYNCHRONOUS_IO            | _SC_ASYNCHRONOUS_IO            |
| 66994 | _POSIX_CLOCK_SELECTION            | _SC_CLOCK_SELECTION            |
| 66995 | _POSIX_CPUTIME                    | _SC_CPUTIME                    |
| 66996 | _POSIX_FSYNC                      | _SC_FSYNC                      |
| 66997 | _POSIX_IPV6                       | _SC_IPV6                       |
| 66998 | _POSIX_JOB_CONTROL                | _SC_JOB_CONTROL                |
| 66999 | _POSIX_MAPPED_FILES               | _SC_MAPPED_FILES               |
| 67000 | _POSIX_MEMLOCK                    | _SC_MEMLOCK                    |
| 67001 | _POSIX_MEMLOCK_RANGE              | _SC_MEMLOCK_RANGE              |
| 67002 | _POSIX_MEMORY_PROTECTION          | _SC_MEMORY_PROTECTION          |
| 67003 | _POSIX_MESSAGE_PASSING            | _SC_MESSAGE_PASSING            |
| 67004 | _POSIX_MONOTONIC_CLOCK            | _SC_MONOTONIC_CLOCK            |
| 67005 | _POSIX_PRIORITIZED_IO             | _SC_PRIORITIZED_IO             |
| 67006 | _POSIX_PRIORITY_SCHEDULING        | _SC_PRIORITY_SCHEDULING        |
| 67007 | _POSIX_RAW_SOCKETS                | _SC_RAW_SOCKETS                |
| 67008 | _POSIX_READER_WRITER_LOCKS        | _SC_READER_WRITER_LOCKS        |
| 67009 | _POSIX_REALTIME_SIGNALS           | _SC_REALTIME_SIGNALS           |
| 67010 | _POSIX_REGEX                      | _SC_REGEX                      |
| 67011 | _POSIX_SAVED_IDS                  | _SC_SAVED_IDS                  |
| 67012 | _POSIX_SEMAPHORES                 | _SC_SEMAPHORES                 |
| 67013 | _POSIX_SHARED_MEMORY_OBJECTS      | _SC_SHARED_MEMORY_OBJECTS      |
| 67014 | _POSIX_SHELL                      | _SC_SHELL                      |
| 67015 | _POSIX_SPAWN                      | _SC_SPAWN                      |
| 67016 | _POSIX_SPIN_LOCKS                 | _SC_SPIN_LOCKS                 |
| 67017 | _POSIX_SPORADIC_SERVER            | _SC_SPORADIC_SERVER            |
| 67018 | _POSIX_SS_REPL_MAX                | _SC_SS_REPL_MAX                |
| 67019 | _POSIX_SYNCHRONIZED_IO            | _SC_SYNCHRONIZED_IO            |
| 67020 | _POSIX_THREAD_ATTR_STACKADDR      | _SC_THREAD_ATTR_STACKADDR      |
| 67021 | _POSIX_THREAD_ATTR_STACKSIZE      | _SC_THREAD_ATTR_STACKSIZE      |
| 67022 | _POSIX_THREAD_CPUTIME             | _SC_THREAD_CPUTIME             |
| 67023 | _POSIX_THREAD_PRIO_INHERIT        | _SC_THREAD_PRIO_INHERIT        |
| 67024 | _POSIX_THREAD_PRIO_PROTECT        | _SC_THREAD_PRIO_PROTECT        |
| 67025 | _POSIX_THREAD_PRIORITY_SCHEDULING | _SC_THREAD_PRIORITY_SCHEDULING |
| 67026 | _POSIX_THREAD_PROCESS_SHARED      | _SC_THREAD_PROCESS_SHARED      |
| 67027 | _POSIX_THREAD_ROBUST_PRIO_INHERIT | _SC_THREAD_ROBUST_PRIO_INHERIT |
| 67028 | _POSIX_THREAD_ROBUST_PRIO_PROTECT | _SC_THREAD_ROBUST_PRIO_PROTECT |
| 67029 | _POSIX_THREAD_SAFE_FUNCTIONS      | _SC_THREAD_SAFE_FUNCTIONS      |
| 67030 | _POSIX_THREAD_SPORADIC_SERVER     | _SC_THREAD_SPORADIC_SERVER     |
| 67031 | _POSIX_THREADS                    | _SC_THREADS                    |
| 67032 | _POSIX_TIMEOUTS                   | _SC_TIMEOUTS                   |

|       | Variable                    | Value of Name            |
|-------|-----------------------------|--------------------------|
| 67033 |                             |                          |
| 67034 | _POSIX_TIMERS               | _SC_TIMERS               |
| 67035 | _POSIX_TRACE                | _SC_TRACE                |
| 67036 | _POSIX_TRACE_EVENT_FILTER   | _SC_TRACE_EVENT_FILTER   |
| 67037 | _POSIX_TRACE_EVENT_NAME_MAX | _SC_TRACE_EVENT_NAME_MAX |
| 67038 | _POSIX_TRACE_INHERIT        | _SC_TRACE_INHERIT        |
| 67039 | _POSIX_TRACE_LOG            | _SC_TRACE_LOG            |
| 67040 | _POSIX_TRACE_NAME_MAX       | _SC_TRACE_NAME_MAX       |
| 67041 | _POSIX_TRACE_SYS_MAX        | _SC_TRACE_SYS_MAX        |
| 67042 | _POSIX_TRACE_USER_EVENT_MAX | _SC_TRACE_USER_EVENT_MAX |
| 67043 | _POSIX_TYPED_MEMORY_OBJECTS | _SC_TYPED_MEMORY_OBJECTS |
| 67044 | _POSIX_VERSION              | _SC_VERSION              |
| 67045 | _POSIX_V7_ILP32_OFF32       | _SC_V7_ILP32_OFF32       |
| 67046 | _POSIX_V7_ILP32_OFFBIG      | _SC_V7_ILP32_OFFBIG      |
| 67047 | _POSIX_V7_LP64_OFF64        | _SC_V7_LP64_OFF64        |
| 67048 | _POSIX_V7_LPBIG_OFFBIG      | _SC_V7_LPBIG_OFFBIG      |



| Variable                | Value of Name              |
|-------------------------|----------------------------|
| _POSIX_V6_ILP32_OFF32   | _SC_V6_ILP32_OFF32         |
| _POSIX_V6_ILP32_OFFBIG  | _SC_V6_ILP32_OFFBIG        |
| _POSIX_V6_LP64_OFF64    | _SC_V6_LP64_OFF64          |
| _POSIX_V6_LPBIG_OFFBIG  | _SC_V6_LPBIG_OFFBIG        |
| _POSIX2_C_BIND          | _SC_2_C_BIND               |
| _POSIX2_C_DEV           | _SC_2_C_DEV                |
| _POSIX2_CHAR_TERM       | _SC_2_CHAR_TERM            |
| _POSIX2_FORT_DEV        | _SC_2_FORT_DEV             |
| _POSIX2_FORT_RUN        | _SC_2_FORT_RUN             |
| _POSIX2_LOCALEDEF       | _SC_2_LOCALEDEF            |
| _POSIX2_PBS             | _SC_2_PBS                  |
| _POSIX2_PBS_ACCOUNTING  | _SC_2_PBS_ACCOUNTING       |
| _POSIX2_PBS_CHECKPOINT  | _SC_2_PBS_CHECKPOINT       |
| _POSIX2_PBS_LOCATE      | _SC_2_PBS_LOCATE           |
| _POSIX2_PBS_MESSAGE     | _SC_2_PBS_MESSAGE          |
| _POSIX2_PBS_TRACK       | _SC_2_PBS_TRACK            |
| _POSIX2_SW_DEV          | _SC_2_SW_DEV               |
| _POSIX2_UPE             | _SC_2_UPE                  |
| _POSIX2_VERSION         | _SC_2_VERSION              |
| _XOPEN_CRYPT            | _SC_XOPEN_CRYPT            |
| _XOPEN_ENH_I18N         | _SC_XOPEN_ENH_I18N         |
| _XOPEN_REALTIME         | _SC_XOPEN_REALTIME         |
| _XOPEN_REALTIME_THREADS | _SC_XOPEN_REALTIME_THREADS |
| _XOPEN_SHM              | _SC_XOPEN_SHM              |
| _XOPEN_STREAMS          | _SC_XOPEN_STREAMS          |
| _XOPEN_UNIX             | _SC_XOPEN_UNIX             |
| _XOPEN_UUCP             | _SC_XOPEN_UUCP             |
| _XOPEN_VERSION          | _SC_XOPEN_VERSION          |

**RETURN VALUE**

If *name* is an invalid value, *sysconf()* shall return  $-1$  and set *errno* to indicate the error. If the variable corresponding to *name* is described in **<limits.h>** as a maximum or minimum value and the variable has no limit, *sysconf()* shall return  $-1$  without changing the value of *errno*. Note that indefinite limits do not imply infinite limits; see **<limits.h>**.

Otherwise, *sysconf()* shall return the current variable value on the system. The value returned shall not be more restrictive than the corresponding value described to the application when it was compiled with the implementation's **<limits.h>** or **<unistd.h>**. The value shall not change during the lifetime of the calling process, except that *sysconf(\_SC\_OPEN\_MAX)* may return different values before and after a call to *setrlimit()* which changes the RLIMIT\_NOFILE soft limit.

If the variable corresponding to *name* is dependent on an unsupported option, the results are unspecified.

**ERRORS**

The *sysconf()* function shall fail if:

[EINVAL] The value of the *name* argument is invalid.



**EXAMPLES**

None.

**APPLICATION USAGE**

As `-1` is a permissible return value in a successful situation, an application wishing to check for error situations should set `errno` to 0, then call `sysconf()`, and, if it returns `-1`, check to see if `errno` is non-zero.

Application developers should check whether an option, such as `_POSIX_TRACE`, is supported prior to obtaining and using values for related variables, such as `_POSIX_TRACE_NAME_MAX`.

**RATIONALE**

This functionality was added in response to requirements of application developers and of system vendors who deal with many international system configurations. It is closely related to `pathconf()` and `fpathconf()`.

Although a conforming application can run on all systems by never demanding more resources than the minimum values published in this volume of POSIX.1-2008, it is useful for that application to be able to use the actual value for the quantity of a resource available on any given system. To do this, the application makes use of the value of a symbolic constant in `<limits.h>` or `<unistd.h>`.

However, once compiled, the application must still be able to cope if the amount of resource available is increased. To that end, an application may need a means of determining the quantity of a resource, or the presence of an option, at execution time.

Two examples are offered:

1. Applications may wish to act differently on systems with or without job control. Applications vendors who wish to distribute only a single binary package to all instances of a computer architecture would be forced to assume job control is never available if it were to rely solely on the `<unistd.h>` value published in this volume of POSIX.1-2008.
2. International applications vendors occasionally require knowledge of the number of clock ticks per second. Without these facilities, they would be required to either distribute their applications partially in source form or to have 50 Hz and 60 Hz versions for the various countries in which they operate.

It is the knowledge that many applications are actually distributed widely in executable form that leads to this facility. If limited to the most restrictive values in the headers, such applications would have to be prepared to accept the most limited environments offered by the smallest microcomputers. Although this is entirely portable, there was a consensus that they should be able to take advantage of the facilities offered by large systems, without the restrictions associated with source and object distributions.

During the discussions of this feature, it was pointed out that it is almost always possible for an application to discern what a value might be at runtime by suitably testing the various functions themselves. And, in any event, it could always be written to adequately deal with error returns from the various functions. In the end, it was felt that this imposed an unreasonable level of complication and sophistication on the application developer.

This runtime facility is not meant to provide ever-changing values that applications have to check multiple times. The values are seen as changing no more frequently than once per system initialization, such as by a system administrator or operator with an automatic configuration program. This volume of POSIX.1-2008 specifies that they shall not change within the lifetime of the process.

Some values apply to the system overall and others vary at the file system or directory level. The

latter are described in *fpathconf()*.

Note that all values returned must be expressible as integers. String values were considered, but the additional flexibility of this approach was rejected due to its added complexity of implementation and use.

Some values, such as {PATH\_MAX}, are sometimes so large that they must not be used to, say, allocate arrays. The *sysconf()* function returns a negative value to show that this symbolic constant is not even defined in this case.

Similar to *pathconf()*, this permits the implementation not to have a limit. When one resource is infinite, returning an error indicating that some other resource limit has been reached is conforming behavior.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*confstr()*, *fpathconf()*

XBD <limits.h>, <unistd.h>

XCU *getconf*

## CHANGE HISTORY

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

### Issue 5

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

The \_XBS\_ variables and name values are added to the table of system variables in the DESCRIPTION. These are all marked EX.

### Issue 6

The symbol CLK\_TCK is obsolescent and removed. It is replaced with the phrase “clock ticks per second”.

The symbol {PASS\_MAX} is removed.

The following changes were made to align with the IEEE P1003.1a draft standard:

- Table entries are added for the following variables: \_SC\_REGEX, \_SC\_SHELL, \_SC\_REGEX\_VERSION, \_SC\_SYMLINK\_MAX.

The following *sysconf()* variables and their associated names are added for alignment with IEEE Std 1003.1d-1999:

\_POSIX\_ADVISORY\_INFO  
\_POSIX\_CPUTIME  
\_POSIX\_SPAWN  
\_POSIX\_SPORADIC\_SERVER  
\_POSIX\_THREAD\_CPUTIME  
\_POSIX\_THREAD\_SPORADIC\_SERVER  
\_POSIX\_TIMEOUTS

The following changes are made to the DESCRIPTION for alignment with IEEE Std 1003.1j-2000:

- A statement expressing the dependency of support for some system variables on implementation options is added.
- The following system variables are added:

```
_POSIX_BARRIERS
_POSIX_CLOCK_SELECTION
_POSIX_MONOTONIC_CLOCK
_POSIX_READER_WRITER_LOCKS
_POSIX_SPIN_LOCKS
_POSIX_TYPED_MEMORY_OBJECTS
```

The following system variables are added for alignment with IEEE Std 1003.2d-1994:

```
_POSIX2_PBS
_POSIX2_PBS_ACCOUNTING
_POSIX2_PBS_LOCATE
_POSIX2_PBS_MESSAGE
_POSIX2_PBS_TRACK
```

The following *sysconf()* variables and their associated names are added for alignment with IEEE Std 1003.1q-2000:

```
_POSIX_TRACE
_POSIX_TRACE_EVENT_FILTER
_POSIX_TRACE_INHERIT
_POSIX_TRACE_LOG
```

The macros associated with the *c89* programming models are marked LEGACY, and new equivalent macros associated with *c99* are introduced.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/62 is applied, updating the DESCRIPTION to denote that the *\_PC\** and *\_SC\** symbols are now required to be supported. A corresponding change has been made in the Base Definitions volume of POSIX.1-2008. The deletion in the second paragraph removes some duplicated text. Additional symbols that were erroneously omitted from this reference page have been added.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/63 is applied, making it clear in the RETURN VALUE section that the value returned for *sysconf(\_SC\_OPEN\_MAX)* may change if a call to *setrlimit()* adjusts the RLIMIT\_NOFILE soft limit.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/134 is applied, updating the DESCRIPTION to remove an erroneous entry for *\_POSIX\_SYMLINK\_MAX*. This corrects an error in IEEE Std 1003.1-2001/Cor 1-2002.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/135 is applied, removing *\_POSIX\_FILE\_LOCKING*, *\_POSIX\_MULTI\_PROCESS*, *\_POSIX2\_C\_VERSION*, and *\_XOPEN\_XCU\_VERSION* (and their associated *\_SC\_\** variables) from the DESCRIPTION and APPLICATION USAGE sections.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/136 is applied, adding the following constants (and their associated *\_SC\_\** variables) to the DESCRIPTION:

67220            \_POSIX\_SS\_REPL\_MAX  
67221            \_POSIX\_TRACE\_EVENT\_NAME\_MAX  
67222            \_POSIX\_TRACE\_NAME\_MAX  
67223            \_POSIX\_TRACE\_SYS\_MAX  
67224            \_POSIX\_TRACE\_USER\_EVENT\_MAX

67225            The RETURN VALUE and APPLICATION USAGE sections are updated to note that if variables  
67226            are dependent on unsupported options, the results are unspecified.

67227            IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/137 is applied, removing  
67228            \_REGEX\_VERSION and \_SC\_REGEX\_VERSION.

67229 **Issue 7**

67230            Austin Group Interpretation 1003.1-2001 #160 is applied.

67231            SD5-XSH-ERN-166 is applied, changing “Maximum size” to “Initial size” for the “Maximum  
67232            size of ...” entries in the table in the DESCRIPTION.

67233            The variables for the supported programming environments are updated to be V7 and the  
67234            LEGACY variables are removed.

67235            The following constants are added:

67236            \_POSIX\_THREAD\_ROBUST\_PRIO\_INHERIT  
67237            \_POSIX\_THREAD\_ROBUST\_PRIO\_PROTECT

67238            The \_XOPEN\_UUCP variable and its associated \_SC\_XOPEN\_UUCP value is added to the table  
67239            of system variables.

67240 **NAME**

67241 syslog — log a message

67242 **SYNOPSIS**

```
67243 XSI      #include <syslog.h>
67244          void syslog(int priority, const char *message, ... /* argument */);
```

67245 **DESCRIPTION**67246 Refer to *closelog()*.

67247 **NAME**

67248       system — issue a command

67249 **SYNOPSIS**

67250       #include &lt;stdlib.h&gt;

67251       int system(const char \*command);

67252 **DESCRIPTION**

67253 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 67254 conflict between the requirements described here and the ISO C standard is unintentional. This  
 67255 volume of POSIX.1-2008 defers to the ISO C standard.

67256 If *command* is a null pointer, the *system()* function shall determine whether the host environment  
 67257 has a command processor. If *command* is not a null pointer, the *system()* function shall pass the  
 67258 string pointed to by *command* to that command processor to be executed in an implementation-  
 67259 defined manner; this might then cause the program calling *system()* to behave in a non-  
 67260 conforming manner or to terminate.

67261 CX       The *system()* function shall behave as if a child process were created using *fork()*, and the child  
 67262 process invoked the *sh* utility using *execl()* as follows:

67263       execl(&lt;shell path&gt;, "sh", "-c", command, (char \*)0);

67264 where <shell path> is an unspecified pathname for the *sh* utility. It is unspecified whether the  
 67265 handlers registered with *pthread\_atfork()* are called as part of the creation of the child process.

67266 The *system()* function shall ignore the SIGINT and SIGQUIT signals, and shall block the  
 67267 SIGCHLD signal, while waiting for the command to terminate. If this might cause the  
 67268 application to miss a signal that would have killed it, then the application should examine the  
 67269 return value from *system()* and take whatever action is appropriate to the application if the  
 67270 command terminated due to receipt of a signal.

67271 The *system()* function shall not affect the termination status of any child of the calling processes  
 67272 other than the process or processes it itself creates.

67273       The *system()* function shall not return until the child process has terminated.67274       The *system()* function need not be thread-safe.67275 **RETURN VALUE**

67276 If *command* is a null pointer, *system()* shall return non-zero to indicate that a command processor  
 67277 CX       is available, or zero if none is available. The *system()* function shall always return non-zero  
 67278 when *command* is NULL.

67279 CX       If *command* is not a null pointer, *system()* shall return the termination status of the command  
 67280 language interpreter in the format specified by *waitpid()*. The termination status shall be as  
 67281 defined for the *sh* utility; otherwise, the termination status is unspecified. If some error prevents  
 67282 the command language interpreter from executing after the child process is created, the return  
 67283 value from *system()* shall be as if the command language interpreter had terminated using  
 67284 *exit*(127) or *\_exit*(127). If a child process cannot be created, or if the termination status for the  
 67285 command language interpreter cannot be obtained, *system()* shall return -1 and set *errno* to  
 67286 indicate the error.

67287 **ERRORS**67288 CX       The *system()* function may set *errno* values as described by *fork()*.

67289 In addition, *system()* may fail if:

67290 CX [ECHILD] The status of the child process created by *system()* is no longer available.

## 67291 EXAMPLES

67292 None.

## 67293 APPLICATION USAGE

67294 If the return value of *system()* is not `-1`, its value can be decoded through the use of the macros  
67295 described in `<sys/wait.h>`. For convenience, these macros are also provided in `<stdlib.h>`.

67296 Note that, while *system()* must ignore `SIGINT` and `SIGQUIT` and block `SIGCHLD` while waiting  
67297 for the child to terminate, the handling of signals in the executed command is as specified by  
67298 *fork()* and *exec*. For example, if `SIGINT` is being caught or is set to `SIG_DFL` when *system()* is  
67299 called, then the child is started with `SIGINT` handling set to `SIG_DFL`.

67300 Ignoring `SIGINT` and `SIGQUIT` in the parent process prevents coordination problems (two  
67301 processes reading from the same terminal, for example) when the executed command ignores or  
67302 catches one of the signals. It is also usually the correct action when the user has given a  
67303 command to the application to be executed synchronously (as in the `'!'` command in many  
67304 interactive applications). In either case, the signal should be delivered only to the child process,  
67305 not to the application itself. There is one situation where ignoring the signals might have less  
67306 than the desired effect. This is when the application uses *system()* to perform some task invisible  
67307 to the user. If the user typed the interrupt character ("`^C`", for example) while *system()* is being  
67308 used in this way, one would expect the application to be killed, but only the executed command  
67309 is killed. Applications that use *system()* in this way should carefully check the return status from  
67310 *system()* to see if the executed command was successful, and should take appropriate action  
67311 when the command fails.

67312 Blocking `SIGCHLD` while waiting for the child to terminate prevents the application from  
67313 catching the signal and obtaining status from *system()*'s child process before *system()* can get the  
67314 status itself.

67315 The context in which the utility is ultimately executed may differ from that in which *system()*  
67316 was called. For example, file descriptors that have the `FD_CLOEXEC` flag set are closed, and the  
67317 process ID and parent process ID are different. Also, if the executed utility changes its  
67318 environment variables or its current working directory, that change is not reflected in the caller's  
67319 context.

67320 There is no defined way for an application to find the specific path for the shell. However,  
67321 *confstr()* can provide a value for *PATH* that is guaranteed to find the *sh* utility.

67322 Using the *system()* function in more than one thread in a process or when the `SIGCHLD` signal is  
67323 being manipulated by more than one thread in a process may produce unexpected results.

## 67324 RATIONALE

67325 The *system()* function should not be used by programs that have set user (or group) ID  
67326 privileges. The *fork()* and *exec* family of functions (except *execlp()* and *execvp()*), should be used  
67327 instead. This prevents any unforeseen manipulation of the environment of the user that could  
67328 cause execution of commands not anticipated by the calling program.

67329 There are three levels of specification for the *system()* function. The ISO C standard gives the  
67330 most basic. It requires that the function exists, and defines a way for an application to query  
67331 whether a command language interpreter exists. It says nothing about the command language  
67332 or the environment in which the command is interpreted.

67333 POSIX.1-2008 places additional restrictions on *system()*. It requires that if there is a command  
67334 language interpreter, the environment must be as specified by *fork()* and *exec*. This ensures, for



example, that close-on-exec works, that file locks are not inherited, and that the process ID is different. It also specifies the return value from *system()* when the command line can be run, thus giving the application some information about the command's completion status.

Finally, POSIX.1-2008 requires the command to be interpreted as in the shell command language defined in the Shell and Utilities volume of POSIX.1-2008.

Note that, *system(NULL)* is required to return non-zero, indicating that there is a command language interpreter. At first glance, this would seem to conflict with the ISO C standard which allows *system(NULL)* to return zero. There is no conflict, however. A system must have a command language interpreter, and is non-conforming if none is present. It is therefore permissible for the *system()* function on such a system to implement the behavior specified by the ISO C standard as long as it is understood that the implementation does not conform to POSIX.1-2008 if *system(NULL)* returns zero.

It was explicitly decided that when *command* is NULL, *system()* should not be required to check to make sure that the command language interpreter actually exists with the correct mode, that there are enough processes to execute it, and so on. The call *system(NULL)* could, theoretically, check for such problems as too many existing child processes, and return zero. However, it would be inappropriate to return zero due to such a (presumably) transient condition. If some condition exists that is not under the control of this application and that would cause any *system()* call to fail, that system has been rendered non-conforming.

Early drafts required, or allowed, *system()* to return with *errno* set to [EINTR] if it was interrupted with a signal. This error return was removed, and a requirement that *system()* not return until the child has terminated was added. This means that if a *waitpid()* call in *system()* exits with *errno* set to [EINTR], *system()* must reissue the *waitpid()*. This change was made for two reasons:

1. There is no way for an application to clean up if *system()* returns [EINTR], short of calling *wait()*, and that could have the undesirable effect of returning the status of children other than the one started by *system()*.
2. While it might require a change in some historical implementations, those implementations already have to be changed because they use *wait()* instead of *waitpid()*.

Note that if the application is catching SIGCHLD signals, it will receive such a signal before a successful *system()* call returns.

To conform to POSIX.1-2008, *system()* must use *waitpid()*, or some similar function, instead of *wait()*.

The following code sample illustrates how *system()* might be implemented on an implementation conforming to POSIX.1-2008.

```
#include <signal.h>
int system(const char *cmd)
{
    int stat;
    pid_t pid;
    struct sigaction sa, savintr, savequit;
    sigset_t saveblock;
    if (cmd == NULL)
        return(1);
    sa.sa_handler = SIG_IGN;
    sigemptyset(&sa.sa_mask);
    sa.sa_flags = 0;
```



```

67382     sigemptyset(&saveintr.sa_mask);
67383     sigemptyset(&savequit.sa_mask);
67384     sigaction(SIGINT, &sa, &saveintr);
67385     sigaction(SIGQUIT, &sa, &savequit);
67386     sigaddset(&sa.sa_mask, SIGCHLD);
67387     sigprocmask(SIG_BLOCK, &sa.sa_mask, &saveblock);
67388     if ((pid = fork()) == 0) {
67389         sigaction(SIGINT, &saveintr, (struct sigaction *)0);
67390         sigaction(SIGQUIT, &savequit, (struct sigaction *)0);
67391         sigprocmask(SIG_SETMASK, &saveblock, (sigset_t *)0);
67392         execl("/bin/sh", "sh", "-c", cmd, (char *)0);
67393         _exit(127);
67394     }
67395     if (pid == -1) {
67396         stat = -1; /* errno comes from fork() */
67397     } else {
67398         while (waitpid(pid, &stat, 0) == -1) {
67399             if (errno != EINTR){
67400                 stat = -1;
67401                 break;
67402             }
67403         }
67404     }
67405     sigaction(SIGINT, &saveintr, (struct sigaction *)0);
67406     sigaction(SIGQUIT, &savequit, (struct sigaction *)0);
67407     sigprocmask(SIG_SETMASK, &saveblock, (sigset_t *)0);
67408     return(stat);
67409 }

```

Note that, while a particular implementation of *system()* (such as the one above) can assume a particular path for the shell, such a path is not necessarily valid on another system. The above example is not portable, and is not intended to be.

Note also that the above example implementation is not thread-safe. Implementations can provide a thread-safe *system()* function, but doing so involves complications such as how to restore the signal dispositions for SIGINT and SIGQUIT correctly if there are overlapping calls, and how to deal with cancellation. The example above would not restore the signal dispositions and would leak a process ID if cancelled. This does not matter for a non-thread-safe implementation since canceling a non-thread-safe function results in undefined behavior (see [Section 2.9.5.2](#), on page 518). To avoid leaking a process ID, a thread-safe implementation would need to terminate the child process when acting on a cancellation.

One reviewer suggested that an implementation of *system()* might want to use an environment variable such as *SHELL* to determine which command interpreter to use. The supposed implementation would use the default command interpreter if the one specified by the environment variable was not available. This would allow a user, when using an application that prompts for command lines to be processed using *system()*, to specify a different command interpreter. Such an implementation is discouraged. If the alternate command interpreter did not follow the command line syntax specified in the Shell and Utilities volume of POSIX.1-2008, then changing *SHELL* would render *system()* non-conforming. This would affect applications that expected the specified behavior from *system()*, and since the Shell and Utilities volume of POSIX.1-2008 does not mention that *SHELL* affects *system()*, the application would not know that it needed to unset *SHELL*.

67432 **FUTURE DIRECTIONS**

67433 None.

67434 **SEE ALSO**67435 Section 2.9.5.2 (on page 518), *exec*, *pipe()*, *pthread\_atfork()*, *wait()*67436 XBD *<limits.h>*, *<signal.h>*, *<stdlib.h>*, *<sys/wait.h>*67437 XCU *sh*67438 **CHANGE HISTORY**

67439 First released in Issue 1. Derived from Issue 1 of the SVID.

67440 **Issue 6**

67441 Extensions beyond the ISO C standard are marked.

67442 **Issue 7**67443 Austin Group Interpretation 1003.1-2001 #055 is applied, clarifying the thread-safety of this  
67444 function and treatment of *at\_fork()* handlers.

67445 Austin Group Interpretation 1003.1-2001 #156 is applied.

67446 SD5-XSH-ERN-30 is applied.

67447 **NAME**

67448 tan, tanf, tanl — tangent function

67449 **SYNOPSIS**

```
67450 #include <math.h>
67451 double tan(double x);
67452 float tanf(float x);
67453 long double tanl(long double x);
```

67454 **DESCRIPTION**

67455 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 67456 conflict between the requirements described here and the ISO C standard is unintentional. This  
 67457 volume of POSIX.1-2008 defers to the ISO C standard.

67458 These functions shall compute the tangent of their argument  $x$ , measured in radians.

67459 An application wishing to check for error situations should set *errno* to zero and call  
 67460 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 67461 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 67462 zero, an error has occurred.

67463 **RETURN VALUE**

67464 Upon successful completion, these functions shall return the tangent of  $x$ .

67465 MXX If the correct value would cause underflow, and is not representable, a range error may occur,  
 67466 MXX and *tan()*, *tanf()*, and *tanl()* shall return 0.0, or (if IEC 60559 Floating-Point is not supported) an  
 67467 implementation-defined value no greater in magnitude than DBL\_MIN, FLT\_MIN, and  
 67468 LDBL\_MIN, respectively.

67469 MX If  $x$  is NaN, a NaN shall be returned.

67470 If  $x$  is  $\pm 0$ ,  $x$  shall be returned.

67471 If  $x$  is subnormal, a range error may occur

67472 MXX and  $x$  should be returned.

67473 MX If  $x$  is not returned, *tan()*, *tanf()*, and *tanl()* shall return an implementation-defined value no  
 67474 greater in magnitude than DBL\_MIN, FLT\_MIN, and LDBL\_MIN, respectively.

67475 If  $x$  is  $\pm\text{Inf}$ , a domain error shall occur, and either a NaN (if supported), or an implementation-  
 67476 defined value shall be returned.

67477 MXX If the correct value would cause underflow, and is representable, a range error may occur and  
 67478 the correct value shall be returned.

67479 XSI If the correct value would cause overflow, a range error shall occur and *tan()*, *tanf()*, and *tanl()*  
 67480 shall return  $\pm\text{HUGE\_VAL}$ ,  $\pm\text{HUGE\_VALF}$ , and  $\pm\text{HUGE\_VALL}$ , respectively, with the same sign  
 67481 as the correct value of the function.

67482 **ERRORS**

67483 These functions shall fail if:

67484 MX **Domain Error** The value of  $x$  is  $\pm\text{Inf}$ .

67485 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 67486 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 67487 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 67488 shall be raised.

67489 XSI **Range Error** The result overflows

67490 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,

67491 then *errno* shall be set to [ERANGE]. If the integer expression

67492 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the overflow

67493 floating-point exception shall be raised.

67494 These functions may fail if:

67495 MX **Range Error** The result underflows, or the value of *x* is subnormal.

67496 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,

67497 then *errno* shall be set to [ERANGE]. If the integer expression

67498 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow

67499 floating-point exception shall be raised.

## 67500 EXAMPLES

### 67501 Taking the Tangent of a 45-Degree Angle

```
67502 #include <math.h>
67503 ...
67504 double radians = 45.0 * M_PI / 180;
67505 double result;
67506 ...
67507 result = tan (radians);
```

## 67508 APPLICATION USAGE

67509 There are no known floating-point representations such that for a normal argument, *tan(x)* is

67510 either overflow or underflow.

67511 These functions may lose accuracy when their argument is near a multiple of  $\pi/2$  or is far from

67512 0.0.

67513 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &

67514 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

## 67515 RATIONALE

67516 None.

## 67517 FUTURE DIRECTIONS

67518 None.

## 67519 SEE ALSO

67520 [\*atan\(\)\*](#), [\*feclearexcept\(\)\*](#), [\*fetestexcept\(\)\*](#), [\*isnan\(\)\*](#)

67521 XBD [Section 4.20](#) (on page 117), [\*\*<math.h>\*\*](#)

## 67522 CHANGE HISTORY

67523 First released in Issue 1. Derived from Issue 1 of the SVID.

### 67524 Issue 5

67525 The last two paragraphs of the DESCRIPTION were included as APPLICATION USAGE notes

67526 in previous issues.

### 67527 Issue 6

67528 The *tanf()* and *tanl()* functions are added for alignment with the ISO/IEC 9899:1999 standard.

67529 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are

67530 revised to align with the ISO/IEC 9899:1999 standard.

67531 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
67532 marked.

67533 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/64 is applied, correcting the last  
67534 paragraph in the RETURN VALUE section.

67535 **Issue 7**

67536 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0635 [68], XSH/TC1-2008/0636 [68],  
67537 and XSH/TC1-2008/0637 [68] are applied.

67538 **NAME**

67539 tanh, tanhf, tanhl — hyperbolic tangent functions

67540 **SYNOPSIS**

67541 #include &lt;math.h&gt;

67542 double tanh(double x);

67543 float tanhf(float x);

67544 long double tanhl(long double x);

67545 **DESCRIPTION**

67546 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 67547 conflict between the requirements described here and the ISO C standard is unintentional. This  
 67548 volume of POSIX.1-2008 defers to the ISO C standard.

67549 These functions shall compute the hyperbolic tangent of their argument  $x$ .

67550 An application wishing to check for error situations should set *errno* to zero and call  
 67551 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 67552 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 67553 zero, an error has occurred.

67554 **RETURN VALUE**67555 Upon successful completion, these functions shall return the hyperbolic tangent of  $x$ .67556 MX If  $x$  is NaN, a NaN shall be returned.67557 If  $x$  is  $\pm 0$ ,  $x$  shall be returned.67558 If  $x$  is  $\pm \text{Inf}$ ,  $\pm 1$  shall be returned.67559 If  $x$  is subnormal, a range error may occur67560 MXX and  $x$  should be returned.

67561 MX If  $x$  is not returned, *tanh()*, *tanhf()*, and *tanhl()* shall return an implementation-defined value no  
 67562 greater in magnitude than DBL\_MIN, FLT\_MIN, and LDBL\_MIN, respectively.

67563 **ERRORS**

67564 These functions may fail if:

67565 MX **Range Error** The value of  $x$  is subnormal.

67566 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 67567 then *errno* shall be set to [ERANGE]. If the integer expression  
 67568 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
 67569 floating-point exception shall be raised.

67570 **EXAMPLES**

67571 None.

67572 **APPLICATION USAGE**

67573 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
 67574 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

67575 **RATIONALE**

67576 None.

67577 **FUTURE DIRECTIONS**

67578 None.

67579 **SEE ALSO**67580 *atanh()*, *feclearexcept()*, *fetestexcept()*, *isnan()*, *tan()*67581 XBD Section 4.20 (on page 117), *<math.h>*67582 **CHANGE HISTORY**

67583 First released in Issue 1. Derived from Issue 1 of the SVID.

67584 **Issue 5**67585 The DESCRIPTION is updated to indicate how an application should check for an error. This  
67586 text was previously published in the APPLICATION USAGE section.67587 **Issue 6**67588 The *tanhf()* and *tanhfll()* functions are added for alignment with the ISO/IEC 9899:1999 standard.67589 The DESCRIPTION, RETURN VALUE, ERRORS, and APPLICATION USAGE sections are  
67590 revised to align with the ISO/IEC 9899:1999 standard.67591 IEC 60559:1989 standard floating-point extensions over the ISO/IEC 9899:1999 standard are  
67592 marked.67593 **Issue 7**

67594 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0638 [68] is applied.

67595 **NAME**

67596       tanl — tangent function

67597 **SYNOPSIS**

67598       #include &lt;math.h&gt;

67599       long double tanl(long double x);

67600 **DESCRIPTION**67601       Refer to *tan()*.



67602 **NAME**

67603 tcdrain — wait for transmission of output

67604 **SYNOPSIS**

67605 #include &lt;termios.h&gt;

67606 int tcdrain(int *fildes*);67607 **DESCRIPTION**67608 The *tcdrain()* function shall block until all output written to the object referred to by *fildes* is  
67609 transmitted. The *fildes* argument is an open file descriptor associated with a terminal.67610 Any attempts to use *tcdrain()* from a process which is a member of a background process group  
67611 on a *fildes* associated with its controlling terminal, shall cause the process group to be sent a  
67612 SIGTTOU signal. If the calling thread is blocking SIGTTOU signals or the process is ignoring  
67613 SIGTTOU signals, the process shall be allowed to perform the operation, and no signal is sent.67614 **RETURN VALUE**67615 Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to  
67616 indicate the error.67617 **ERRORS**67618 The *tcdrain()* function shall fail if:67619 [EBADF] The *fildes* argument is not a valid file descriptor.67620 [EINTR] A signal interrupted *tcdrain()*.67621 [EIO] The process group of the writing process is orphaned, the calling thread is not  
67622 blocking SIGTTOU, and the process is not ignoring SIGTTOU.67623 [ENOTTY] The file associated with *fildes* is not a terminal.67624 **EXAMPLES**

67625 None.

67626 **APPLICATION USAGE**

67627 None.

67628 **RATIONALE**

67629 None.

67630 **FUTURE DIRECTIONS**

67631 None.

67632 **SEE ALSO**67633 *tcflush()*

67634 XBD Chapter 11 (on page 199), &lt;termios.h&gt;

67635 **CHANGE HISTORY**

67636 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

67637 **Issue 6**

67638 The following new requirements on POSIX implementations derive from alignment with the  
67639 Single UNIX Specification:

- 67640 • In the DESCRIPTION, the final paragraph is no longer conditional on  
67641 \_POSIX\_JOB\_CONTROL. This is a FIPS requirement.
- 67642 • The [EIO] error is added.

67643 **Issue 7**

67644 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0639 [79], XSH/TC1-2008/0640 [79],  
67645 and XSH/TC1-2008/0641 [79] are applied.

**NAME**

tcflow — suspend or restart the transmission or reception of data

**SYNOPSIS**

```
#include <termios.h>

int tcflow(int fildes, int action);
```

**DESCRIPTION**

The *tcflow()* function shall suspend or restart transmission or reception of data on the object referred to by *fildes*, depending on the value of *action*. The *fildes* argument is an open file descriptor associated with a terminal.

- If *action* is TCOOFF, output shall be suspended.
- If *action* is TCOON, suspended output shall be restarted.
- If *action* is TCIOFF and *fildes* refers to a terminal device, the system shall transmit a STOP character, which is intended to cause the terminal device to stop transmitting data to the system. If *fildes* is associated with a pseudo-terminal, the STOP character need not be transmitted.
- If *action* is TCION and *fildes* refers to a terminal device, the system shall transmit a START character, which is intended to cause the terminal device to start transmitting data to the system. If *fildes* is associated with a pseudo-terminal, the START character need not be transmitted.

The default on the opening of a terminal file is that neither its input nor its output are suspended.

Attempts to use *tcflow()* from a process which is a member of a background process group on a *fildes* associated with its controlling terminal, shall cause the process group to be sent a SIGTTOU signal. If the calling thread is blocking SIGTTOU signals or the process is ignoring SIGTTOU signals, the process shall be allowed to perform the operation, and no signal is sent.

**RETURN VALUE**

Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to indicate the error.

**ERRORS**

The *tcflow()* function shall fail if:

- |          |                                                                                                                                            |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------|
| [EBADF]  | The <i>fildes</i> argument is not a valid file descriptor.                                                                                 |
| [EINVAL] | The <i>action</i> argument is not a supported value.                                                                                       |
| [EIO]    | The process group of the writing process is orphaned, the calling thread is not blocking SIGTTOU, and the process is not ignoring SIGTTOU. |
| [ENOTTY] | The file associated with <i>fildes</i> is not a terminal.                                                                                  |

67681 **EXAMPLES**

67682       None.

67683 **APPLICATION USAGE**

67684       None.

67685 **RATIONALE**

67686       None.

67687 **FUTURE DIRECTIONS**

67688       None.

67689 **SEE ALSO**67690       *tcsendbreak()*67691       XBD [Chapter 11](#) (on page 199), [<termios.h>](#)67692 **CHANGE HISTORY**

67693       First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

67694 **Issue 6**67695       The following new requirements on POSIX implementations derive from alignment with the  
67696       Single UNIX Specification:

- 67697
- The [EIO] error is added.

67698 **Issue 7**67699       SD5-XSH-ERN-190 is applied, clarifying in the DESCRIPTION the transmission of START and  
67700       STOP characters.67701       POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0642 [79], XSH/TC1-2008/0643 [79],  
67702       and XSH/TC1-2008/0644 [79] are applied.

67703 **NAME**

67704 tcflush — flush non-transmitted output data, non-read input data, or both

67705 **SYNOPSIS**

67706 #include &lt;termios.h&gt;

67707 int tcflush(int *fildes*, int *queue\_selector*);67708 **DESCRIPTION**

67709 Upon successful completion, *tcflush()* shall discard data written to the object referred to by *fildes*  
 67710 (an open file descriptor associated with a terminal) but not transmitted, or data received but not  
 67711 read, depending on the value of *queue\_selector*:

- 67712 • If *queue\_selector* is TCIFLUSH, it shall flush data received but not read.
- 67713 • If *queue\_selector* is TCOFLUSH, it shall flush data written but not transmitted.
- 67714 • If *queue\_selector* is TCIOFLUSH, it shall flush both data received but not read and data  
 67715 written but not transmitted.

67716 Attempts to use *tcflush()* from a process which is a member of a background process group on a  
 67717 *fildes* associated with its controlling terminal shall cause the process group to be sent a SIGTTOU  
 67718 signal. If the calling thread is blocking SIGTTOU signals or the process is ignoring SIGTTOU  
 67719 signals, the process shall be allowed to perform the operation, and no signal is sent.

67720 **RETURN VALUE**

67721 Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to  
 67722 indicate the error.

67723 **ERRORS**67724 The *tcflush()* function shall fail if:

- |       |          |                                                                                 |
|-------|----------|---------------------------------------------------------------------------------|
| 67725 | [EBADF]  | The <i>fildes</i> argument is not a valid file descriptor.                      |
| 67726 | [EINVAL] | The <i>queue_selector</i> argument is not a supported value.                    |
| 67727 | [EIO]    | The process group of the writing process is orphaned, the calling thread is not |
| 67728 |          | blocking SIGTTOU, and the process is not ignoring SIGTTOU.                      |
| 67729 | [ENOTTY] | The file associated with <i>fildes</i> is not a terminal.                       |

67730 **EXAMPLES**

67731 None.

67732 **APPLICATION USAGE**

67733 None.

67734 **RATIONALE**

67735 None.

67736 **FUTURE DIRECTIONS**

67737 None.

67738 **SEE ALSO**67739 *tcdrain()*

67740 XBD Chapter 11 (on page 199), &lt;termios.h&gt;

67741 **CHANGE HISTORY**

67742 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

67743 **Issue 6**

67744 The Open Group Corrigendum U035/1 is applied. In the ERRORS and APPLICATION USAGE  
67745 sections, references to *tcflow()* are replaced with *tcflush()*.

67746 The following new requirements on POSIX implementations derive from alignment with the  
67747 Single UNIX Specification:

- 67748 • In the DESCRIPTION, the final paragraph is no longer conditional on  
67749 \_POSIX\_JOB\_CONTROL. This is a FIPS requirement.
- 67750 • The [EIO] error is added.

67751 **Issue 7**

67752 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0645 [79], XSH/TC1-2008/0646 [79],  
67753 and XSH/TC1-2008/0647 [79] are applied.

67754 **NAME**

67755 tcgetattr — get the parameters associated with the terminal

67756 **SYNOPSIS**

67757 #include &lt;termios.h&gt;

67758 int tcgetattr(int *fildev*, struct termios \**termios\_p*);67759 **DESCRIPTION**

67760 The *tcgetattr()* function shall get the parameters associated with the terminal referred to by *fildev*  
 67761 and store them in the **termios** structure referenced by *termios\_p*. The *fildev* argument is an open  
 67762 file descriptor associated with a terminal.

67763 The *termios\_p* argument is a pointer to a **termios** structure.

67764 The *tcgetattr()* operation is allowed from any process.

67765 If the terminal device supports different input and output baud rates, the baud rates stored in  
 67766 the **termios** structure returned by *tcgetattr()* shall reflect the actual baud rates, even if they are  
 67767 equal. If differing baud rates are not supported, the rate returned as the output baud rate shall  
 67768 be the actual baud rate. If the terminal device does not support split baud rates, the input baud  
 67769 rate stored in the **termios** structure shall be the output rate (as one of the symbolic values).

67770 **RETURN VALUE**

67771 Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to  
 67772 indicate the error.

67773 **ERRORS**

67774 The *tcgetattr()* function shall fail if:

67775 [EBADF] The *fildev* argument is not a valid file descriptor.

67776 [ENOTTY] The file associated with *fildev* is not a terminal.

67777 **EXAMPLES**

67778 None.

67779 **APPLICATION USAGE**

67780 None.

67781 **RATIONALE**

67782 Care must be taken when changing the terminal attributes. Applications should always do a  
 67783 *tcgetattr()*, save the **termios** structure values returned, and then do a *tcsetattr()*, changing only  
 67784 the necessary fields. The application should use the values saved from the *tcgetattr()* to reset the  
 67785 terminal state whenever it is done with the terminal. This is necessary because terminal  
 67786 attributes apply to the underlying port and not to each individual open instance; that is, all  
 67787 processes that have used the terminal see the latest attribute changes.

67788 A program that uses these functions should be written to catch all signals and take other  
 67789 appropriate actions to ensure that when the program terminates, whether planned or not, the  
 67790 terminal device's state is restored to its original state.

67791 Existing practice dealing with error returns when only part of a request can be honored is based  
 67792 on calls to the *ioctl()* function. In historical BSD and System V implementations, the  
 67793 corresponding *ioctl()* returns zero if the requested actions were semantically correct, even if  
 67794 some of the requested changes could not be made. Many existing applications assume this  
 67795 behavior and would no longer work correctly if the return value were changed from zero to -1  
 67796 in this case.

67797 Note that either specification has a problem. When zero is returned, it implies everything

67798 succeeded even if some of the changes were not made. When -1 is returned, it implies  
67799 everything failed even though some of the changes were made.

67800 Applications that need all of the requested changes made to work properly should follow  
67801 `tcsetattr()` with a call to `tcgetattr()` and compare the appropriate field values.

#### 67802 **FUTURE DIRECTIONS**

67803 None.

#### 67804 **SEE ALSO**

67805 [`tcsetattr\(\)`](#)

67806 XBD [Chapter 11](#) (on page 199), [`<termios.h>`](#)

#### 67807 **CHANGE HISTORY**

67808 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

#### 67809 **Issue 6**

67810 In the DESCRIPTION, the rate returned as the input baud rate shall be the output rate.  
67811 Previously, the number zero was also allowed but was obsolescent.



67812 **NAME**

67813 tcgetpgrp — get the foreground process group ID

67814 **SYNOPSIS**

67815 #include &lt;unistd.h&gt;

67816 pid\_t tcgetpgrp(int *fildes*);67817 **DESCRIPTION**67818 The *tcgetpgrp()* function shall return the value of the process group ID of the foreground process  
67819 group associated with the terminal.67820 If there is no foreground process group, *tcgetpgrp()* shall return a value greater than 1 that does  
67821 not match the process group ID of any existing process group.67822 The *tcgetpgrp()* function is allowed from a process that is a member of a background process  
67823 group; however, the information may be subsequently changed by a process that is a member of  
67824 a foreground process group.67825 **RETURN VALUE**67826 Upon successful completion, *tcgetpgrp()* shall return the value of the process group ID of the  
67827 foreground process associated with the terminal. Otherwise, -1 shall be returned and *errno* set to  
67828 indicate the error.67829 **ERRORS**67830 The *tcgetpgrp()* function shall fail if:67831 [EBADF] The *fildes* argument is not a valid file descriptor.67832 [ENOTTY] The calling process does not have a controlling terminal, or the file is not the  
67833 controlling terminal.67834 **EXAMPLES**

67835 None.

67836 **APPLICATION USAGE**

67837 None.

67838 **RATIONALE**

67839 None.

67840 **FUTURE DIRECTIONS**

67841 None.

67842 **SEE ALSO**67843 *setsid()*, *setpgid()*, *tcsetpgrp()*

67844 XBD &lt;sys/types.h&gt;, &lt;unistd.h&gt;

67845 **CHANGE HISTORY**

67846 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

67847 **Issue 6**

67848 In the SYNOPSIS, the optional include of the &lt;sys/types.h&gt; header is removed.

67849 The following new requirements on POSIX implementations derive from alignment with the  
67850 Single UNIX Specification:

- 67851 • The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was  
67852 required for conforming implementations of previous POSIX specifications, it was not  
67853 required for UNIX applications.

67854  
67855

- In the DESCRIPTION, text previously conditional on support for \_POSIX\_JOB\_CONTROL is now mandatory. This is a FIPS requirement.

**67856 NAME**

67857 tcgetsid — get the process group ID for the session leader for the controlling terminal

**67858 SYNOPSIS**

67859 #include <termios.h>  
67860 pid\_t tcgetsid(int *fildes*);

**67861 DESCRIPTION**

67862 The *tcgetsid()* function shall obtain the process group ID of the session for which the terminal  
67863 specified by *fildes* is the controlling terminal.

**67864 RETURN VALUE**

67865 Upon successful completion, *tcgetsid()* shall return the process group ID of the session  
67866 associated with the terminal. Otherwise, a value of -1 shall be returned and *errno* set to indicate  
67867 the error.

**67868 ERRORS**

67869 The *tcgetsid()* function shall fail if:

|       |          |                                                                                  |
|-------|----------|----------------------------------------------------------------------------------|
| 67870 | [EBADF]  | The <i>fildes</i> argument is not a valid file descriptor.                       |
| 67871 | [ENOTTY] | The calling process does not have a controlling terminal, or the file is not the |
| 67872 |          | controlling terminal.                                                            |

**67873 EXAMPLES**

67874 None.

**67875 APPLICATION USAGE**

67876 None.

**67877 RATIONALE**

67878 None.

**67879 FUTURE DIRECTIONS**

67880 None.

**67881 SEE ALSO**

67882 XBD [<termios.h>](#)

**67883 CHANGE HISTORY**

67884 First released in Issue 4, Version 2.

**67885 Issue 5**

67886 Moved from X/OPEN UNIX extension to BASE.

67887 The [EACCES] error has been removed from the list of mandatory errors, and the description of  
67888 [ENOTTY] has been reworded.

**67889 Issue 7**

67890 SD5-XSH-ERN-180 is applied, clarifying the RETURN VALUE section.

67891 The *tcgetsid()* function is moved from the XSI option to the Base.

67892 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0648 [421] is applied.

67893 **NAME**

67894 `tcsendbreak` — send a break for a specific duration

67895 **SYNOPSIS**

67896 `#include <termios.h>`  
67897 `int tcsendbreak(int fildes, int duration);`

67898 **DESCRIPTION**

67899 If the terminal is using asynchronous serial data transmission, `tcsendbreak()` shall cause  
67900 transmission of a continuous stream of zero-valued bits for a specific duration. If *duration* is 0, it  
67901 shall cause transmission of zero-valued bits for at least 0.25 seconds, and not more than 0.5  
67902 seconds. If *duration* is not 0, it shall send zero-valued bits for an implementation-defined period  
67903 of time.

67904 The *fildes* argument is an open file descriptor associated with a terminal.

67905 If the terminal is not using asynchronous serial data transmission, it is implementation-defined  
67906 whether `tcsendbreak()` sends data to generate a break condition or returns without taking any  
67907 action.

67908 Attempts to use `tcsendbreak()` from a process which is a member of a background process group  
67909 on a *fildes* associated with its controlling terminal shall cause the process group to be sent a  
67910 SIGTTOU signal. If the calling thread is blocking SIGTTOU signals or the process is ignoring  
67911 SIGTTOU signals, the process shall be allowed to perform the operation, and no signal is sent.

67912 **RETURN VALUE**

67913 Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to  
67914 indicate the error.

67915 **ERRORS**

67916 The `tcsendbreak()` function shall fail if:

- |       |          |                                                                                 |
|-------|----------|---------------------------------------------------------------------------------|
| 67917 | [EBADF]  | The <i>fildes</i> argument is not a valid file descriptor.                      |
| 67918 | [EIO]    | The process group of the writing process is orphaned, the calling thread is not |
| 67919 |          | blocking SIGTTOU, and the process is not ignoring SIGTTOU.                      |
| 67920 | [ENOTTY] | The file associated with <i>fildes</i> is not a terminal.                       |

67921 **EXAMPLES**

67922 None.

67923 **APPLICATION USAGE**

67924 None.

67925 **RATIONALE**

67926 None.

67927 **FUTURE DIRECTIONS**

67928 None.

67929 **SEE ALSO**

67930 XBD [Chapter 11](#) (on page 199), `<termios.h>`

67931 **CHANGE HISTORY**

67932 First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

67933 **Issue 6**

67934 The following new requirements on POSIX implementations derive from alignment with the  
67935 Single UNIX Specification:

- 67936 • In the DESCRIPTION, text previously conditional on \_POSIX\_JOB\_CONTROL is now  
67937 mandated. This is a FIPS requirement.
- 67938 • The [EIO] error is added.

67939 **Issue 7**

67940 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0649 [79], XSH/TC1-2008/0650 [79],  
67941 and XSH/TC1-2008/0651 [79] are applied.

## NAME

tcsetattr — set the parameters associated with the terminal

## SYNOPSIS

```
#include <termios.h>
```

```
int tcsetattr(int fildes, int optional_actions,  
              const struct termios *termios_p);
```

## DESCRIPTION

The *tcsetattr()* function shall set the parameters associated with the terminal referred to by the open file descriptor *fildes* (an open file descriptor associated with a terminal) from the **termios** structure referenced by *termios\_p* as follows:

- If *optional\_actions* is TCSANOW, the change shall occur immediately.
- If *optional\_actions* is TCSADRAIN, the change shall occur after all output written to *fildes* is transmitted. This function should be used when changing parameters that affect output.
- If *optional\_actions* is TCSAFLUSH, the change shall occur after all output written to *fildes* is transmitted, and all input so far received but not read shall be discarded before the change is made.

If the output baud rate stored in the **termios** structure pointed to by *termios\_p* is the zero baud rate, B0, the modem control lines shall no longer be asserted. Normally, this shall disconnect the line.

If the input baud rate stored in the **termios** structure pointed to by *termios\_p* is 0, the input baud rate given to the hardware is the same as the output baud rate stored in the **termios** structure.

The *tcsetattr()* function shall return successfully if it was able to perform any of the requested actions, even if some of the requested actions could not be performed. It shall set all the attributes that the implementation supports as requested and leave all the attributes not supported by the implementation unchanged. If no part of the request can be honored, it shall return -1 and set *errno* to [EINVAL]. If the input and output baud rates differ and are a combination that is not supported, neither baud rate shall be changed. A subsequent call to *tcgetattr()* shall return the actual state of the terminal device (reflecting both the changes made and not made in the previous *tcsetattr()* call). The *tcsetattr()* function shall not change the values found in the **termios** structure under any circumstances.

The effect of *tcsetattr()* is undefined if the value of the **termios** structure pointed to by *termios\_p* was not derived from the result of a call to *tcgetattr()* on *fildes*; an application should modify only fields and flags defined by this volume of POSIX.1-2008 between the call to *tcgetattr()* and *tcsetattr()*, leaving all other fields and flags unmodified.

No actions defined by this volume of POSIX.1-2008, other than a call to *tcsetattr()*, a close of the last file descriptor in the system associated with this terminal device, or an open of the first file descriptor in the system associated with this terminal device (using the O\_TTY\_INIT flag if it is non-zero and the device is not a pseudo-terminal), shall cause any of the terminal attributes defined by this volume of POSIX.1-2008 to change.

If *tcsetattr()* is called from a process which is a member of a background process group on a *fildes* associated with its controlling terminal:

- If the calling thread is blocking SIGTTOU signals or the process is ignoring SIGTTOU signals, the operation completes normally and no signal is sent.

- Otherwise, a SIGTTOU signal shall be sent to the process group.

**RETURN VALUE**

Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to indicate the error.

**ERRORS**

The *tcsetattr()* function shall fail if:

- |       |          |                                                                                                                                                                                   |
|-------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 67991 | [EBADF]  | The <i>fildev</i> argument is not a valid file descriptor.                                                                                                                        |
| 67992 | [EINTR]  | A signal interrupted <i>tcsetattr()</i> .                                                                                                                                         |
| 67993 | [EINVAL] | The <i>optional_actions</i> argument is not a supported value, or an attempt was made to change an attribute represented in the <b>termios</b> structure to an unsupported value. |
| 67996 | [EIO]    | The process group of the writing process is orphaned, the calling thread is not blocking SIGTTOU, and the process is not ignoring SIGTTOU.                                        |
| 67998 | [ENOTTY] | The file associated with <i>fildev</i> is not a terminal.                                                                                                                         |

**EXAMPLES**

None.

**APPLICATION USAGE**

If trying to change baud rates, applications should call *tcsetattr()* then call *tcgetattr()* in order to determine what baud rates were actually selected.

In general, there are two reasons for an application to change the parameters associated with a terminal device:

1. The device already has working parameter settings but the application needs a different behavior, such as non-canonical mode instead of canonical mode. The application sets (or clears) only a few flags or *c\_cc[]* values. Typically, the terminal device in this case is either the controlling terminal for the process or a pseudo-terminal.
2. The device is a modem or similar piece of equipment connected by a serial line, and it was not open before the application opened it. In this case, the application needs to initialize all of the parameter settings “from scratch”. However, since the **termios** structure may include both standard and non-standard parameters, the application cannot just initialize the whole structure in an arbitrary way (e.g., using *memset()*) as this may cause some of the non-standard parameters to be set incorrectly, resulting in non-conforming behavior of the terminal device. Conversely, the application cannot just set the standard parameters, assuming that the non-standard parameters will already have suitable values, as the device might previously have been used with non-conforming parameter settings (and some implementations retain the settings from one use to the next). The solution is to open the terminal device using the *O\_TTY\_INIT* flag to initialize the terminal device to have conforming parameter settings, obtain those settings using *tcgetattr()*, and then set all of the standard parameters to the desired settings.

**RATIONALE**

The *tcsetattr()* function can be interrupted in the following situations:

- It is interrupted while waiting for output to drain.
- It is called from a process in a background process group and SIGTTOU is caught.

See also the RATIONALE section in *tcgetattr()*.

**FUTURE DIRECTIONS**

Using an input baud rate of 0 to set the input rate equal to the output rate may not necessarily be supported in a future version of this volume of POSIX.1-2008.

**SEE ALSO**

*cfgetispeed()*, *tcgetattr()*

XBD [Chapter 11](#) (on page 199), [<termios.h>](#)

**CHANGE HISTORY**

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

**Issue 6**

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, text previously conditional on `_POSIX_JOB_CONTROL` is now mandated. This is a FIPS requirement.
- The [EIO] error is added.

In the DESCRIPTION, the text describing use of *tcsetattr()* from a process which is a member of a background process group is clarified.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #144 is applied.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0652 [79], XSH/TC1-2008/0653 [79], and XSH/TC1-2008/0654 [79] are applied.



68048 **NAME**

68049 tcsetpgrp — set the foreground process group ID

68050 **SYNOPSIS**

68051 #include &lt;unistd.h&gt;

68052 int tcsetpgrp(int *fildes*, pid\_t *pgid\_id*);68053 **DESCRIPTION**

68054 If the process has a controlling terminal, *tcsetpgrp()* shall set the foreground process group ID  
 68055 associated with the terminal to *pgid\_id*. The application shall ensure that the file associated with  
 68056 *fildes* is the controlling terminal of the calling process and the controlling terminal is currently  
 68057 associated with the session of the calling process. The application shall ensure that the value of  
 68058 *pgid\_id* matches a process group ID of a process in the same session as the calling process.

68059 Attempts to use *tcsetpgrp()* from a process which is a member of a background process group on  
 68060 a *fildes* associated with its controlling terminal shall cause the process group to be sent a  
 68061 SIGTTOU signal. If the calling thread is blocking SIGTTOU signals or the process is ignoring  
 68062 SIGTTOU signals, the process shall be allowed to perform the operation, and no signal is sent.

68063 **RETURN VALUE**

68064 Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to  
 68065 indicate the error.

68066 **ERRORS**68067 The *tcsetpgrp()* function shall fail if:68068 [EBADF] The *fildes* argument is not a valid file descriptor.68069 [EINVAL] This implementation does not support the value in the *pgid\_id* argument.

68070 [EIO] The process group of the writing process is orphaned, the calling thread is not  
 68071 blocking SIGTTOU, and the process is not ignoring SIGTTOU.

68072 [ENOTTY] The calling process does not have a controlling terminal, or the file is not the  
 68073 controlling terminal, or the controlling terminal is no longer associated with  
 68074 the session of the calling process.

68075 [EPERM] The value of *pgid\_id* is a value supported by the implementation, but does not  
 68076 match the process group ID of a process in the same session as the calling  
 68077 process.

68078 **EXAMPLES**

68079 None.

68080 **APPLICATION USAGE**

68081 None.

68082 **RATIONALE**

68083 None.

68084 **FUTURE DIRECTIONS**

68085 None.

68086 **SEE ALSO**68087 *tcgetpgrp()*

68088 XBD &lt;sys/types.h&gt;, &lt;unistd.h&gt;

**CHANGE HISTORY**

First released in Issue 3. Included for alignment with the POSIX.1-1988 standard.

**Issue 6**

In the SYNOPSIS, the inclusion of `<sys/types.h>` is no longer required.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.
- In the DESCRIPTION and ERRORS sections, text previously conditional on `_POSIX_JOB_CONTROL` is now mandated. This is a FIPS requirement.

The normative text is updated to avoid use of the term “must” for application requirements.

The Open Group Corrigendum U047/4 is applied.

**Issue 7**

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0655 [79] and XSH/TC1-2008/0656 [79] are applied.

**NAME**

tdelete, tfind, tsearch, twalk — manage a binary search tree

**SYNOPSIS**

```

#include <search.h>

void *tdelete(const void *restrict key, void **restrict rootp,
              int(*compar)(const void *, const void *));
void *tfind(const void *key, void *const *rootp,
            int(*compar)(const void *, const void *));
void *tsearch(const void *key, void **rootp,
              int (*compar)(const void *, const void *));
void twalk(const void *root,
           void (*action)(const void *, VISIT, int));

```

**DESCRIPTION**

The *tdelete()*, *tfind()*, *tsearch()*, and *twalk()* functions manipulate binary search trees. Comparisons are made with a user-supplied routine, the address of which is passed as the *compar* argument. This routine is called with two arguments, which are the pointers to the elements being compared. The application shall ensure that the user-supplied routine returns an integer less than, equal to, or greater than 0, according to whether the first argument is to be considered less than, equal to, or greater than the second argument. The comparison function need not compare every byte, so arbitrary data may be contained in the elements in addition to the values being compared.

The *tsearch()* function shall build and access the tree. The *key* argument is a pointer to an element to be accessed or stored. If there is a node in the tree whose element is equal to the value pointed to by *key*, a pointer to this found node shall be returned. Otherwise, the value pointed to by *key* shall be inserted (that is, a new node is created and the value of *key* is copied to this node), and a pointer to this node returned. Only pointers are copied, so the application shall ensure that the calling routine stores the data. The *rootp* argument points to a variable that points to the root node of the tree. A null pointer value for the variable pointed to by *rootp* denotes an empty tree; in this case, the variable shall be set to point to the node which shall be at the root of the new tree.

Like *tsearch()*, *tfind()* shall search for a node in the tree, returning a pointer to it if found. However, if it is not found, *tfind()* shall return a null pointer. The arguments for *tfind()* are the same as for *tsearch()*.

The *tdelete()* function shall delete a node from a binary search tree. The arguments are the same as for *tsearch()*. The variable pointed to by *rootp* shall be changed if the deleted node was the root of the tree. If the deleted node was the root of the tree and had no children, the variable pointed to by *rootp* shall be set to a null pointer. The *tdelete()* function shall return a pointer to the parent of the deleted node, or an unspecified non-null pointer if the deleted node was the root node, or a null pointer if the node is not found.

If *tsearch()* adds an element to a tree, or *tdelete()* successfully deletes an element from a tree, the concurrent use of that tree in another thread, or use of pointers produced by a previous call to *tfind()* or *tsearch()*, produces undefined results.

The *twalk()* function shall traverse a binary search tree. The *root* argument is a pointer to the root node of the tree to be traversed. (Any node in a tree may be used as the root for a walk below that node.) The argument *action* is the name of a routine to be invoked at each node. This routine is, in turn, called with three arguments. The first argument shall be the address of the node being visited. The structure pointed to by this argument is unspecified and shall not be modified by

the application, but it shall be possible to cast a pointer-to-node into a pointer-to-pointer-to-element to access the element stored in the node. The second argument shall be a value from an enumeration data type:

```
typedef enum { preorder, postorder, endorder, leaf } VISIT;
```

(defined in `<search.h>`), depending on whether this is the first, second, or third time that the node is visited (during a depth-first, left-to-right traversal of the tree), or whether the node is a leaf. The third argument shall be the level of the node in the tree, with the root being level 0.

If the calling function alters the pointer to the root, the result is undefined.

If the functions pointed to by *action* or *compar* (for any of these binary search functions) change the tree, the results are undefined.

These functions are thread-safe only as long as multiple threads do not access the same tree.

## RETURN VALUE

If the node is found, both *tsearch()* and *tfind()* shall return a pointer to it. If not, *tfind()* shall return a null pointer, and *tsearch()* shall return a pointer to the inserted item.

A null pointer shall be returned by *tsearch()* if there is not enough space available to create a new node.

A null pointer shall be returned by *tdelete()*, *tfind()*, and *tsearch()* if *rootp* is a null pointer on entry.

The *tdelete()* function shall return a pointer to the parent of the deleted node, or an unspecified non-null pointer if the deleted node was the root node, or a null pointer if the node is not found.

The *twalk()* function shall not return a value.

## ERRORS

No errors are defined.

## EXAMPLES

The following code reads in strings and stores structures containing a pointer to each string and a count of its length. It then walks the tree, printing out the stored strings and their lengths in alphabetical order.

```
#include <limits.h>
#include <search.h>
#include <stdlib.h>
#include <string.h>
#include <stdio.h>

struct element {          /* Pointers to these are stored in the tree. */
    int    count;
    char   string[];
};

void *root = NULL;        /* This points to the root. */

int main(void)
{
    char   str[_POSIX2_LINE_MAX+1];
    int    length = 0;
    struct element *elementptr;
    void   *node;
    void   print_node(const void *, VISIT, int);
```

```

68196         int      node_compare(const void *, const void *),
68197                 delete_root(const void *, const void *);

68198     while (fgets(str, sizeof(str), stdin)) {
68199         /* Set element. */
68200         length = strlen(str);
68201         if (str[length-1] == '\n')
68202             str[--length] = '\0';
68203         elementptr = malloc(sizeof(struct element) + length + 1);
68204         strcpy(elementptr->string, str);
68205         elementptr->count = 1;
68206         /* Put element into the tree. */
68207         node = tsearch((void *)elementptr, &root, node_compare);
68208         if (node == NULL) {
68209             fprintf(stderr,
68210                 "tsearch: Not enough space available\n");
68211             exit(EXIT_FAILURE);
68212         }
68213         else if (*(struct element **)node != elementptr) {
68214             /* A node containing the element already exists */
68215             (*(struct element **)node)->count++;
68216             free(elementptr);
68217         }
68218     }
68219     twalk(root, print_node);

68220     /* Delete all nodes in the tree */
68221     while (root != NULL) {
68222         elementptr = *(struct element **)root;
68223         printf("deleting node: string = %s, count = %d\n",
68224             elementptr->string,
68225             elementptr->count);
68226         tdelete((void *)elementptr, &root, delete_root);
68227         free(elementptr);
68228     }

68229     return 0;
68230 }

68231 /*
68232  * This routine compares two nodes, based on an
68233  * alphabetical ordering of the string field.
68234  */
68235 int
68236 node_compare(const void *node1, const void *node2)
68237 {
68238     return strcmp(((const struct element *) node1)->string,
68239         ((const struct element *) node2)->string);
68240 }

68241 /*
68242  * This comparison routine can be used with tdelete()
68243  * when explicitly deleting a root node, as no comparison
68244  * is necessary.

```

```

68245     */
68246     int
68247     delete_root(const void *node1, const void *node2)
68248     {
68249         return 0;
68250     }
68251     /*
68252     *   This routine prints out a node, the second time
68253     *   twalk encounters it or if it is a leaf.
68254     */
68255     void
68256     print_node(const void *ptr, VISIT order, int level)
68257     {
68258         const struct element *p = *(const struct element **) ptr;
68259         if (order == postorder || order == leaf) {
68260             (void) printf("string = %s,   count = %d\n",
68261                 p->string, p->count);
68262         }
68263     }

```

#### APPLICATION USAGE

The *root* argument to *twalk()* is one level of indirection less than the *rootp* arguments to *tdelete()* and *tsearch()*.

There are two nomenclatures used to refer to the order in which tree nodes are visited. The *twalk()* function uses **preorder**, **postorder**, and **endorder** to refer respectively to visiting a node before any of its children, after its left child and before its right, and after both its children. The alternative nomenclature uses **preorder**, **inorder**, and **postorder** to refer to the same visits, which could result in some confusion over the meaning of **postorder**.

Since the return value of *tdelete()* is an unspecified non-null pointer in the case that the root of the tree has been deleted, applications should only use the return value of *tdelete()* as indication of success or failure and should not assume it can be dereferenced. Some implementations in this case will return a pointer to the new root of the tree (or to an empty tree if the deleted root node was the only node in the tree); other implementations return arbitrary non-null pointers.

#### RATIONALE

None.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

[\*hcreate\(\)\*](#), [\*lsearch\(\)\*](#)

XBD [\*\*<search.h>\*\*](#)

#### CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

#### Issue 5

The last paragraph of the DESCRIPTION was included as an APPLICATION USAGE note in previous issues.

68289 **Issue 6**

68290 The normative text is updated to avoid use of the term “must” for application requirements.

68291 The **restrict** keyword is added to the *tdelete()* prototype for alignment with the  
68292 ISO/IEC 9899:1999 standard.

68293 **Issue 7**

68294 Austin Group Interpretation 1003.1-2001 #149 is applied, clarifying concurrent use of the tree in  
68295 another thread.

68296 Austin Group Interpretation 1003.1-2001 #151 is applied, clarifying behavior for *tdelete()* when  
68297 the deleted node is the root node.

68298 Austin Group Interpretation 1003.1-2001 #153 is applied.

68299 **NAME**

68300 telldir — current location of a named directory stream

68301 **SYNOPSIS**

```
68302 XSI    #include <dirent.h>
68303        long telldir(DIR *dirp);
```

68304 **DESCRIPTION**

68305 The *telldir()* function shall obtain the current location associated with the directory stream  
68306 specified by *dirp*.

68307 If the most recent operation on the directory stream was a *seekdir()*, the directory position  
68308 returned from the *telldir()* shall be the same as that supplied as a *loc* argument for *seekdir()*.

68309 **RETURN VALUE**

68310 Upon successful completion, *telldir()* shall return the current location of the specified directory  
68311 stream.

68312 **ERRORS**

68313 No errors are defined.

68314 **EXAMPLES**

68315 None.

68316 **APPLICATION USAGE**

68317 None.

68318 **RATIONALE**

68319 None.

68320 **FUTURE DIRECTIONS**

68321 None.

68322 **SEE ALSO**68323 *fdopendir()*, *readdir()*, *seekdir()*

68324 XBD &lt;dirent.h&gt;

68325 **CHANGE HISTORY**

68326 First released in Issue 2.



68327 **NAME**

68328 tempnam — create a name for a temporary file

68329 **SYNOPSIS**

```
68330 OB XSI #include <stdio.h>
68331 char *tempnam(const char *dir, const char *pfx);
```

68332 **DESCRIPTION**68333 The *tempnam()* function shall generate a pathname that may be used for a temporary file.

68334 The *tempnam()* function allows the user to control the choice of a directory. The *dir* argument  
 68335 points to the name of the directory in which the file is to be created. If *dir* is a null pointer or  
 68336 points to a string which is not a name for an appropriate directory, the path prefix defined as  
 68337 P\_tmpdir in the <stdio.h> header shall be used. If that directory is not accessible, an  
 68338 implementation-defined directory may be used.

68339 Many applications prefer their temporary files to have certain initial letter sequences in their  
 68340 names. The *pfx* argument should be used for this. This argument may be a null pointer or point  
 68341 to a string of up to five bytes to be used as the beginning of the filename.

68342 Some implementations of *tempnam()* may use *tmpnam()* internally. On such implementations, if  
 68343 called more than {TMP\_MAX} times in a single process, the behavior is implementation-defined.

68344 **RETURN VALUE**

68345 Upon successful completion, *tempnam()* shall allocate space for a string, put the generated  
 68346 pathname in that space, and return a pointer to it. The pointer shall be suitable for use in a  
 68347 subsequent call to *free()*. Otherwise, it shall return a null pointer and set *errno* to indicate the  
 68348 error.

68349 **ERRORS**68350 The *tempnam()* function shall fail if:

68351 [ENOMEM] Insufficient storage space is available.

68352 **EXAMPLES**68353 **Generating a Pathname**

68354 The following example generates a pathname for a temporary file in directory */tmp*, with the  
 68355 prefix *file*. After the pathname has been created, the call to *free()* deallocates the space used to  
 68356 store the pathname.

```
68357 #include <stdio.h>
68358 #include <stdlib.h>
68359 ...
68360 const char *directory = "/tmp";
68361 const char *fileprefix = "file";
68362 char *file;

68363 file = tempnam(directory, fileprefix);
68364 free(file);
```

68365 **APPLICATION USAGE**

68366 This function only creates pathnames. It is the application's responsibility to create and remove  
 68367 the files. Between the time a pathname is created and the file is opened, it is possible for some  
 68368 other process to create a file with the same name. Applications may find *tmpfile()* more useful.

68369 Applications should use the *tmpfile()*, *mkdtemp()*, or *mkstemp()* functions instead of the

68370 obsolescent *tempnam()* function.

68371 **RATIONALE**

68372 None.

68373 **FUTURE DIRECTIONS**

68374 The *tempnam()* function may be removed in a future version.

68375 **SEE ALSO**

68376 *fopen()*, *free()*, *mkdtemp()*, *open()*, *tmpfile()*, *tmpnam()*, *unlink()*

68377 XBD <stdio.h>

68378 **CHANGE HISTORY**

68379 First released in Issue 1. Derived from Issue 1 of the SVID.

68380 **Issue 5**

68381 The last paragraph of the DESCRIPTION was included as an APPLICATION USAGE note in  
68382 previous issues.

68383 **Issue 7**

68384 The *tempnam()* function is marked obsolescent.

68385 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0657 [291], XSH/TC1-2008/0658 [137],  
68386 and XSH/TC1-2008/0659 [137] are applied.

68387 **NAME**

68388       tfind — search binary search tree

68389 **SYNOPSIS**

```
68390 XSI      #include <search.h>
68391          void *tfind(const void *key, void *const *rootp,
68392                     int (*compar)(const void *, const void *));
```

68393 **DESCRIPTION**68394       Refer to *tdelete()*.

68395 .mc +

## 68396 NAME

68397 tgamma, tgammaf, tgammal — compute gamma() function

## 68398 SYNOPSIS

```
68399 #include <math.h>
68400 double tgamma(double x);
68401 float tgammaf(float x);
68402 long double tgammal(long double x);
```

## 68403 DESCRIPTION

68404 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
68405 conflict between the requirements described here and the ISO C standard is unintentional. This  
68406 volume of POSIX.1-2008 defers to the ISO C standard.

68407 These functions shall compute  $\Gamma(x)$  where  $\Gamma(x)$  is defined as  $\int_0^{\infty} e^{-t} t^{x-1} dt$ .

68408 An application wishing to check for error situations should set *errno* to zero and call  
68409 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
68410 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
68411 zero, an error has occurred.

## 68412 RETURN VALUE

68413 Upon successful completion, these functions shall return the gamma of *x*.

68414 CX If *x* is a negative integer, a domain error may occur and either a NaN (if supported) or an  
68415 MX implementation-defined value shall be returned. On systems that support the IEC 60559  
68416 Floating-Point option, a domain error shall occur and a NaN shall be returned.

68417 If *x* is  $\pm 0$ , *tgamma()*, *tgammaf()*, and *tgammal()* shall return  $\pm \text{HUGE\_VAL}$ ,  $\pm \text{HUGE\_VALF}$ , and  
68418 MX  $\pm \text{HUGE\_VALL}$ , respectively. On systems that support the IEC 60559 Floating-Point option, a  
68419 CX pole error shall occur; otherwise, a pole error may occur.

68420 If the correct value would cause overflow, a range error shall occur and *tgamma()*, *tgammaf()*,  
68421 and *tgammal()* shall return  $\pm \text{HUGE\_VAL}$ ,  $\pm \text{HUGE\_VALF}$ , or  $\pm \text{HUGE\_VALL}$ , respectively, with  
68422 the same sign as the correct value of the function.

68423 MX If the correct value would cause underflow, and is not representable, a range error may occur,  
68424 MX and *tgamma()*, *tgammaf()*, and *tgammal()* shall return 0.0, or (if IEC 60559 Floating-Point is not  
68425 supported) an implementation-defined value no greater in magnitude than DBL\_MIN,  
68426 FLT\_MIN, and LDBL\_MIN, respectively.

68427 MX If the correct value would cause underflow, and is representable, a range error may occur and  
68428 the correct value shall be returned.

68429 If *x* is subnormal and  $1/x$  is representable,  $1/x$  should be returned.

68430 MX If *x* is NaN, a NaN shall be returned.

68431 If *x* is  $+\text{Inf}$ , *x* shall be returned.

68432 If *x* is  $-\text{Inf}$ , a domain error shall occur, and a NaN shall be returned.

## 68433 ERRORS

68434 These functions shall fail if:

68435 MX **Domain Error** The value of *x* is a negative integer, or *x* is  $-\text{Inf}$ .  
68436 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
68437 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*

|       |    |                                                                                               |
|-------|----|-----------------------------------------------------------------------------------------------|
| 68438 |    | & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception shall be raised.     |
| 68439 |    |                                                                                               |
| 68440 | MX | <b>Pole Error</b> The value of $x$ is zero.                                                   |
| 68441 |    | If the integer expression ( <i>math_errhandling</i> & MATH_ERRNO) is non-zero,                |
| 68442 |    | then <i>errno</i> shall be set to [ERANGE]. If the integer expression                         |
| 68443 |    | ( <i>math_errhandling</i> & MATH_ERREXCEPT) is non-zero, then the divide-by-zero              |
| 68444 |    | floating-point exception shall be raised.                                                     |
| 68445 |    | <b>Range Error</b> The value overflows.                                                       |
| 68446 |    | If the integer expression ( <i>math_errhandling</i> & MATH_ERRNO) is non-zero,                |
| 68447 |    | then <i>errno</i> shall be set to [ERANGE]. If the integer expression                         |
| 68448 |    | ( <i>math_errhandling</i> & MATH_ERREXCEPT) is non-zero, then the overflow                    |
| 68449 |    | floating-point exception shall be raised.                                                     |
| 68450 |    | These functions may fail if:                                                                  |
| 68451 |    | <b>Domain Error</b> The value of $x$ is a negative integer.                                   |
| 68452 |    | If the integer expression ( <i>math_errhandling</i> & MATH_ERRNO) is non-zero,                |
| 68453 |    | then <i>errno</i> shall be set to [EDOM]. If the integer expression ( <i>math_errhandling</i> |
| 68454 |    | & MATH_ERREXCEPT) is non-zero, then the invalid floating-point exception                      |
| 68455 |    | shall be raised.                                                                              |
| 68456 |    | <b>Pole Error</b> The value of $x$ is zero.                                                   |
| 68457 |    | If the integer expression ( <i>math_errhandling</i> & MATH_ERRNO) is non-zero,                |
| 68458 |    | then <i>errno</i> shall be set to [ERANGE]. If the integer expression                         |
| 68459 |    | ( <i>math_errhandling</i> & MATH_ERREXCEPT) is non-zero, then the divide-by-zero              |
| 68460 |    | floating-point exception shall be raised.                                                     |
| 68461 |    | <b>Range Error</b> The result underflows.                                                     |
| 68462 |    | If the integer expression ( <i>math_errhandling</i> & MATH_ERRNO) is non-zero,                |
| 68463 |    | then <i>errno</i> shall be set to [ERANGE]. If the integer expression                         |
| 68464 |    | ( <i>math_errhandling</i> & MATH_ERREXCEPT) is non-zero, then the underflow                   |
| 68465 |    | floating-point exception shall be raised.                                                     |

**EXAMPLES**

None.

**APPLICATION USAGE**

On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* & MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

**RATIONALE**

This function is named *tgamma()* in order to avoid conflicts with the historical *gamma()* and *lgamma()* functions.

**FUTURE DIRECTIONS**

It is possible that the error response for a negative integer argument may be changed to a pole error and a return value of  $\pm\text{Inf}$ .

**SEE ALSO**

*feclearexcept()*, *fetestexcept()*, *lgamma()*

XBD Section 4.20 (on page 117), <math.h>

68480 **CHANGE HISTORY**

68481 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

68482 IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/65 is applied, correcting the third  
68483 paragraph in the RETURN VALUE section.

68484 **Issue 7**

68485 ISO/IEC 9899:1999 standard, Technical Corrigendum 2 #52 (SD5-XSH-ERN-85) is applied.

68486 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0660 [68], XSH/TC1-2008/0661 [320],  
68487 and XSH/TC1-2008/0662 [68] are applied. .mc +

68488 **NAME**

68489       time — get time

68490 **SYNOPSIS**

68491       #include &lt;time.h&gt;

68492       time\_t time(time\_t \*tloc);

68493 **DESCRIPTION**

68494 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
 68495       conflict between the requirements described here and the ISO C standard is unintentional. This  
 68496       volume of POSIX.1-2008 defers to the ISO C standard.

68497 CX       The *time()* function shall return the value of time in seconds since the Epoch.

68498       The *tloc* argument points to an area where the return value is also stored. If *tloc* is a null pointer,  
 68499       no value is stored.

68500 **RETURN VALUE**

68501       Upon successful completion, *time()* shall return the value of time. Otherwise, (**time\_t**)−1 shall be  
 68502       returned.

68503 **ERRORS**

68504       The *time()* function may fail if:

68505 CX       [**EOverflow**] The number of seconds since the Epoch will not fit in an object of type **time\_t**.

68506 **EXAMPLES**68507       **Getting the Current Time**

68508       The following example uses the *time()* function to calculate the time elapsed, in seconds, since  
 68509       the Epoch, *localtime()* to convert that value to a broken-down time, and *asctime()* to convert the  
 68510       broken-down time values into a printable string.

```
68511       #include <stdio.h>
68512       #include <time.h>

68513       int main(void)
68514       {
68515       time_t result;

68516       result = time(NULL);
68517       printf("%s%ju secs since the Epoch\n",
68518              asctime(localtime(&result)),
68519              (uintmax_t)result);
68520       return(0);
68521       }
```

68522       This example writes the current time to *stdout* in a form like this:

```
68523       Wed Jun 26 10:32:15 1996
68524       835810335 secs since the Epoch
```

## Timing an Event

The following example gets the current time, prints it out in the user's format, and prints the number of minutes to an event being timed.

```
#include <time.h>
#include <stdio.h>
...
time_t now;
int minutes_to_event;
...
time(&now);
minutes_to_event = ...;
printf("The time is ");
puts(asctime(localtime(&now)));
printf("There are %d minutes to the event.\n",
      minutes_to_event);
...
```

## APPLICATION USAGE

None.

## RATIONALE

The *time()* function returns a value in seconds while *clock\_gettime()* and *gettimeofday()* return a **struct timespec** (seconds and nanoseconds) and **struct timeval** (seconds and microseconds), respectively, and are therefore capable of returning more precise times. The *times()* function is also capable of more precision than *time()* as it returns a value in clock ticks, although it returns the elapsed time since an arbitrary point such as system boot time, not since the epoch.

Implementations in which **time\_t** is a 32-bit signed integer (many historical implementations) fail in the year 2038. POSIX.1-2008 does not address this problem. However, the use of the **time\_t** type is mandated in order to ease the eventual fix.

On some systems the *time()* function is implemented using a system call that does not return an error condition in addition to the return value. On these systems it is impossible to differentiate between valid and invalid return values and hence overflow conditions cannot be reliably detected.

The use of the **<time.h>** header instead of **<sys/types.h>** allows compatibility with the ISO C standard.

Many historical implementations (including Version 7) and the 1984 /usr/group standard use **long** instead of **time\_t**. This volume of POSIX.1-2008 uses the latter type in order to agree with the ISO C standard.

## FUTURE DIRECTIONS

In a future version of this volume of POSIX.1-2008, **time\_t** is likely to be required to be capable of representing times far in the future. Whether this will be mandated as a 64-bit type or a requirement that a specific date in the future be representable (for example, 10000 AD) is not yet determined. Systems purchased after the approval of this volume of POSIX.1-2008 should be evaluated to determine whether their lifetime will extend past 2038.

## SEE ALSO

*asctime()*, *clock()*, *clock\_getres()*, *ctime()*, *difftime()*, *futimens()*, *gettimeofday()*, *gmtime()*, *localtime()*, *mktime()*, *strptime()*, *strptime()*, *times()*, *utime()*

XBD **<time.h>**



68571 **CHANGE HISTORY**

68572 First released in Issue 1. Derived from Issue 1 of the SVID.

68573 **Issue 6**

68574 Extensions beyond the ISO C standard are marked.

68575 The EXAMPLES, RATIONALE, and FUTURE DIRECTIONS sections are added.

68576 **Issue 7**68577 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0663 [106], XSH/TC1-2008/0664 [350],  
68578 XSH/TC1-2008/0665 [106], XSH/TC1-2008/0666 [350], and XSH/TC1-2008/0667 [350] are  
68579 applied.

68580

NAME

68581

timer\_create — create a per-process timer

68582

SYNOPSIS

68583

CX

```
#include <signal.h>
#include <time.h>

int timer_create(clockid_t clockid, struct sigevent *restrict evp,
timer_t *restrict timerid);
```

68584

68585

68586

68587

DESCRIPTION

68588

The *timer\_create()* function shall create a per-process timer using the specified clock, *clock\_id*, as the timing base. The *timer\_create()* function shall return, in the location referenced by *timerid*, a timer ID of type **timer\_t** used to identify the timer in timer requests. This timer ID shall be unique within the calling process until the timer is deleted. The particular clock, *clock\_id*, is defined in **<time.h>**. The timer whose ID is returned shall be in a disarmed state upon return from *timer\_create()*.

68589

68590

68591

68592

68593

68594

The *evp* argument, if non-NULL, points to a **sigevent** structure. This structure, allocated by the application, defines the asynchronous notification to occur as specified in [Section 2.4.1](#) (on page 488) when the timer expires. If the *evp* argument is NULL, the effect is as if the *evp* argument pointed to a **sigevent** structure with the *sigev\_notify* member having the value SIGEV\_SIGNAL, the *sigev\_signo* having a default signal number, and the *sigev\_value* member having the value of the timer ID.

68595

68596

68597

68598

68599

68600

Each implementation shall define a set of clocks that can be used as timing bases for per-process timers. All implementations shall support a *clock\_id* of CLOCK\_REALTIME. If the Monotonic Clock option is supported, implementations shall support a *clock\_id* of CLOCK\_MONOTONIC.

68601

MON

68602

68603

Per-process timers shall not be inherited by a child process across a *fork()* and shall be disarmed and deleted by an *exec*.

68604

68605

CPT

If \_POSIX\_CPUTIME is defined, implementations shall support *clock\_id* values representing the CPU-time clock of the calling process.

68606

68607

TCT

If \_POSIX\_THREAD\_CPUTIME is defined, implementations shall support *clock\_id* values representing the CPU-time clock of the calling thread.

68608

68609

CPT|TCT

It is implementation-defined whether a *timer\_create()* function will succeed if the value defined by *clock\_id* corresponds to the CPU-time clock of a process or thread different from the process or thread invoking the function.

68610

68611

68612

TSA

If *evp*→*sigev\_sigev\_notify* is SIGEV\_THREAD and *sev*→*sigev\_notify\_attributes* is not NULL, if the attribute pointed to by *sev*→*sigev\_notify\_attributes* has a thread stack address specified by a call to *pthread\_attr\_setstack()*, the results are unspecified if the signal is generated more than once.

68613

68614

68615

RETURN VALUE

68616

If the call succeeds, *timer\_create()* shall return zero and update the location referenced by *timerid* to a **timer\_t**, which can be passed to the per-process timer calls. If an error occurs, the function shall return a value of –1 and set *errno* to indicate the error. The value of *timerid* is undefined if an error occurs.

68617

68618

68619

68620

ERRORS

68621

The *timer\_create()* function shall fail if:

68622

|          |                                                                            |
|----------|----------------------------------------------------------------------------|
| [EAGAIN] | The system lacks sufficient signal queuing resources to honor the request. |
|----------|----------------------------------------------------------------------------|

68623 [EAGAIN] The calling process has already created all of the timers it is allowed by this  
 68624 implementation.

68625 [EINVAL] The specified clock ID is not defined.

68626 CPT|TCT [ENOTSUP] The implementation does not support the creation of a timer attached to the  
 68627 CPU-time clock that is specified by *clock\_id* and associated with a process or  
 68628 thread different from the process or thread invoking *timer\_create()*.

## 68629 EXAMPLES

68630 None.

## 68631 APPLICATION USAGE

68632 If a timer is created which has *evp->sigev\_sigev\_notify* set to SIGEV\_THREAD and the attribute  
 68633 pointed to by *evp->sigev\_notify\_attributes* has a thread stack address specified by a call to  
 68634 *pthread\_attr\_setstack()*, the memory dedicated as a thread stack cannot be recovered. The reason  
 68635 for this is that the threads created in response to a timer expiration are created detached, or in an  
 68636 unspecified way if the thread attribute's *detachstate* is PTHREAD\_CREATE\_JOINABLE. In  
 68637 neither case is it valid to call *pthread\_join()*, which makes it impossible to determine the lifetime  
 68638 of the created thread which thus means the stack memory cannot be reused.

## 68639 RATIONALE

### 68640 Periodic Timer Overrun and Resource Allocation

68641 The specified timer facilities may deliver realtime signals (that is, queued signals) on  
 68642 implementations that support this option. Since realtime applications cannot afford to lose  
 68643 notifications of asynchronous events, like timer expirations or asynchronous I/O completions, it  
 68644 must be possible to ensure that sufficient resources exist to deliver the signal when the event  
 68645 occurs. In general, this is not a difficulty because there is a one-to-one correspondence between a  
 68646 request and a subsequent signal generation. If the request cannot allocate the signal delivery  
 68647 resources, it can fail the call with an [EAGAIN] error.

68648 Periodic timers are a special case. A single request can generate an unspecified number of  
 68649 signals. This is not a problem if the requesting process can service the signals as fast as they are  
 68650 generated, thus making the signal delivery resources available for delivery of subsequent  
 68651 periodic timer expiration signals. But, in general, this cannot be assured—processing of periodic  
 68652 timer signals may “overrun”; that is, subsequent periodic timer expirations may occur before the  
 68653 currently pending signal has been delivered.

68654 Also, for signals, according to the POSIX.1-1990 standard, if subsequent occurrences of a  
 68655 pending signal are generated, it is implementation-defined whether a signal is delivered for each  
 68656 occurrence. This is not adequate for some realtime applications. So a mechanism is required to  
 68657 allow applications to detect how many timer expirations were delayed without requiring an  
 68658 indefinite amount of system resources to store the delayed expirations.

68659 The specified facilities provide for an overrun count. The overrun count is defined as the  
 68660 number of extra timer expirations that occurred between the time a timer expiration signal is  
 68661 generated and the time the signal is delivered. The signal-catching function, if it is concerned  
 68662 with overruns, can retrieve this count on entry. With this method, a periodic timer only needs  
 68663 one “signal queuing resource” that can be allocated at the time of the *timer\_create()* function call.

68664 A function is defined to retrieve the overrun count so that an application need not allocate static  
 68665 storage to contain the count, and an implementation need not update this storage  
 68666 asynchronously on timer expirations. But, for some high-frequency periodic applications, the  
 68667 overhead of an additional system call on each timer expiration may be prohibitive. The  
 68668 functions, as defined, permit an implementation to maintain the overrun count in user space,

associated with the *timerid*. The *timer\_getoverrun()* function can then be implemented as a macro that uses the *timerid* argument (which may just be a pointer to a user space structure containing the counter) to locate the overrun count with no system call overhead. Other implementations, less concerned with this class of applications, can avoid the asynchronous update of user space by maintaining the count in a system structure at the cost of the extra system call to obtain it.

#### Timer Expiration Signal Parameters

The Realtime Signals Extension option supports an application-specific datum that is delivered to the extended signal handler. This value is explicitly specified by the application, along with the signal number to be delivered, in a **sigevent** structure. The type of the application-defined value can be either an integer constant or a pointer. This explicit specification of the value, as opposed to always sending the timer ID, was selected based on existing practice.

It is common practice for realtime applications (on non-POSIX systems or realtime extended POSIX systems) to use the parameters of event handlers as the case label of a switch statement or as a pointer to an application-defined data structure. Since *timer\_ids* are dynamically allocated by the *timer\_create()* function, they can be used for neither of these functions without additional application overhead in the signal handler; for example, to search an array of saved timer IDs to associate the ID with a constant or application data structure.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*clock\_getres()*, *timer\_delete()*, *timer\_getoverrun()*

XBD **<signal.h>**, **<time.h>**

#### CHANGE HISTORY

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

#### Issue 6

The *timer\_create()* function is marked as part of the Timers option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Timers option.

CPU-time clocks are added for alignment with IEEE Std 1003.1d-1999.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by adding the requirement for the CLOCK\_MONOTONIC clock under the Monotonic Clock option.

The **restrict** keyword is added to the *timer\_create()* prototype for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/138 is applied, updating the DESCRIPTION and APPLICATION USAGE sections to describe the case when a timer is created with the notification method set to SIGEV\_THREAD.

#### Issue 7

The *timer\_create()* function is moved from the Timers option to the Base.

**NAME**

timer\_delete — delete a per-process timer

**SYNOPSIS**

```
CX      #include <time.h>
68711   int timer_delete(timer_t timerid);
```

**DESCRIPTION**

The *timer\_delete()* function deletes the specified timer, *timerid*, previously created by the *timer\_create()* function. If the timer is armed when *timer\_delete()* is called, the behavior shall be as if the timer is automatically disarmed before removal. The disposition of pending signals for the deleted timer is unspecified.

The behavior is undefined if the value specified by the *timerid* argument to *timer\_delete()* does not correspond to a timer ID returned by *timer\_create()* but not yet deleted by *timer\_delete()*.

**RETURN VALUE**

If successful, the *timer\_delete()* function shall return a value of zero. Otherwise, the function shall return a value of -1 and set *errno* to indicate the error.

**ERRORS**

No errors are defined.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

If an implementation detects that the value specified by the *timerid* argument to *timer\_delete()* does not correspond to a timer ID returned by *timer\_create()* but not yet deleted by *timer\_delete()*, it is recommended that the function should fail and report an [EINVAL] error.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*timer\_create()*  
XBD <time.h>

**CHANGE HISTORY**

First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

**Issue 6**

The *timer\_delete()* function is marked as part of the Timers option.

The [ENOSYS] error condition has been removed as stubs need not be provided if an implementation does not support the Timers option.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/139 is applied, updating the ERRORS section so that the [EINVAL] error becomes optional.

**Issue 7**

The *timer\_delete()* function is moved from the Timers option to the Base.

**NAME**

timer\_getoverrun, timer\_gettime, timer\_settime — per-process timers

**SYNOPSIS**

```
CX      #include <time.h>

68751      int timer_getoverrun(timer_t timerid);
68752      int timer_gettime(timer_t timerid, struct itimerspec *value);
68753      int timer_settime(timer_t timerid, int flags,
68754          const struct itimerspec *restrict value,
68755          struct itimerspec *restrict ovalue);
```

**DESCRIPTION**

The *timer\_gettime()* function shall store the amount of time until the specified timer, *timerid*, expires and the reload value of the timer into the space pointed to by the *value* argument. The *it\_value* member of this structure shall contain the amount of time before the timer expires, or zero if the timer is disarmed. This value is returned as the interval until timer expiration, even if the timer was armed with absolute time. The *it\_interval* member of *value* shall contain the reload value last set by *timer\_settime()*.

The *timer\_settime()* function shall set the time until the next expiration of the timer specified by *timerid* from the *it\_value* member of the *value* argument and arm the timer if the *it\_value* member of *value* is non-zero. If the specified timer was already armed when *timer\_settime()* is called, this call shall reset the time until next expiration to the *value* specified. If the *it\_value* member of *value* is zero, the timer shall be disarmed. The effect of disarming or resetting a timer with pending expiration notifications is unspecified.

If the flag *TIMER\_ABSTIME* is not set in the argument *flags*, *timer\_settime()* shall behave as if the time until next expiration is set to be equal to the interval specified by the *it\_value* member of *value*. That is, the timer shall expire in *it\_value* nanoseconds from when the call is made. If the flag *TIMER\_ABSTIME* is set in the argument *flags*, *timer\_settime()* shall behave as if the time until next expiration is set to be equal to the difference between the absolute time specified by the *it\_value* member of *value* and the current value of the clock associated with *timerid*. That is, the timer shall expire when the clock reaches the value specified by the *it\_value* member of *value*. If the specified time has already passed, the function shall succeed and the expiration notification shall be made.

The reload value of the timer shall be set to the value specified by the *it\_interval* member of *value*. When a timer is armed with a non-zero *it\_interval*, a periodic (or repetitive) timer is specified.

Time values that are between two consecutive non-negative integer multiples of the resolution of the specified timer shall be rounded up to the larger multiple of the resolution. Quantization error shall not cause the timer to expire earlier than the rounded time value.

If the argument *ovalue* is not NULL, the *timer\_settime()* function shall store, in the location referenced by *ovalue*, a value representing the previous amount of time before the timer would have expired, or zero if the timer was disarmed, together with the previous timer reload value. Timers shall not expire before their scheduled time.

Only a single signal shall be queued to the process for a given timer at any point in time. When a timer for which a signal is still pending expires, no signal shall be queued, and a timer overrun shall occur. When a timer expiration signal is delivered to or accepted by a process, the *timer\_getoverrun()* function shall return the timer expiration overrun count for the specified timer. The overrun count returned contains the number of extra timer expirations that occurred between the time the signal was generated (queued) and when it was delivered or accepted, up



to but not including an implementation-defined maximum of {DELAYTIMER\_MAX}. If the number of such extra expirations is greater than or equal to {DELAYTIMER\_MAX}, then the overrun count shall be set to {DELAYTIMER\_MAX}. The value returned by *timer\_getoverrun()* shall apply to the most recent expiration signal delivery or acceptance for the timer. If no expiration signal has been delivered for the timer, the return value of *timer\_getoverrun()* is unspecified.

The behavior is undefined if the value specified by the *timerid* argument to *timer\_getoverrun()*, *timer\_gettime()*, or *timer\_settime()* does not correspond to a timer ID returned by *timer\_create()* but not yet deleted by *timer\_delete()*.

## RETURN VALUE

If the *timer\_getoverrun()* function succeeds, it shall return the timer expiration overrun count as explained above.

If the *timer\_gettime()* or *timer\_settime()* functions succeed, a value of 0 shall be returned.

If an error occurs for any of these functions, the value -1 shall be returned, and *errno* set to indicate the error.

## ERRORS

The *timer\_settime()* function shall fail if:

[EINVAL] A *value* structure specified a nanosecond value less than zero or greater than or equal to 1 000 million, and the *it\_value* member of that structure did not specify zero seconds and nanoseconds.

The *timer\_settime()* function may fail if:

[EINVAL] The *it\_interval* member of *value* is not zero and the timer was created with notification by creation of a new thread (*sigev\_sigev\_notify* was SIGEV\_THREAD) and a fixed stack address has been set in the thread attribute pointed to by *sigev\_notify\_attributes*.

## EXAMPLES

None.

## APPLICATION USAGE

Using fixed stack addresses is problematic when timer expiration is signaled by the creation of a new thread. Since it cannot be assumed that the thread created for one expiration is finished before the next expiration of the timer, it could happen that two threads use the same memory as a stack at the same time. This is invalid and produces undefined results.

## RATIONALE

Practical clocks tick at a finite rate, with rates of 100 hertz and 1 000 hertz being common. The inverse of this tick rate is the clock resolution, also called the clock granularity, which in either case is expressed as a time duration, being 10 milliseconds and 1 millisecond respectively for these common rates. The granularity of practical clocks implies that if one reads a given clock twice in rapid succession, one may get the same time value twice; and that timers must wait for the next clock tick after the theoretical expiration time, to ensure that a timer never returns too soon. Note also that the granularity of the clock may be significantly coarser than the resolution of the data format used to set and get time and interval values. Also note that some implementations may choose to adjust time and/or interval values to exactly match the ticks of the underlying clock.

This volume of POSIX.1-2008 defines functions that allow an application to determine the implementation-supported resolution for the clocks and requires an implementation to document the resolution supported for timers and *nanosleep()* if they differ from the supported

68840 clock resolution. This is more of a procurement issue than a runtime application issue.

68841 If an implementation detects that the value specified by the *timerid* argument to +  
68842 *timer\_getoverrun()*, *timer\_gettime()*, or *timer\_settime()* does not correspond to a timer ID returned +  
68843 by *timer\_create()* but not yet deleted by *timer\_delete()*, it is recommended that the function +  
68844 should fail and report an [EINVAL] error.

#### 68845 FUTURE DIRECTIONS

68846 None.

#### 68847 SEE ALSO

68848 *clock\_getres()*, *timer\_create()*

68849 XBD <time.h>

#### 68850 CHANGE HISTORY

68851 First released in Issue 5. Included for alignment with the POSIX Realtime Extension.

#### 68852 Issue 6

68853 The *timer\_getoverrun()*, *timer\_gettime()*, and *timer\_settime()* functions are marked as part of the  
68854 Timers option.

68855 The [ENOSYS] error condition has been removed as stubs need not be provided if an  
68856 implementation does not support the Timers option.

68857 The [EINVAL] error condition is updated to include the following: “and the *it\_value* member of  
68858 that structure did not specify zero seconds and nanoseconds.” This change is for IEEE PASC  
68859 Interpretation 1003.1 #89.

68860 The DESCRIPTION for *timer\_getoverrun()* is updated to clarify that “If no expiration signal has  
68861 been delivered for the timer, or if the Realtime Signals Extension is not supported, the return  
68862 value of *timer\_getoverrun()* is unspecified”.

68863 The **restrict** keyword is added to the *timer\_settime()* prototype for alignment with the  
68864 ISO/IEC 9899:1999 standard.

68865 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/140 is applied, updating the ERRORS  
68866 section so that the mandatory [EINVAL] error (“The *timerid* argument does not correspond to an  
68867 ID returned by *timer\_create()* but not yet deleted by *timer\_delete()*”) becomes optional.

68868 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/141 is applied, updating the ERRORS  
68869 section to include an optional [EINVAL] error for the case when a timer is created with the  
68870 notification method set to SIGEV\_THREAD. APPLICATION USAGE text is also added.

#### 68871 Issue 7

68872 The *timer\_getoverrun()*, *timer\_gettime()*, and *timer\_settime()* functions are moved from the Timers  
68873 option to the Base.

68874 Functionality relating to the Realtime Signals Extension option is moved to the Base.



**NAME**

times — get process and waited-for child process times

**SYNOPSIS**

```
#include <sys/times.h>

clock_t times(struct tms *buffer);
```

**DESCRIPTION**

The *times()* function shall fill the **tms** structure pointed to by *buffer* with time-accounting information. The **tms** structure is defined in **<sys/times.h>**.

All times are measured in terms of the number of clock ticks used.

The times of a terminated child process shall be included in the *tms\_cutime* and *tms\_cstime* elements of the parent when *wait()*, *waitid()*, or *waitpid()* returns the process ID of this terminated child. If a child process has not waited for its children, their times shall not be included in its times.

- The *tms\_utime* structure member is the CPU time charged for the execution of user instructions of the calling process.
- The *tms\_stime* structure member is the CPU time charged for execution by the system on behalf of the calling process.
- The *tms\_cutime* structure member is the sum of the *tms\_utime* and *tms\_cutime* times of the child processes.
- The *tms\_cstime* structure member is the sum of the *tms\_stime* and *tms\_cstime* times of the child processes.

**RETURN VALUE**

Upon successful completion, *times()* shall return the elapsed real time, in clock ticks, since an arbitrary point in the past (for example, system start-up time). This point does not change from one invocation of *times()* within the process to another. The return value may overflow the possible range of type **clock\_t**. If *times()* fails, **(clock\_t)-1** shall be returned and *errno* set to indicate the error.

**ERRORS**

The *times()* function shall fail if:

[EOVERFLOW] The return value would overflow the range of **clock\_t**.

**EXAMPLES****Timing a Database Lookup**

The following example defines two functions, *start\_clock()* and *end\_clock()*, that are used to time a lookup. It also defines variables of type **clock\_t** and **tms** to measure the duration of transactions. The *start\_clock()* function saves the beginning times given by the *times()* function. The *end\_clock()* function gets the ending times and prints the difference between the two times.

```
#include <sys/times.h>
#include <stdio.h>
...
void start_clock(void);
void end_clock(char *msg);
...
static clock_t st_time;
static clock_t en_time;
```

```

68919     static struct tms st_cpu;
68920     static struct tms en_cpu;
68921     ...
68922     void
68923     start_clock()
68924     {
68925         st_time = times(&st_cpu);
68926     }
68927
68928     /* This example assumes that the result of each subtraction
68929        is within the range of values that can be represented in
68930        an integer type. */
68931     void
68932     end_clock(char *msg)
68933     {
68934         en_time = times(&en_cpu);
68935
68936         fputs(msg, stdout);
68937         printf("Real Time: %jd, User Time %jd, System Time %jd\n",
68938             (intmax_t)(en_time - st_time),
68939             (intmax_t)(en_cpu.tms_utime - st_cpu.tms_utime),
68940             (intmax_t)(en_cpu.tms_stime - st_cpu.tms_stime));
68941     }

```

## APPLICATION USAGE

Applications should use `sysconf(_SC_CLK_TCK)` to determine the number of clock ticks per second as it may vary from system to system.

## RATIONALE

The accuracy of the times reported is intentionally left unspecified to allow implementations flexibility in design, from uniprocessor to multi-processor networks.

The inclusion of times of child processes is recursive, so that a parent process may collect the total times of all of its descendants. But the times of a child are only added to those of its parent when its parent successfully waits on the child. Thus, it is not guaranteed that a parent process can always see the total times of all its descendants; see also the discussion of the term “realtime” in [alarm\(\)](#).

If the type `clock_t` is defined to be a signed 32-bit integer, it overflows in somewhat more than a year if there are 60 clock ticks per second, or less than a year if there are 100. There are individual systems that run continuously for longer than that. This volume of POSIX.1-2008 permits an implementation to make the reference point for the returned value be the start-up time of the process, rather than system start-up time.

The term “charge” in this context has nothing to do with billing for services. The operating system accounts for time used in this way. That information must be correct, regardless of how that information is used.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[alarm\(\)](#), [exec](#), [fork\(\)](#), [sysconf\(\)](#), [time\(\)](#), [wait\(\)](#), [waitid\(\)](#)

XBD [<sys/times.h>](#)

68964

68965

CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

68966 **NAME**

68967       timezone — difference from UTC and local standard time

68968 **SYNOPSIS**

```
68969 XSI      #include <time.h>
68970          extern long timezone;
```

68971 **DESCRIPTION**68972       Refer to *tzset()*.

68973 **NAME**

68974 tmpfile — create a temporary file

68975 **SYNOPSIS**

68976 #include &lt;stdio.h&gt;

68977 FILE \*tmpfile(void);

68978 **DESCRIPTION**

68979 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 68980 conflict between the requirements described here and the ISO C standard is unintentional. This  
 68981 volume of POSIX.1-2008 defers to the ISO C standard.

68982 The *tmpfile()* function shall create a temporary file and open a corresponding stream. The file  
 68983 shall be automatically deleted when all references to the file are closed. The file shall be opened  
 68984 as in *fopen()* for update (*wb+*), except that implementations may restrict the permissions, either  
 68985 by clearing the file mode bits or setting them to the value *S\_IRUSR* | *S\_IWUSR*.

68986 CX In some implementations, a permanent file may be left behind if the process calling *tmpfile()* is  
 68987 killed while it is processing a call to *tmpfile()*.

68988 An error message may be written to standard error if the stream cannot be opened.

68989 **RETURN VALUE**

68990 Upon successful completion, *tmpfile()* shall return a pointer to the stream of the file that is  
 68991 CX created. Otherwise, it shall return a null pointer and set *errno* to indicate the error.

68992 **ERRORS**68993 The *tmpfile()* function shall fail if:

68994 CX [EINTR] A signal was caught during *tmpfile()*.

68995 CX [EMFILE] All file descriptors available to the process are currently open.

68996 CX [EMFILE] {STREAM\_MAX} streams are currently open in the calling process.

68997 CX [ENFILE] The maximum allowable number of files is currently open in the system.

68998 CX [ENOSPC] The directory or file system which would contain the new file cannot be  
 68999 expanded.

69000 CX [EOVERFLOW] The file is a regular file and the size of the file cannot be represented correctly  
 69001 in an object of type *off\_t*.

69002 The *tmpfile()* function may fail if:

69003 CX [EMFILE] {FOPEN\_MAX} streams are currently open in the calling process.

69004 CX [ENOMEM] Insufficient storage space is available.

69005 **EXAMPLES**69006 **Creating a Temporary File**

69007 The following example creates a temporary file for update, and returns a pointer to a stream for  
 69008 the created file in the *fp* variable.

69009 #include &lt;stdio.h&gt;

69010 ...

69011 FILE \*fp;

69012 fp = tmpfile ();

**APPLICATION USAGE**

It should be possible to open at least {TMP\_MAX} temporary files during the lifetime of the program (this limit may be shared with *tmpnam()*) and there should be no limit on the number simultaneously open other than this limit and any limit on the number of open file descriptors or streams ({OPEN\_MAX}, {FOPEN\_MAX}, {STREAM\_MAX}).

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

Section 2.5 (on page 495), *fopen()*, *mkdtemp()*, *tmpnam()*, *unlink()*

XBD <stdio.h>

**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 5**

Large File Summit extensions are added.

The last two paragraphs of the DESCRIPTION were included as APPLICATION USAGE notes in previous issues.

**Issue 6**

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the ERRORS section, the [EOVERFLOW] condition is added. This change is to support large files.
- The [EMFILE] optional error condition is added.

The APPLICATION USAGE section is added for alignment with the ISO/IEC 9899:1999 standard.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #025 is applied, clarifying that implementations may restrict the permissions of the file created.

SD5-XBD-ERN-4 is applied, changing the definition of the [EMFILE] error.

SD5-XSH-ERN-149 is applied, adding the mandatory [EMFILE] error condition for {STREAM\_MAX} streams open.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0668 [14] is applied.

69047 **NAME**

69048 tmpnam — create a name for a temporary file

69049 **SYNOPSIS**

```
69050 OB      #include <stdio.h>
69051      char *tmpnam(char *s);
```

69052 **DESCRIPTION**

69053 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 69054 conflict between the requirements described here and the ISO C standard is unintentional. This  
 69055 volume of POSIX.1-2008 defers to the ISO C standard.

69056 The *tmpnam()* function shall generate a string that is a valid pathname that does not name an  
 69057 existing file. The function is potentially capable of generating {TMP\_MAX} different strings, but  
 69058 any or all of them may already be in use by existing files and thus not be suitable return values.

69059 The *tmpnam()* function generates a different string each time it is called from the same process,  
 69060 up to {TMP\_MAX} times. If it is called more than {TMP\_MAX} times, the behavior is  
 69061 implementation-defined.

69062 The implementation shall behave as if no function defined in this volume of POSIX.1-2008,  
 69063 except *tmpnam()*, calls *tmpnam()*.

69064 CX The *tmpnam()* function need not be thread-safe if called with a NULL parameter.

69065 **RETURN VALUE**

69066 Upon successful completion, *tmpnam()* shall return a pointer to a string. If no suitable string can  
 69067 be generated, the *tmpnam()* function shall return a null pointer.

69068 If the argument *s* is a null pointer, *tmpnam()* shall leave its result in an internal static object and  
 69069 return a pointer to that object. Subsequent calls to *tmpnam()* may modify the same object. If the  
 69070 argument *s* is not a null pointer, it is presumed to point to an array of at least L\_tmpnam **chars**;  
 69071 *tmpnam()* shall write its result in that array and shall return the argument as its value.

69072 **ERRORS**

69073 No errors are defined.

69074 **EXAMPLES**69075 **Generating a Pathname**

69076 The following example generates a unique pathname and stores it in the array pointed to by *ptr*.

```
69077 #include <stdio.h>
69078 ...
69079 char pathname[L_tmpnam+1];
69080 char *ptr;
69081 ptr = tmpnam(pathname);
```

69082 **APPLICATION USAGE**

69083 This function only creates pathnames. It is the application's responsibility to create and remove  
 69084 the files.

69085 Between the time a pathname is created and the file is opened, it is possible for some other  
 69086 process to create a file with the same name. Applications may find *tmpfile()* more useful.

69087 Applications should use the *tmpfile()*, *mkstemp()*, or *mkdtemp()* functions instead of the  
 69088 obsolescent *tmpnam()* function.

69089 **RATIONALE**

69090 None.

69091 **FUTURE DIRECTIONS**69092 The *tmpnam()* function may be removed in a future version.69093 **SEE ALSO**69094 *fopen()*, *open()*, *mkdtemp()*, *tmpnam()*, *tmpfile()*, *unlink()*

69095 XBD &lt;stdio.h&gt;

69096 **CHANGE HISTORY**

69097 First released in Issue 1. Derived from Issue 1 of the SVID.

69098 **Issue 5**

69099 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

69100 **Issue 6**

69101 Extensions beyond the ISO C standard are marked.

69102 The normative text is updated to avoid use of the term “must” for application requirements.

69103 The DESCRIPTION is expanded for alignment with the ISO/IEC 9899:1999 standard.

69104 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/142 is applied, updating the  
69105 DESCRIPTION to allow implementations of the *tmpnam()* function to call *tmpnam()*.69106 **Issue 7**69107 Austin Group Interpretation 1003.1-2001 #148 is applied, clarifying that the *tmpnam()* function  
69108 need not be thread-safe if called with a NULL parameter.69109 The *tmpnam()* function is marked obsolescent.69110 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0669 [291] and XSH/TC1-2008/0670  
69111 [291,429] are applied.



**69112 NAME**

69113       toascii — translate an integer to a 7-bit ASCII character

**69114 SYNOPSIS**

```
69115 OB XSI #include <ctype.h>  
69116       int toascii(int c);
```

**69117 DESCRIPTION**

69118       The *toascii()* function shall convert its argument into a 7-bit ASCII character.

**69119 RETURN VALUE**

69120       The *toascii()* function shall return the value (*c* &0x7f).

**69121 ERRORS**

69122       No errors are returned.

**69123 EXAMPLES**

69124       None.

**69125 APPLICATION USAGE**

69126       The *toascii()* function cannot be used portably in a localized application.

**69127 RATIONALE**

69128       None.

**69129 FUTURE DIRECTIONS**

69130       The *toascii()* function may be removed in a future version.

**69131 SEE ALSO**

69132       *isascii()*

69133       XBD <ctype.h>

**69134 CHANGE HISTORY**

69135       First released in Issue 1. Derived from Issue 1 of the SVID.

**69136 Issue 7**

69137       The *toascii()* function is marked obsolescent.

69138 **NAME**

69139       tolower, tolower\_l — transliterate uppercase characters to lowercase

69140 **SYNOPSIS**

```
69141       #include <ctype.h>
69142       int tolower(int c);
69143 CX     int tolower_l(int c, locale_t locale);
```

69144 **DESCRIPTION**

69145 CX     For *tolower()*: The functionality described on this reference page is aligned with the ISO C  
69146 standard. Any conflict between the requirements described here and the ISO C standard is  
69147 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

69148 CX     The *tolower()* and *tolower\_l()* functions have as a domain a type **int**, the value of which is  
69149 representable as an **unsigned char** or the value of EOF. If the argument has any other value, the  
69150 CX     behavior is undefined. If the argument of *tolower()* or *tolower\_l()* represents an uppercase letter,  
69151 and there exists a corresponding lowercase letter as defined by character type information in the  
69152 CX     current locale or in the locale represented by *locale*, respectively (category *LC\_CTYPE*), the  
69153 result shall be the corresponding lowercase letter. All other arguments in the domain are  
69154 returned unchanged.

69155 CX     The behavior is undefined if the *locale* argument to *tolower\_l()* is the special locale object  
69156 *LC\_GLOBAL\_LOCALE* or is not a valid locale object handle.

69157 **RETURN VALUE**

69158 CX     Upon successful completion, the *tolower()* and *tolower\_l()* functions shall return the lowercase  
69159 letter corresponding to the argument passed; otherwise, they shall return the argument  
69160 unchanged.

69161 **ERRORS**

69162       No errors are defined.

69163 **EXAMPLES**

69164       None.

69165 **APPLICATION USAGE**

69166       None.

69167 **RATIONALE**

69168       None.

69169 **FUTURE DIRECTIONS**

69170       None.

69171 **SEE ALSO**

69172       [\*setlocale\(\)\*](#), [\*uselocale\(\)\*](#)  
69173       XBD [Chapter 7](#) (on page 135), [\*<ctype.h>\*](#), [\*<locale.h>\*](#)

69174 **CHANGE HISTORY**

69175       First released in Issue 1. Derived from Issue 1 of the SVID.

69176 **Issue 6**

69177       Extensions beyond the ISO C standard are marked.

69178 **Issue 7**

69179 The *tolower\_l()* function is added from The Open Group Technical Standard, 2006, Extended API  
69180 Set Part 4.

69181 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0671 [283] and XSH/TC1-2008/0672  
69182 [283] are applied.

69183 **NAME**

69184 toupper, toupper\_l — transliterate lowercase characters to uppercase

69185 **SYNOPSIS**

```
69186 #include <ctype.h>
69187 int toupper(int c);
69188 CX int toupper_l(int c, locale_t locale);
```

69189 **DESCRIPTION**

69190 CX For `toupper()`: The functionality described on this reference page is aligned with the ISO C  
69191 standard. Any conflict between the requirements described here and the ISO C standard is  
69192 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

69193 CX The `toupper()` and `toupper_l()` functions have as a domain a type `int`, the value of which is  
69194 representable as an **unsigned char** or the value of EOF. If the argument has any other value, the  
69195 behavior is undefined.

69196 CX If the argument of `toupper()` or `toupper_l()` represents a lowercase letter, and there exists a  
69197 CX corresponding uppercase letter as defined by character type information in the current locale or  
69198 in the locale represented by `locale`, respectively (category `LC_CTYPE`), the result shall be the  
69199 corresponding uppercase letter.

69200 All other arguments in the domain are returned unchanged.

69201 CX The behavior is undefined if the `locale` argument to `toupper_l()` is the special locale object  
69202 `LC_GLOBAL_LOCALE` or is not a valid locale object handle.

69203 **RETURN VALUE**

69204 CX Upon successful completion, `toupper()` and `toupper_l()` shall return the uppercase letter  
69205 corresponding to the argument passed; otherwise, they shall return the argument unchanged.

69206 **ERRORS**

69207 No errors are defined.

69208 **EXAMPLES**

69209 None.

69210 **APPLICATION USAGE**

69211 None.

69212 **RATIONALE**

69213 None.

69214 **FUTURE DIRECTIONS**

69215 None.

69216 **SEE ALSO**

69217 [`setlocale\(\)`](#), [`uselocale\(\)`](#)

69218 XBD [Chapter 7](#) (on page 135), [`<ctype.h>`](#), [`<locale.h>`](#)

69219 **CHANGE HISTORY**

69220 First released in Issue 1. Derived from Issue 1 of the SVID.

69221 **Issue 6**

69222 Extensions beyond the ISO C standard are marked.

69223 **Issue 7**

69224 SD5-XSH-ERN-181 is applied, clarifying the RETURN VALUE section.

69225 The *toupper\_l()* function is added from The Open Group Technical Standard, 2006, Extended API  
69226 Set Part 4.

69227 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0673 [283] and XSH/TC1-2008/0674  
69228 [283] are applied.

69229 **NAME**

69230 towctrans, towctrans\_l — wide-character transliteration

69231 **SYNOPSIS**

69232 #include &lt;wctype.h&gt;

69233 wint\_t towctrans(wint\_t wc, wctrans\_t desc);

69234 CX wint\_t towctrans\_l(wint\_t wc, wctrans\_t desc,  
69235 locale\_t locale);69236 **DESCRIPTION**69237 CX For towctrans(): The functionality described on this reference page is aligned with the ISO C  
69238 standard. Any conflict between the requirements described here and the ISO C standard is  
69239 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.69240 CX The towctrans() and towctrans\_l() functions shall transliterate the wide-character code wc using  
69241 the mapping described by desc.69242 CX The current setting of the LC\_CTYPE category in the current locale or in the locale represented  
69243 CX by locale, respectively, should be the same as during the call to wctrans() or wctrans\_l() that  
69244 returned the value desc.69245 If the value of desc is invalid (that is, not obtained by a call to wctrans() or desc is invalidated by a  
69246 subsequent call to setlocale() that has affected category LC\_CTYPE), the result is unspecified.69247 CX If the value of desc is invalid (that is, not obtained by a call to wctrans\_l() with the same locale  
69248 object locale) the result is unspecified.69249 CX An application wishing to check for error situations should set errno to 0 before calling  
69250 towctrans() or towctrans\_l().

69251 If errno is non-zero on return, an error has occurred.

69252 The behavior is undefined if the locale argument to towctrans\_l() is the special locale object  
69253 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.69254 **RETURN VALUE**69255 CX If successful, the towctrans() and towctrans\_l() functions shall return the mapped value of wc  
69256 using the mapping described by desc. Otherwise, they shall return wc unchanged.69257 **ERRORS**

69258 These functions may fail if:

69259 CX [EINVAL] desc contains an invalid transliteration descriptor.

69260 **EXAMPLES**

69261 None.

69262 **APPLICATION USAGE**69263 The strings "tolower" and "toupper" are reserved for the standard mapping names. In the  
69264 table below, the functions in the left column are equivalent to the functions in the right column.

|                             |                                             |
|-----------------------------|---------------------------------------------|
| 69265 tolower(wc)           | towctrans(wc, wctrans("tolower"))           |
| 69266 tolower_l(wc, locale) | towctrans_l(wc, wctrans("tolower"), locale) |
| 69267 toupper(wc)           | towctrans(wc, wctrans("toupper"))           |
| 69268 toupper_l(wc, locale) | towctrans_l(wc, wctrans("toupper"), locale) |

69269 **RATIONALE**

69270 None.

69271 **FUTURE DIRECTIONS**

69272 None.

69273 **SEE ALSO**69274 *tolower()*, *towupper()*, *wctrans()*69275 XBD <**wctype.h**>69276 **CHANGE HISTORY**

69277 First released in Issue 5. Derived from ISO/IEC 9899:1990/Amendment 1:1995 (E).

69278 **Issue 6**

69279 Extensions beyond the ISO C standard are marked.

69280 **Issue 7**69281 The *towctrans\_l()* function is added from The Open Group Technical Standard, 2006, Extended  
69282 API Set Part 4.69283 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0675 [302], XSH/TC1-2008/0676 [283],  
69284 and XSH/TC1-2008/0677 [283] are applied.

69285 **NAME**

69286 towlower, towlower\_l — transliterate uppercase wide-character code to lowercase

69287 **SYNOPSIS**

69288 #include &lt;wctype.h&gt;

69289 wint\_t towlower(wint\_t wc);

69290 CX wint\_t towlower\_l(wint\_t wc, locale\_t locale);

69291 **DESCRIPTION**69292 CX For *towlower()*: The functionality described on this reference page is aligned with the ISO C  
69293 standard. Any conflict between the requirements described here and the ISO C standard is  
69294 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.69295 CX The *towlower()* and *towlower\_l()* functions have as a domain a type **wint\_t**, the value of which  
69296 the application shall ensure is a character representable as a **wchar\_t**, and a wide-character code  
69297 corresponding to a valid character in the locale used by the function or the value of WEOF. If  
69298 CX the argument has any other value, the behavior is undefined. If the argument of *towlower()* or  
69299 *towlower\_l()* represents an uppercase wide-character code, and there exists a corresponding  
69300 CX lowercase wide-character code as defined by character type information in the current locale or  
69301 in the locale represented by *locale*, respectively (category *LC\_CTYPE*), the result shall be the  
69302 corresponding lowercase wide-character code. All other arguments in the domain are returned  
69303 unchanged.69304 CX The behavior is undefined if the *locale* argument to *towlower\_l()* is the special locale object  
69305 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.69306 **RETURN VALUE**69307 CX Upon successful completion, the *towlower()* and *towlower\_l()* functions shall return the  
69308 lowercase letter corresponding to the argument passed; otherwise, they shall return the  
69309 argument unchanged.69310 **ERRORS**

69311 No errors are defined.

69312 **EXAMPLES**

69313 None.

69314 **APPLICATION USAGE**

69315 None.

69316 **RATIONALE**

69317 None.

69318 **FUTURE DIRECTIONS**

69319 None.

69320 **SEE ALSO**69321 *setlocale()*, *uselocale()*

69322 XBD Chapter 7 (on page 135), &lt;locale.h&gt;, &lt;wctype.h&gt;

69323 **CHANGE HISTORY**

69324 First released in Issue 4.



69325 **Issue 5**

69326 The following change has been made in this version for alignment with  
69327 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 69328       • The SYNOPSIS has been changed to indicate that this function and associated data types  
69329       are now made visible by inclusion of the **<wctype.h>** header rather than **<wchar.h>**.

69330 **Issue 6**

69331 The normative text is updated to avoid use of the term “must” for application requirements.

69332 **Issue 7**

69333 The *tolower\_l()* function is added from The Open Group Technical Standard, 2006, Extended  
69334 API Set Part 4.

69335 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0678 [302], XSH/TC1-2008/0679 [283],  
69336 and XSH/TC1-2008/0680 [283] are applied.

69337 **NAME**

69338 towupper, towupper\_l — transliterate lowercase wide-character code to uppercase

69339 **SYNOPSIS**

69340 #include &lt;wctype.h&gt;

69341 wint\_t towupper(wint\_t wc);

69342 CX wint\_t towupper\_l(wint\_t wc, locale\_t locale);

69343 **DESCRIPTION**69344 CX For *towupper()*: The functionality described on this reference page is aligned with the ISO C  
69345 standard. Any conflict between the requirements described here and the ISO C standard is  
69346 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.69347 CX The *towupper()* and *towupper\_l()* functions have as a domain a type **wint\_t**, the value of which  
69348 the application shall ensure is a character representable as a **wchar\_t**, and a wide-character code  
69349 corresponding to a valid character in the locale used by the function or the value of WEOF. If  
69350 CX the argument has any other value, the behavior is undefined. If the argument of *towupper()* or  
69351 *towupper\_l()* represents a lowercase wide-character code, and there exists a corresponding  
69352 CX uppercase wide-character code as defined by character type information in the current locale or  
69353 in the locale represented by *locale*, respectively (category *LC\_CTYPE*), the result shall be the  
69354 corresponding uppercase wide-character code. All other arguments in the domain are returned  
69355 unchanged.69356 CX The behavior is undefined if the *locale* argument to *towupper\_l()* is the special locale object  
69357 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.69358 **RETURN VALUE**69359 CX Upon successful completion, the *towupper()* and *towupper\_l()* functions shall return the  
69360 uppercase letter corresponding to the argument passed. Otherwise, they shall return the  
69361 argument unchanged.69362 **ERRORS**

69363 No errors are defined.

69364 **EXAMPLES**

69365 None.

69366 **APPLICATION USAGE**

69367 None.

69368 **RATIONALE**

69369 None.

69370 **FUTURE DIRECTIONS**

69371 None.

69372 **SEE ALSO**69373 *setlocale()*, *uselocale()*

69374 XBD Chapter 7 (on page 135), &lt;locale.h&gt;, &lt;wctype.h&gt;

69375 **CHANGE HISTORY**

69376 First released in Issue 4.

69377 **Issue 5**

69378 The following change has been made in this version for alignment with  
69379 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 69380 • The SYNOPSIS has been changed to indicate that this function and associated data types  
69381 are now made visible by inclusion of the `<wctype.h>` header rather than `<wchar.h>`.

69382 **Issue 6**

69383 The normative text is updated to avoid use of the term “must” for application requirements.

69384 **Issue 7**

69385 The `towupper_l()` function is added from The Open Group Technical Standard, 2006, Extended  
69386 API Set Part 4.

69387 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0681 [302], XSH/TC1-2008/0682 [283],  
69388 and XSH/TC1-2008/0683 [283] are applied.

69389 **NAME**

69390 trunc, truncf, trunc1 — round to truncated integer value

69391 **SYNOPSIS**

```
69392 #include <math.h>
69393 double trunc(double x);
69394 float truncf(float x);
69395 long double trunc1(long double x);
```

69396 **DESCRIPTION**

69397 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
69398 conflict between the requirements described here and the ISO C standard is unintentional. This  
69399 volume of POSIX.1-2008 defers to the ISO C standard.

69400 These functions shall round their argument to the integer value, in floating format, nearest to but  
69401 no larger in magnitude than the argument.

69402 **RETURN VALUE**

69403 Upon successful completion, these functions shall return the truncated integer value.

69404 MX The result shall have the same sign as *x*.

69405 MX If *x* is NaN, a NaN shall be returned.

69406 If *x* is  $\pm 0$  or  $\pm \text{Inf}$ , *x* shall be returned.

69407 **ERRORS**

69408 No errors are defined.

69409 **EXAMPLES**

69410 None.

69411 **APPLICATION USAGE**

69412 The integral value returned by these functions need not be expressible as an **intmax\_t**. The  
69413 return value should be tested before assigning it to an integer type to avoid the undefined  
69414 results of an integer overflow.

69415 These functions may raise the inexact floating-point exception if the result differs in value from  
69416 the argument.

69417 **RATIONALE**

69418 None.

69419 **FUTURE DIRECTIONS**

69420 None.

69421 **SEE ALSO**

69422 XBD [<math.h>](#)

69423 **CHANGE HISTORY**

69424 First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

69425 **Issue 7**

69426 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0684 [346] is applied.

69427 **NAME**

69428 truncate — truncate a file to a specified length

69429 **SYNOPSIS**

69430 #include &lt;unistd.h&gt;

69431 int truncate(const char \*path, off\_t length);

69432 **DESCRIPTION**69433 The *truncate()* function shall cause the regular file named by *path* to have a size which shall be  
69434 equal to *length* bytes.69435 If the file previously was larger than *length*, the extra data is discarded. If the file was previously  
69436 shorter than *length*, its size is increased, and the extended area appears as if it were zero-filled.

69437 The application shall ensure that the process has write permission for the file.

69438 XSI If the request would cause the file size to exceed the soft file size limit for the process, the  
69439 request shall fail and the implementation shall generate the SIGXFSZ signal for the process.69440 The *truncate()* function shall not modify the file offset for any open file descriptions associated  
69441 with the file. Upon successful completion, *truncate()* shall mark for update the last data  
69442 modification and last file status change timestamps of the file, and the S\_ISUID and S\_ISGID bits  
69443 of the file mode may be cleared.69444 **RETURN VALUE**69445 Upon successful completion, *truncate()* shall return 0. Otherwise, -1 shall be returned, and *errno*  
69446 set to indicate the error.69447 **ERRORS**69448 The *truncate()* function shall fail if:

69449 [EINTR] A signal was caught during execution.

69450 [EINVAL] The *length* argument was less than 0.

69451 [EFBIG] or [EINVAL]

69452 The *length* argument was greater than the maximum file size.

69453 [EIO] An I/O error occurred while reading from or writing to a file system.

69454 [EACCES] A component of the path prefix denies search permission, or write permission  
69455 is denied on the file.

69456 [EISDIR] The named file is a directory.

69457 [ELOOP] A loop exists in symbolic links encountered during resolution of the *path*  
69458 argument.

69459 [ENAMETOOLONG]

69460 The length of a component of a pathname is longer than {NAME\_MAX}.

69461 [ENOENT] A component of *path* does not name an existing file or *path* is an empty string.69462 [ENOTDIR] A component of the path prefix names an existing file that is neither a  
69463 directory nor a symbolic link to a directory, or the *path* argument contains at  
69464 least one non-*<slash>* character and ends with one or more trailing *<slash>*  
69465 characters and the last pathname component names an existing file that is  
69466 neither a directory nor a symbolic link to a directory.

69467 [EROFS] The named file resides on a read-only file system.  
69468 The *truncate()* function may fail if:  
69469 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
69470 resolution of the *path* argument.  
69471 [ENAMETOOLONG]  
69472 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
69473 symbolic link produced an intermediate result with a length that exceeds  
69474 {PATH\_MAX}.

#### 69475 EXAMPLES

69476 None.

#### 69477 APPLICATION USAGE

69478 None.

#### 69479 RATIONALE

69480 None.

#### 69481 FUTURE DIRECTIONS

69482 None.

#### 69483 SEE ALSO

69484 *open()*

69485 XBD <unistd.h>

#### 69486 CHANGE HISTORY

69487 First released in Issue 4, Version 2.

#### 69488 Issue 5

69489 Moved from X/OPEN UNIX extension to BASE.

69490 Large File Summit extensions are added.

#### 69491 Issue 6

69492 This reference page is split out from the *ftruncate()* reference page.

69493 The normative text is updated to avoid use of the term “must” for application requirements.

69494 The wording of the mandatory [ELOOP] error condition is updated, and a second optional  
69495 [ELOOP] error condition is added.

#### 69496 Issue 7

69497 Austin Group Interpretation 1003.1-2001 #143 is applied.

69498 The *truncate()* function is moved from the XSI option to the Base.

69499 Changes are made related to support for finegrained timestamps.

69500 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a  
69501 pathname exists but is not a directory or a symbolic link to a directory.

69502 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0685 [324] is applied.

69503 **NAME**

69504       truncf, trunc — round to truncated integer value

69505 **SYNOPSIS**

69506       #include &lt;math.h&gt;

69507       float truncf(float x);

69508       long double trunc1(long double x);

69509 **DESCRIPTION**69510       Refer to *trunc()*.

69511 **NAME**

69512       tsearch — search a binary search tree

69513 **SYNOPSIS**

```
69514 XSI      #include <search.h>
69515          void *tsearch(const void *key, void **rootp,
69516                      int (*compar)(const void *, const void *));
```

69517 **DESCRIPTION**69518       Refer to *tdelete()*.



**NAME**

ttyname, ttyname\_r — find the pathname of a terminal

**SYNOPSIS**

```
#include <unistd.h>
```

```
char *ttyname(int fildes);
```

```
int ttyname_r(int fildes, char *name, size_t namesize);
```

**DESCRIPTION**

The *ttyname()* function shall return a pointer to a string containing a null-terminated pathname of the terminal associated with file descriptor *fildes*. The application shall not modify the string returned. The returned pointer might be invalidated or the string content might be overwritten by a subsequent call to *ttyname()*. The returned pointer and the string content might also be invalidated if the calling thread is terminated.

The *ttyname()* function need not be thread-safe.

The *ttyname\_r()* function shall store the null-terminated pathname of the terminal associated with the file descriptor *fildes* in the character array referenced by *name*. The array is *namesize* characters long and should have space for the name and the terminating null character. The maximum length of the terminal name shall be {TTY\_NAME\_MAX}.

**RETURN VALUE**

Upon successful completion, *ttyname()* shall return a pointer to a string. Otherwise, a null pointer shall be returned and *errno* set to indicate the error.

If successful, the *ttyname\_r()* function shall return zero. Otherwise, an error number shall be returned to indicate the error.

**ERRORS**

The *ttyname()* function may fail if:

[EBADF] The *fildes* argument is not a valid file descriptor.

[ENOTTY] The file associated with the *fildes* argument is not a terminal.

The *ttyname\_r()* function may fail if:

[EBADF] The *fildes* argument is not a valid file descriptor.

[ENOTTY] The file associated with the *fildes* argument is not a terminal.

[ERANGE] The value of *namesize* is smaller than the length of the string to be returned including the terminating null character.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

The term “terminal” is used instead of the historical term “terminal device” in order to avoid a reference to an undefined term.

The thread-safe version places the terminal name in a user-supplied buffer and returns a non-zero value if it fails. The non-thread-safe version may return the name in a static data area that may be overwritten by each call.

## 69560 FUTURE DIRECTIONS

69561 None.

## 69562 SEE ALSO

69563 XBD <unistd.h>

## 69564 CHANGE HISTORY

69565 First released in Issue 1. Derived from Issue 1 of the SVID.

### 69566 Issue 5

69567 The *ttynamename\_r()* function is included for alignment with the POSIX Threads Extension.

69568 A note indicating that the *ttynamename()* function need not be reentrant is added to the  
69569 DESCRIPTION.

### 69570 Issue 6

69571 The *ttynamename\_r()* function is marked as part of the Thread-Safe Functions option.

69572 The following new requirements on POSIX implementations derive from alignment with the  
69573 Single UNIX Specification:

- 69574 • The statement that *errno* is set on error is added.
- 69575 • The [EBADF] and [ENOTTY] optional error conditions are added.

### 69576 Issue 7

69577 Austin Group Interpretation 1003.1-2001 #156 is applied.

69578 SD5-XSH-ERN-100 is applied, correcting the definition of the [ENOTTY] error condition.

69579 The *ttynamename\_r()* function is moved from the Thread-Safe Functions option to the Base.

69580 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0686 [75] is applied.

69581 **NAME**

69582 twalk — traverse a binary search tree

69583 **SYNOPSIS**

```
69584 XSI      #include <search.h>
69585          void twalk(const void *root,
69586                   void (*action)(const void *, VISIT, int ));
```

69587 **DESCRIPTION**69588 Refer to *tdelete()*.

**NAME**

daylight, timezone, tzname, tzset — set timezone conversion information

**SYNOPSIS**

```
#include <time.h>
```

```
extern int daylight;
extern long timezone;
extern char *tzname[2];
void tzset(void);
```

**DESCRIPTION**

The `tzset()` function shall use the value of the environment variable `TZ` to set time conversion information used by `ctime()`, `localtime()`, `mktime()`, and `strftime()`. If `TZ` is absent from the environment, implementation-defined default timezone information shall be used.

The `tzset()` function shall set the external variable `tzname` as follows:

```
tzname[0] = "std";
tzname[1] = "dst";
```

where `std` and `dst` are as described in XBD Chapter 8 (on page 173).

The `tzset()` function also shall set the external variable `daylight` to 0 if Daylight Savings Time conversions should never be applied for the timezone in use; otherwise, non-zero. The external variable `timezone` shall be set to the difference, in seconds, between Coordinated Universal Time (UTC) and local standard time.

If a thread accesses `tzname`, `daylight`, or `timezone` directly while another thread is in a call to `tzset()`, or to any function that is required or allowed to set timezone information as if by calling `tzset()`, the behavior is undefined.

**RETURN VALUE**

The `tzset()` function shall not return a value.

**ERRORS**

No errors are defined.

**EXAMPLES**

Example `TZ` variables and their timezone differences are given in the table below:

| <i>TZ</i> | <i>timezone</i> |
|-----------|-----------------|
| EST5EDT   | 5*60*60         |
| GMT0      | 0*60*60         |
| JST-9     | -9*60*60        |
| MET-1MEST | -1*60*60        |
| MST7MDT   | 7*60*60         |
| PST8PDT   | 8*60*60         |

**APPLICATION USAGE**

Since the `ctime()`, `localtime()`, `mktime()`, `strftime()`, and `strftime_l()` functions are required to set timezone information as if by calling `tzset()`, there is no need for an explicit `tzset()` call before using these functions. However, portable applications should call `tzset()` explicitly before using `ctime_r()` or `localtime_r()` because setting timezone information is optional for those functions.

69630 **RATIONALE**

69631 None.

69632 **FUTURE DIRECTIONS**

69633 None.

69634 **SEE ALSO**69635 *ctime()*, *localtime()*, *mktime()*, *strftime()*69636 XBD Chapter 8 (on page 173), **<time.h>**69637 **CHANGE HISTORY**

69638 First released in Issue 1. Derived from Issue 1 of the SVID.

69639 **Issue 6**

69640 The example is corrected.

69641 **NAME**

69642 ulimit — get and set process limits

69643 **SYNOPSIS**

```
69644 OB XSI #include <ulimit.h>
69645 long ulimit(int cmd, ...);
```

69646 **DESCRIPTION**

69647 The *ulimit()* function shall control process limits. The process limits that can be controlled by  
 69648 this function include the maximum size of a single file that can be written (this is equivalent to  
 69649 using *setrlimit()* with *RLIMIT\_FSIZE*). The *cmd* values, defined in **<ulimit.h>**, include:

69650 **UL\_GETFSIZE** Return the file size limit (*RLIMIT\_FSIZE*) of the process. The limit shall be in  
 69651 units of 512-byte blocks and shall be inherited by child processes. Files of any  
 69652 size can be read. The return value shall be the integer part of the soft file size  
 69653 limit divided by 512. If the result cannot be represented as a **long**, the result is  
 69654 unspecified.

69655 **UL\_SETFSIZE** Set the file size limit for output operations of the process to the value of the  
 69656 second argument, taken as a **long**, multiplied by 512. If the result would  
 69657 overflow an **rlim\_t**, the actual value set is unspecified. Any process may  
 69658 decrease its own limit, but only a process with appropriate privileges may  
 69659 increase the limit. The return value shall be the integer part of the new file size  
 69660 limit divided by 512.

69661 The *ulimit()* function shall not change the setting of *errno* if successful.

69662 As all return values are permissible in a successful situation, an application wishing to check for  
 69663 error situations should set *errno* to 0, then call *ulimit()*, and, if it returns  $-1$ , check to see if *errno* is  
 69664 non-zero.

69665 **RETURN VALUE**

69666 Upon successful completion, *ulimit()* shall return the value of the requested limit. Otherwise,  $-1$   
 69667 shall be returned and *errno* set to indicate the error.

69668 **ERRORS**

69669 The *ulimit()* function shall fail and the limit shall be unchanged if:

|       |          |                                                                                       |
|-------|----------|---------------------------------------------------------------------------------------|
| 69670 | [EINVAL] | The <i>cmd</i> argument is not valid.                                                 |
| 69671 | [EPERM]  | A process not having appropriate privileges attempts to increase its file size limit. |
| 69672 |          |                                                                                       |

69673 **EXAMPLES**

69674 None.

69675 **APPLICATION USAGE**

69676 Since the *ulimit()* function uses type **long** rather than **rlim\_t**, this function is not sufficient for file  
 69677 sizes on many current systems. Applications should use the *getrlimit()* or *setrlimit()* functions  
 69678 instead of the obsolescent *ulimit()* function.

69679 **RATIONALE**

69680 None.

69681 **FUTURE DIRECTIONS**

69682       The *ulimit()* function may be removed in a future version.

69683 **SEE ALSO**

69684       *exec*, *getrlimit()*, *write()*

69685       XBD <ulimit.h>

69686 **CHANGE HISTORY**

69687       First released in Issue 1. Derived from Issue 1 of the SVID.

69688 **Issue 5**

69689       In the description of UL\_SETFSIZE, the text is corrected to refer to **rlim\_t** rather than the  
69690       spurious **rlimit\_t**.

69691       The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

69692 **Issue 7**

69693       The *ulimit()* function is marked obsolescent.

**NAME**

umask — set and get the file mode creation mask

**SYNOPSIS**

```
#include <sys/stat.h>
```

```
mode_t umask(mode_t cmask);
```

**DESCRIPTION**

The *umask()* function shall set the file mode creation mask of the process to *cmask* and return the previous value of the mask. Only the file permission bits of *cmask* (see <sys/stat.h>) are used; the meaning of the other bits is implementation-defined.

The file mode creation mask of the process is used to turn off permission bits in the *mode* argument supplied during calls to the following functions:

- *open()*, *openat()*, *creat()*, *mkdir()*, *mkdirat()*, *mkfifo()*, and *mkfifoat()*
- *mknod()*, *mknodat()*
- *mq\_open()*
- *sem\_open()*

Bit positions that are set in *cmask* are cleared in the mode of the created file.

**RETURN VALUE**

The file permission bits in the value returned by *umask()* shall be the previous value of the file mode creation mask. The state of any other bits in that value is unspecified, except that a subsequent call to *umask()* with the returned value as *cmask* shall leave the state of the mask the same as its state before the first call, including any unspecified use of those bits.

**ERRORS**

No errors are defined.

**EXAMPLES**

None.

**APPLICATION USAGE**

None.

**RATIONALE**

Unsigned argument and return types for *umask()* were proposed. The return type and the argument were both changed to **mode\_t**.

Historical implementations have made use of additional bits in *cmask* for their implementation-defined purposes. The addition of the text that the meaning of other bits of the field is implementation-defined permits these implementations to conform to this volume of POSIX.1-2008.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*creat()*, *exec*, *mkdir()*, *mkfifo()*, *mknod()*, *mq\_open()*, *open()*, *sem\_open()*

XBD <sys/stat.h>, <sys/types.h>



**CHANGE HISTORY**

First released in Issue 1. Derived from Issue 1 of the SVID.

**Issue 6**

In the SYNOPSIS, the optional include of the `<sys/types.h>` header is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include `<sys/types.h>` has been removed. Although `<sys/types.h>` was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/143 is applied, adding the `mknod()`, `mq_open()`, and `sem_open()` functions to the DESCRIPTION and SEE ALSO sections.

69744 **NAME**69745        `uname` — get the name of the current system69746 **SYNOPSIS**69747        `#include <sys/utsname.h>`69748        `int uname(struct utsname *name);`69749 **DESCRIPTION**69750        The `uname()` function shall store information identifying the current system in the structure pointed to by `name`.69752        The `uname()` function uses the **utsname** structure defined in `<sys/utsname.h>`.69753        The `uname()` function shall return a string naming the current system in the character array `sysname`. Similarly, `nodename` shall contain the name of this node within an implementation-defined communications network. The arrays `release` and `version` shall further identify the operating system. The array `machine` shall contain a name that identifies the hardware that the system is running on.

69758        The format of each member is implementation-defined.

69759 **RETURN VALUE**69760        Upon successful completion, a non-negative value shall be returned. Otherwise, `-1` shall be returned and `errno` set to indicate the error.69762 **ERRORS**

69763        No errors are defined.

69764 **EXAMPLES**

69765        None.

69766 **APPLICATION USAGE**69767        The inclusion of the `nodename` member in this structure does not imply that it is sufficient information for interfacing to communications networks.69769 **RATIONALE**69770        The values of the structure members are not constrained to have any relation to the version of this volume of POSIX.1-2008 implemented in the operating system. An application should instead depend on `_POSIX_VERSION` and related constants defined in `<unistd.h>`.69773        This volume of POSIX.1-2008 does not define the sizes of the members of the structure and permits them to be of different sizes, although most implementations define them all to be the same size: eight bytes plus one byte for the string terminator. That size for `nodename` is not enough for use with many networks.69777        The `uname()` function originated in System III, System V, and related implementations, and it does not exist in Version 7 or 4.3 BSD. The values it returns are set at system compile time in those historical implementations.69780        4.3 BSD has `gethostname()` and `gethostid()`, which return a symbolic name and a numeric value, respectively. There are related `sethostname()` and `sethostid()` functions that are used to set the values the other two functions return. The former functions are included in this specification, the latter are not.69784 **FUTURE DIRECTIONS**

69785        None.

69786 **SEE ALSO**69787 XBD [<sys/utsname.h>](#)69788 **CHANGE HISTORY**

69789 First released in Issue 1. Derived from Issue 1 of the SVID.

69790 **NAME**

69791 ungetc — push byte back into input stream

69792 **SYNOPSIS**

69793 #include &lt;stdio.h&gt;

69794 int ungetc(int *c*, FILE \**stream*);69795 **DESCRIPTION**

69796 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
69797 conflict between the requirements described here and the ISO C standard is unintentional. This  
69798 volume of POSIX.1-2008 defers to the ISO C standard.

69799 The *ungetc()* function shall push the byte specified by *c* (converted to an **unsigned char**) back  
69800 onto the input stream pointed to by *stream*. The pushed-back bytes shall be returned by  
69801 subsequent reads on that stream in the reverse order of their pushing. A successful intervening  
69802 CX call (with the stream pointed to by *stream*) to a file-positioning function (*fseek()*, *fseeko()*,  
69803 CX *fsetpos()*, or *rewind()*) or *fflush()* shall discard any pushed-back bytes for the stream. The  
69804 external storage corresponding to the stream shall be unchanged.

69805 One byte of push-back shall be provided. If *ungetc()* is called too many times on the same stream  
69806 without an intervening read or file-positioning operation on that stream, the operation may fail.

69807 If the value of *c* equals that of the macro EOF, the operation shall fail and the input stream shall  
69808 be left unchanged.

69809 A successful call to *ungetc()* shall clear the end-of-file indicator for the stream. The value of the  
69810 file-position indicator for the stream after all pushed-back bytes have been read, or discarded by  
69811 CX calling *fseek()*, *fseeko()*, *fsetpos()*, or *rewind()* (but not *fflush()*), shall be the same as it was before  
69812 the bytes were pushed back. The file-position indicator is decremented by each successful call to  
69813 *ungetc()*; if its value was 0 before a call, its value is unspecified after the call.

69814 **RETURN VALUE**

69815 Upon successful completion, *ungetc()* shall return the byte pushed back after conversion.  
69816 Otherwise, it shall return EOF.

69817 **ERRORS**

69818 No errors are defined.

69819 **EXAMPLES**

69820 None.

69821 **APPLICATION USAGE**

69822 None.

69823 **RATIONALE**

69824 None.

69825 **FUTURE DIRECTIONS**

69826 None.

69827 **SEE ALSO**69828 Section 2.5 (on page 495), *fseek()*, *getc()*, *fsetpos()*, *read()*, *rewind()*, *setbuf()*

69829 XBD &lt;stdio.h&gt;

69830 **CHANGE HISTORY**

69831 First released in Issue 1. Derived from Issue 1 of the SVID.

69832 **Issue 7**

69833

69834

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0687 [87,93], XSH/TC1-2008/0688 [87], and XSH/TC1-2008/0689 [14] are applied.

69835 **NAME**

69836 ungetwc — push wide-character code back into the input stream

69837 **SYNOPSIS**

69838 #include &lt;stdio.h&gt;

69839 #include &lt;wchar.h&gt;

69840 wint\_t ungetwc(wint\_t wc, FILE \*stream);

69841 **DESCRIPTION**

69842 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 69843 conflict between the requirements described here and the ISO C standard is unintentional. This  
 69844 volume of POSIX.1-2008 defers to the ISO C standard.

69845 The *ungetwc()* function shall push the character corresponding to the wide-character code  
 69846 specified by *wc* back onto the input stream pointed to by *stream*. The pushed-back characters  
 69847 shall be returned by subsequent reads on that stream in the reverse order of their pushing. A  
 69848 successful intervening call (with the stream pointed to by *stream*) to a file-positioning function  
 69849 CX (*fseek()*, *fseeko()*, *fsetpos()*, or *rewind()*) or *fflush()* shall discard any pushed-back characters for  
 69850 the stream. The external storage corresponding to the stream is unchanged.

69851 At least one character of push-back shall be provided. If *ungetwc()* is called too many times on  
 69852 the same stream without an intervening read or file-positioning operation on that stream, the  
 69853 operation may fail.

69854 If the value of *wc* equals that of the macro WEOF, the operation shall fail and the input stream  
 69855 shall be left unchanged.

69856 A successful call to *ungetwc()* shall clear the end-of-file indicator for the stream. The value of the  
 69857 file-position indicator for the stream after all pushed-back characters have been read, or  
 69858 CX discarded by calling *fseek()*, *fseeko()*, *fsetpos()*, or *rewind()* (but not *fflush()*), shall be the same as  
 69859 it was before the characters were pushed back. The file-position indicator is decremented (by  
 69860 one or more) by each successful call to *ungetwc()*; if its value was 0 before a call, its value is  
 69861 unspecified after the call.

69862 **RETURN VALUE**

69863 Upon successful completion, *ungetwc()* shall return the wide-character code corresponding to  
 69864 the pushed-back character. Otherwise, it shall return WEOF.

69865 **ERRORS**69866 The *ungetwc()* function may fail if:

69867 CX [EILSEQ] An invalid character sequence is detected, or a wide-character code does not  
 69868 correspond to a valid character.

69869 **EXAMPLES**

69870 None.

69871 **APPLICATION USAGE**

69872 None.

69873 **RATIONALE**

69874 None.

69875 **FUTURE DIRECTIONS**

69876 None.

69877 **SEE ALSO**69878       Section 2.5 (on page 495), *fseek()*, *fsetpos()*, *read()*, *rewind()*, *setbuf()*69879       XBD **<stdio.h>**, **<wchar.h>**69880 **CHANGE HISTORY**

69881       First released in Issue 4. Derived from the MSE working draft.

69882 **Issue 5**69883       The Optional Header (OH) marking is removed from **<stdio.h>**.69884 **Issue 6**

69885       The [EILSEQ] optional error condition is marked CX.

69886 **Issue 7**69887       POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0690 [87,93], XSH/TC1-2008/0691  
69888       [87], and XSH/TC1-2008/0692 [14] are applied.

## NAME

unlink, unlinkat — remove a directory entry

## SYNOPSIS

```
#include <unistd.h>
```

```
int unlink(const char *path);
```

```
#include <fcntl.h>
```

```
int unlinkat(int fd, const char *path, int flag);
```

## DESCRIPTION

The *unlink()* function shall remove a link to a file. If *path* names a symbolic link, *unlink()* shall remove the symbolic link named by *path* and shall not affect any file or directory named by the contents of the symbolic link. Otherwise, *unlink()* shall remove the link named by the pathname pointed to by *path* and shall decrement the link count of the file referenced by the link.

When the file's link count becomes 0 and no process has the file open, the space occupied by the file shall be freed and the file shall no longer be accessible. If one or more processes have the file open when the last link is removed, the link shall be removed before *unlink()* returns, but the removal of the file contents shall be postponed until all references to the file are closed.

The *path* argument shall not name a directory unless the process has appropriate privileges and the implementation supports using *unlink()* on directories.

Upon successful completion, *unlink()* shall mark for update the last data modification and last file status change timestamps of the parent directory. Also, if the file's link count is not 0, the last file status change timestamp of the file shall be marked for update.

The *unlinkat()* function shall be equivalent to the *unlink()* or *rmdir()* function except in the case where *path* specifies a relative path. In this case the directory entry to be removed is determined relative to the directory associated with the file descriptor *fd* instead of the current working directory. If the access mode of the open file description associated with the file descriptor is not O\_SEARCH, the function shall check whether directory searches are permitted using the current permissions of the directory underlying the file descriptor. If the access mode is O\_SEARCH, the function shall not perform the check.

Values for *flag* are constructed by a bitwise-inclusive OR of flags from the following list, defined in <fcntl.h>:

AT\_REMOVEDIR

Remove the directory entry specified by *fd* and *path* as a directory, not a normal file.

If *unlinkat()* is passed the special value AT\_FDCWD in the *fd* parameter, the current working directory shall be used and the behavior shall be identical to a call to *unlink()* or *rmdir()* respectively, depending on whether or not the AT\_REMOVEDIR bit is set in *flag*.

## RETURN VALUE

Upon successful completion, these functions shall return 0. Otherwise, these functions shall return -1 and set *errno* to indicate the error. If -1 is returned, the named file shall not be changed.

## ERRORS

These functions shall fail and shall not unlink the file if:

[EACCES] Search permission is denied for a component of the path prefix, or write permission is denied on the directory containing the directory entry to be removed.



|       |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------|-------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 69933 | [EBUSY]                 | The file named by the <i>path</i> argument cannot be unlinked because it is being used by the system or another process and the implementation considers this an error.                                                                                                                                                                                                                                           |
| 69934 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69935 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69936 | [ELOOP]                 | A loop exists in symbolic links encountered during resolution of the <i>path</i> argument.                                                                                                                                                                                                                                                                                                                        |
| 69937 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69938 | [ENAMETOOLONG]          |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69939 |                         | The length of a component of a pathname is longer than {NAME_MAX}.                                                                                                                                                                                                                                                                                                                                                |
| 69940 | [ENOENT]                | A component of <i>path</i> does not name an existing file or <i>path</i> is an empty string.                                                                                                                                                                                                                                                                                                                      |
| 69941 | [ENOTDIR]               | A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the <i>path</i> argument contains at least one non- <code>&lt;slash&gt;</code> character and ends with one or more trailing <code>&lt;slash&gt;</code> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory. |
| 69942 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69943 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69944 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69945 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69946 | [EPERM]                 | The file named by <i>path</i> is a directory, and either the calling process does not have appropriate privileges, or the implementation prohibits using <i>unlink()</i> on directories.                                                                                                                                                                                                                          |
| 69947 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69948 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69949 | XSI [EPERM] or [EACCES] |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69950 |                         | The S_ISVTX flag is set on the directory containing the file referred to by the <i>path</i> argument and the process does not satisfy the criteria specified in XBD <a href="#">Section 4.3</a> (on page 108).                                                                                                                                                                                                    |
| 69951 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69952 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69953 | [EROFS]                 | The directory entry to be unlinked is part of a read-only file system.                                                                                                                                                                                                                                                                                                                                            |
| 69954 |                         | The <i>unlinkat()</i> function shall fail if:                                                                                                                                                                                                                                                                                                                                                                     |
| 69955 | [EACCES]                | The access mode of the open file description associated with <i>fd</i> is not O_SEARCH and the permissions of the directory underlying <i>fd</i> do not permit directory searches.                                                                                                                                                                                                                                |
| 69956 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69957 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69958 | [EBADF]                 | The <i>path</i> argument does not specify an absolute path and the <i>fd</i> argument is neither AT_FDCWD nor a valid file descriptor open for reading or searching.                                                                                                                                                                                                                                              |
| 69959 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69960 | [ENOTDIR]               | The <i>path</i> argument is not an absolute path and <i>fd</i> is a file descriptor associated with a non-directory file.                                                                                                                                                                                                                                                                                         |
| 69961 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69962 | [EEXIST] or [ENOTEMPTY] |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69963 |                         | The <i>flag</i> parameter has the AT_REMOVEDIR bit set and the <i>path</i> argument names a directory that is not an empty directory, or there are hard links to the directory other than dot or a single entry in dot-dot.                                                                                                                                                                                       |
| 69964 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69965 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69966 | [ENOTDIR]               | The <i>flag</i> parameter has the AT_REMOVEDIR bit set and <i>path</i> does not name a directory.                                                                                                                                                                                                                                                                                                                 |
| 69967 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69968 |                         | These functions may fail and not unlink the file if:                                                                                                                                                                                                                                                                                                                                                              |
| 69969 | XSI [EBUSY]             | The file named by <i>path</i> is a named STREAM.                                                                                                                                                                                                                                                                                                                                                                  |
| 69970 | [ELOOP]                 | More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the <i>path</i> argument.                                                                                                                                                                                                                                                                                                            |
| 69971 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69972 | [ENAMETOOLONG]          |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69973 |                         | The length of a pathname exceeds {PATH_MAX}, or pathname resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.                                                                                                                                                                                                                                                     |
| 69974 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 69975 |                         |                                                                                                                                                                                                                                                                                                                                                                                                                   |

69976 [ETXTBSY] The entry to be unlinked is the last directory entry to a pure procedure (shared  
69977 text) file that is being executed.

69978 The *unlinkat()* function may fail if:

69979 [EINVAL] The value of the *flag* argument is not valid.

## 69980 EXAMPLES

### 69981 Removing a Link to a File

69982 The following example shows how to remove a link to a file named */home/cnd/mod1* by  
69983 removing the entry named */modules/pass1*.

```
69984 #include <unistd.h>
69985 char *path = "/modules/pass1";
69986 int status;
69987 ...
69988 status = unlink(path);
```

### 69989 Checking for an Error

69990 The following example fragment creates a temporary password lock file named **LOCKFILE**,  
69991 which is defined as */etc/ptmp*, and gets a file descriptor for it. If the file cannot be opened for  
69992 writing, *unlink()* is used to remove the link between the file descriptor and **LOCKFILE**.

```
69993 #include <sys/types.h>
69994 #include <stdio.h>
69995 #include <fcntl.h>
69996 #include <errno.h>
69997 #include <unistd.h>
69998 #include <sys/stat.h>
69999 #define LOCKFILE "/etc/ptmp"
70000 int pfd; /* Integer for file descriptor returned by open call. */
70001 FILE *fpfd; /* File pointer for use in putpwent(). */
70002 ...
70003 /* Open password Lock file. If it exists, this is an error. */
70004 if ((pfd = open(LOCKFILE, O_WRONLY| O_CREAT | O_EXCL, S_IRUSR
70005 | S_IWUSR | S_IRGRP | S_IROTH)) == -1) {
70006     fprintf(stderr, "Cannot open /etc/ptmp. Try again later.\n");
70007     exit(1);
70008 }
70009 /* Lock file created; proceed with fdopen of lock file so that
70010 putpwent() can be used.
70011 */
70012 if ((fpfd = fdopen(pfd, "w")) == NULL) {
70013     close(pfd);
70014     unlink(LOCKFILE);
70015     exit(1);
70016 }
```

## Replacing Files

The following example fragment uses *unlink()* to discard links to files, so that they can be replaced with new versions of the files. The first call removes the link to **LOCKFILE** if an error occurs. Successive calls remove the links to **SAVEFILE** and **PASSWDFILE** so that new links can be created, then removes the link to **LOCKFILE** when it is no longer needed.

```
#include <sys/types.h>
#include <stdio.h>
#include <fcntl.h>
#include <errno.h>
#include <unistd.h>
#include <sys/stat.h>

#define LOCKFILE "/etc/ptmp"
#define PASSWDFILE "/etc/passwd"
#define SAVEFILE "/etc/opasswd"
...
/* If no change was made, assume error and leave passwd unchanged. */
if (!valid_change) {
    fprintf(stderr, "Could not change password for user %s\n", user);
    unlink(LOCKFILE);
    exit(1);
}

/* Change permissions on new password file. */
chmod(LOCKFILE, S_IRUSR | S_IRGRP | S_IROTH);

/* Remove saved password file. */
unlink(SAVEFILE);

/* Save current password file. */
link(PASSWDFILE, SAVEFILE);

/* Remove current password file. */
unlink(PASSWDFILE);

/* Save new password file as current password file. */
link(LOCKFILE, PASSWDFILE);

/* Remove lock file. */
unlink(LOCKFILE);

exit(0);
```

## APPLICATION USAGE

Applications should use *rmdir()* to remove a directory.

## RATIONALE

Unlinking a directory is restricted to the superuser in many historical implementations for reasons given in *link()* (see also *rename()*).

The meaning of [EBUSY] in historical implementations is “mount point busy”. Since this volume of POSIX.1-2008 does not cover the system administration concepts of mounting and unmounting, the description of the error was changed to “resource busy”. (This meaning is used by some device drivers when a second process tries to open an exclusive use device.) The wording is also intended to allow implementations to refuse to remove a directory if it is the root or current working directory of any process.

70062 The standard developers reviewed TR 24715-2006 and noted that LSB-conforming  
70063 implementations may return [EISDIR] instead of [EPERM] when unlinking a directory. A change  
70064 to permit this behavior by changing the requirement for [EPERM] to [EPERM] or [EISDIR] was  
70065 considered, but decided against since it would break existing strictly conforming and  
70066 conforming applications. Applications written for portability to both POSIX.1-2008 and the LSB  
70067 should be prepared to handle either error code.

70068 The purpose of the *unlinkat()* function is to remove directory entries in directories other than the  
70069 current working directory without exposure to race conditions. Any part of the path of a file  
70070 could be changed in parallel to a call to *unlink()*, resulting in unspecified behavior. By opening a  
70071 file descriptor for the target directory and using the *unlinkat()* function it can be guaranteed that  
70072 the removed directory entry is located relative to the desired directory.

#### 70073 FUTURE DIRECTIONS

70074 None.

#### 70075 SEE ALSO

70076 *close()*, *link()*, *remove()*, *rename()*, *rmdir()*, *symlink()*

70077 XBD Section 4.3 (on page 108), *<fcntl.h>*, *<unistd.h>*

#### 70078 CHANGE HISTORY

70079 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 70080 Issue 5

70081 The [EBUSY] error is added to the optional part of the ERRORS section.

#### 70082 Issue 6

70083 The following new requirements on POSIX implementations derive from alignment with the  
70084 Single UNIX Specification:

- 70085 • In the DESCRIPTION, the effect is specified if *path* specifies a symbolic link.
- 70086 • The [ELOOP] mandatory error condition is added.
- 70087 • A second [ENAMETOOLONG] is added as an optional error condition.
- 70088 • The [ETXTBSY] optional error condition is added.

70089 The following changes were made to align with the IEEE P1003.1a draft standard:

- 70090 • The [ELOOP] optional error condition is added.

70091 The normative text is updated to avoid use of the term “must” for application requirements.

#### 70092 Issue 7

70093 Austin Group Interpretation 1003.1-2001 #143 is applied.

70094 Austin Group Interpretation 1003.1-2001 #181 is applied, updating the requirements for  
70095 operations when the S\_ISVTX bit is set.

70096 Text arising from the LSB Conflicts TR is added to the RATIONALE about the use of [EPERM]  
70097 and [EISDIR].

70098 The *unlinkat()* function is added from The Open Group Technical Standard, 2006, Extended API  
70099 Set Part 2.

70100 Changes are made related to support for finegrained timestamps.

70101 Changes are made to allow a directory to be opened for searching.

70102 The [ENOTDIR] error condition is clarified to cover the condition where the last component of a  
70103 pathname exists but is not a directory or a symbolic link to a directory.  
70104 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0693 [461], XSH/TC1-2008/0694 [324],  
70105 XSH/TC1-2008/0695 [278], and XSH/TC1-2008/0696 [278] are applied.

70106 **NAME**

70107 unlockpt — unlock a pseudo-terminal master/slave pair

70108 **SYNOPSIS**

```
70109 XSI #include <stdlib.h>  
70110 int unlockpt(int fildes);
```

70111 **DESCRIPTION**

70112 The *unlockpt()* function shall unlock the slave pseudo-terminal device associated with the master  
70113 to which *fildes* refers.

70114 Conforming applications shall ensure that they call *unlockpt()* before opening the slave side of a  
70115 pseudo-terminal device.

70116 **RETURN VALUE**

70117 Upon successful completion, *unlockpt()* shall return 0. Otherwise, it shall return -1 and set *errno*  
70118 to indicate the error.

70119 **ERRORS**

70120 The *unlockpt()* function may fail if:

70121 [EBADF] The *fildes* argument is not a file descriptor open for writing.

70122 [EINVAL] The *fildes* argument is not associated with a master pseudo-terminal device.

70123 **EXAMPLES**

70124 None.

70125 **APPLICATION USAGE**

70126 None.

70127 **RATIONALE**

70128 See the RATIONALE section for *posix\_openpt()*.

70129 **FUTURE DIRECTIONS**

70130 None.

70131 **SEE ALSO**

70132 *grantpt()*, *open()*, *posix\_openpt()*, *ptsname()*

70133 XBD <stdlib.h>

70134 **CHANGE HISTORY**

70135 First released in Issue 4, Version 2.

70136 **Issue 5**

70137 Moved from X/OPEN UNIX extension to BASE.

70138 **Issue 6**

70139 The normative text is updated to avoid use of the term “must” for application requirements.

70140 **Issue 7**

70141 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0697 [96] is applied.

70142 **NAME**

70143       unsetenv — remove an environment variable

70144 **SYNOPSIS**

```
70145 CX      #include <stdlib.h>
70146      int unsetenv(const char *name);
```

70147 **DESCRIPTION**

70148       The *unsetenv()* function shall remove an environment variable from the environment of the  
 70149       calling process. The *name* argument points to a string, which is the name of the variable to be  
 70150       removed. The named argument shall not contain an '=' character. If the named variable does  
 70151       not exist in the current environment, the environment shall be unchanged and the function is  
 70152       considered to have completed successfully.

70153       The *unsetenv()* function shall update the list of pointers to which *environ* points.

70154       The *unsetenv()* function need not be thread-safe.

70155 **RETURN VALUE**

70156       Upon successful completion, zero shall be returned. Otherwise, -1 shall be returned, *errno* set to  
 70157       indicate the error, and the environment shall be unchanged.

70158 **ERRORS**

70159       The *unsetenv()* function shall fail if:

|       |          |                                                                                      |
|-------|----------|--------------------------------------------------------------------------------------|
| 70160 | [EINVAL] | The <i>name</i> argument points to an empty string, or points to a string containing |
| 70161 |          | an '=' character.                                                                    |

70162 **EXAMPLES**

70163       None.

70164 **APPLICATION USAGE**

70165       None.

70166 **RATIONALE**

70167       Refer to the RATIONALE section in *setenv()*.

70168 **FUTURE DIRECTIONS**

70169       None.

70170 **SEE ALSO**

70171       *getenv()*, *setenv()*

70172       XBD *<stdlib.h>*, *<sys/types.h>*

70173 **CHANGE HISTORY**

70174       First released in Issue 6. Derived from the IEEE P1003.1a draft standard.

70175 **Issue 7**

70176       Austin Group Interpretation 1003.1-2001 #156 is applied.

70177       POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0698 [167] and XSH/TC1-2008/0699  
 70178       [185] are applied.

**NAME**

uselocale — use locale in current thread

**SYNOPSIS**

```
CX      #include <locale.h>
       locale_t uselocale(locale_t newloc);
```

**DESCRIPTION**

The *uselocale()* function shall set or query the current locale for the calling thread.

The value for the *newloc* argument shall be one of the following:

1. A value returned by the *newlocale()* or *duplocale()* functions
2. The special locale object descriptor `LC_GLOBAL_LOCALE`
3. `(locale_t)0`

If the *newloc* argument is `(locale_t)0`, the current locale shall not be changed; this value can be used to query the current locale setting. If the *newloc* argument is `LC_GLOBAL_LOCALE`, any thread-local locale for the calling thread shall be uninstalled; the thread shall again use the global locale as the current locale, and changes to the global locale shall affect the thread. Otherwise, the locale represented by *newloc* shall be installed as a thread-local locale to be used as the current locale for the calling thread.

Once the *uselocale()* function has been called to install a thread-local locale, the behavior of every interface using data from the current locale shall be affected for the calling thread. The current locale for other threads shall remain unchanged.

**RETURN VALUE**

Upon successful completion, the *uselocale()* function shall return a handle for the thread-local locale that was in use as the current locale for the calling thread on entry to the function, or `LC_GLOBAL_LOCALE` if no thread-local locale was in use. Otherwise, *uselocale()* shall return `(locale_t)0` and set *errno* to indicate the error.

**ERRORS**

The *uselocale()* function may fail if:

[EINVAL] *newloc* is not a valid locale object and is not `(locale_t)0`.

**EXAMPLES**

None.

**APPLICATION USAGE**

Unlike the *setlocale()* function, the *uselocale()* function does not allow replacing some locale categories only. Applications that need to install a locale which differs only in a few categories must use *newlocale()* to change a locale object equivalent to the currently used locale and install it.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.



70218 **SEE ALSO**70219 *duplocale(), freelocale(), newlocale(), setlocale()*

70220 XBD &lt;locale.h&gt;

70221 **CHANGE HISTORY**

70222 First released in Issue 7.

70223 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0700 [290] and XSH/TC1-2008/0701  
70224 [334] are applied.

## NAME

utime — set file access and modification times

## SYNOPSIS

```
#include <utime.h>

int utime(const char *path, const struct utimbuf *times);
```

## DESCRIPTION

The *utime()* function shall set the access and modification times of the file named by the *path* argument.

If *times* is a null pointer, the access and modification times of the file shall be set to the current time. The effective user ID of the process shall match the owner of the file, or the process has write permission to the file or has appropriate privileges, to use *utime()* in this manner.

If *times* is not a null pointer, *times* shall be interpreted as a pointer to a **utimbuf** structure and the access and modification times shall be set to the values contained in the designated structure. Only a process with the effective user ID equal to the user ID of the file or a process with appropriate privileges may use *utime()* this way.

The **utimbuf** structure is defined in the **<utime.h>** header. The times in the structure **utimbuf** are measured in seconds since the Epoch.

Upon successful completion, the *utime()* function shall mark the last file status change timestamp for update; see **<sys/stat.h>**.

## RETURN VALUE

Upon successful completion, 0 shall be returned. Otherwise, -1 shall be returned and *errno* shall be set to indicate the error, and the file times shall not be affected.

## ERRORS

The *utime()* function shall fail if:

[EACCES] Search permission is denied by a component of the path prefix; or the *times* argument is a null pointer and the effective user ID of the process does not match the owner of the file, the process does not have write permission for the file, and the process does not have appropriate privileges.

[ELOOP] A loop exists in symbolic links encountered during resolution of the *path* argument.

[ENAMETOOLONG] The length of a component of a pathname is longer than {NAME\_MAX}.

[ENOENT] A component of *path* does not name an existing file or *path* is an empty string.

[ENOTDIR] A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the *path* argument contains at least one non-**<slash>** character and ends with one or more trailing **<slash>** characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.

[EPERM] The *times* argument is not a null pointer and the effective user ID of the calling process does not match the owner of the file and the calling process does not have appropriate privileges.

70266 [EROFS] The file system containing the file is read-only.  
 70267 The *utime()* function may fail if:  
 70268 [ELOOP] More than {SYMLOOP\_MAX} symbolic links were encountered during  
 70269 resolution of the *path* argument.  
 70270 [ENAMETOOLONG]  
 70271 The length of a pathname exceeds {PATH\_MAX}, or pathname resolution of a  
 70272 symbolic link produced an intermediate result with a length that exceeds  
 70273 {PATH\_MAX}.

**EXAMPLES**

70274 None.  
 70275

**APPLICATION USAGE**

70276 Since the **utimbuf** structure only contains **time\_t** variables and is not accurate to fractions of a  
 70277 second, applications should use the *utimensat()* function instead of the obsolescent *utime()*  
 70278 function.  
 70279

**RATIONALE**

70280 The *actime* structure member must be present so that an application may set it, even though an  
 70281 implementation may ignore it and not change the last data access timestamp on the file. If an  
 70282 application intends to leave one of the times of a file unchanged while changing the other, it  
 70283 should use *stat()* or *fstat()* to retrieve the file's *st\_atim* and *st\_mtim* parameters, set *actime* and  
 70284 *modtime* in the buffer, and change one of them before making the *utime()* call.  
 70285

**FUTURE DIRECTIONS**

70286 The *utime()* function may be removed in a future version.  
 70287

**SEE ALSO**

70288 *fstat()*, *fstatat()*, *futimens()*  
 70289 XBD <sys/stat.h>, <utime.h>  
 70290

**CHANGE HISTORY**

70291 First released in Issue 1. Derived from Issue 1 of the SVID.  
 70292

**Issue 6**

70293 The following new requirements on POSIX implementations derive from alignment with the  
 70294 Single UNIX Specification:  
 70295

- 70296 • The requirement to include <sys/types.h> has been removed. Although <sys/types.h> was  
 70297 required for conforming implementations of previous POSIX specifications, it was not  
 70298 required for UNIX applications.
- 70299 • The [ELOOP] mandatory error condition is added.
- 70300 • A second [ENAMETOOLONG] is added as an optional error condition.

70301 The following changes were made to align with the IEEE P1003.1a draft standard:

- 70302 • The [ELOOP] optional error condition is added.

70303 The normative text is updated to avoid use of the term “must” for application requirements.

**Issue 7**

70304 Austin Group Interpretation 1003.1-2001 #143 is applied.  
 70305

70306 The *utime()* function is marked obsolescent.

70307 Changes are made related to support for finegrained timestamps.

70308

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0702 [324] is applied.

70309 **NAME**

70310 utimensat, utimes — set file access and modification times |

70311 **SYNOPSIS**

70312 #include &lt;sys/stat.h&gt;

70313 int utimensat(int *fd*, const char \**path*, const struct timespec *times*[2],  
70314 int *flag*);

70315 XSI #include &lt;sys/time.h&gt;

70316 int utimes(const char \**path*, const struct timeval *times*[2]);70317 **DESCRIPTION**70318 Refer to *futimens()*.

**va\_arg()**70319 **NAME**

70320       va\_arg, va\_copy, va\_end, va\_start — handle variable argument list

70321 **SYNOPSIS**

70322       #include &lt;stdarg.h&gt;

70323       type va\_arg(va\_list ap, type);

70324       void va\_copy(va\_list dest, va\_list src);

70325       void va\_end(va\_list ap);

70326       void va\_start(va\_list ap, argN);

70327 **DESCRIPTION**

70328       Refer to XBD &lt;stdarg.h&gt;

70329 **NAME**70330 `vdprintf`, `vfprintf`, `vprintf`, `vsprintf`, `vsnprintf` — format output of a stdarg argument list70331 **SYNOPSIS**70332 `#include <stdarg.h>`70333 `#include <stdio.h>`

```

70334 CX int vdprintf(int fildes, const char *restrict format, va_list ap);
70335 int vfprintf(FILE *restrict stream, const char *restrict format,
70336 va_list ap);
70337 int vprintf(const char *restrict format, va_list ap);
70338 int vsnprintf(char *restrict s, size_t n, const char *restrict format,
70339 va_list ap);
70340 int vsprintf(char *restrict s, const char *restrict format, va_list ap);

```

70341 **DESCRIPTION**

70342 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 70343 conflict between the requirements described here and the ISO C standard is unintentional. This  
 70344 volume of POSIX.1-2008 defers to the ISO C standard.

70345 CX The `vdprintf()`, `vfprintf()`, `vprintf()`, `vsnprintf()`, and `vsprintf()` functions shall be equivalent to the  
 70346 CX `dprintf()`, `fprintf()`, `printf()`, `snprintf()`, and `sprintf()` functions respectively, except that instead of  
 70347 being called with a variable number of arguments, they are called with an argument list as  
 70348 defined by `<stdarg.h>`.

70349 These functions shall not invoke the `va_end` macro. As these functions invoke the `va_arg` macro,  
 70350 the value of `ap` after the return is unspecified.

70351 **RETURN VALUE**70352 Refer to `fprintf()`.70353 **ERRORS**70354 Refer to `fprintf()`.70355 **EXAMPLES**

70356 None.

70357 **APPLICATION USAGE**70358 Applications using these functions should call `va_end(ap)` afterwards to clean up.70359 **RATIONALE**

70360 None.

70361 **FUTURE DIRECTIONS**

70362 None.

70363 **SEE ALSO**70364 [Section 2.5](#) (on page 495), `fprintf()`70365 XBD `<stdarg.h>`, `<stdio.h>`70366 **CHANGE HISTORY**

70367 First released in Issue 1. Derived from Issue 1 of the SVID.

70368 **Issue 5**70369 The `vsnprintf()` function is added.

70370 **Issue 6**

70371 The *vfprintf()*, *vprintf()*, *vsprintf()*, and *vsprintf()* functions are updated for alignment with the  
70372 ISO/IEC 9899:1999 standard.

70373 **Issue 7**

70374 The *vdprintf()* function is added to complement the *dprintf()* function from The Open Group  
70375 Technical Standard, 2006, Extended API Set Part 1.

70376 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0703 [14] is applied.



**NAME**

vfscanf, vscanf, vsscanf — format input of a stdarg argument list

**SYNOPSIS**

```
#include <stdarg.h>
#include <stdio.h>

int vfscanf(FILE *restrict stream, const char *restrict format,
            va_list arg);
int vscanf(const char *restrict format, va_list arg);
int vsscanf(const char *restrict s, const char *restrict format,
            va_list arg);
```

**DESCRIPTION**

**CX** The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *vscanf()*, *vfscanf()*, and *vsscanf()* functions shall be equivalent to the *scanf()*, *fscanf()*, and *sscanf()* functions, respectively, except that instead of being called with a variable number of arguments, they are called with an argument list as defined in the **<stdarg.h>** header. These functions shall not invoke the *va\_end* macro. As these functions invoke the *va\_arg* macro, the value of *ap* after the return is unspecified.

**RETURN VALUE**

Refer to *fscanf()*.

**ERRORS**

Refer to *fscanf()*.

**EXAMPLES**

None.

**APPLICATION USAGE**

Applications using these functions should call *va\_end(ap)* afterwards to clean up.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

Section 2.5 (on page 495), *fscanf()*

XBD **<stdarg.h>**, **<stdio.h>**

**CHANGE HISTORY**

First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

**Issue 7**

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0704 [14] is applied.

70415 **NAME**

70416 vfwprintf, vswprintf, vwprintf — wide-character formatted output of a stdarg argument list

70417 **SYNOPSIS**

```
70418 #include <stdarg.h>
70419 #include <stdio.h>
70420 #include <wchar.h>

70421 int vfwprintf(FILE *restrict stream, const wchar_t *restrict format,
70422             va_list arg);
70423 int vswprintf(wchar_t *restrict ws, size_t n,
70424             const wchar_t *restrict format, va_list arg);
70425 int vwprintf(const wchar_t *restrict format, va_list arg);
```

70426 **DESCRIPTION**

70427 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
70428 conflict between the requirements described here and the ISO C standard is unintentional. This  
70429 volume of POSIX.1-2008 defers to the ISO C standard.

70430 The *vfwprintf()*, *vswprintf()*, and *vwprintf()* functions shall be equivalent to *fwprintf()*, *swprintf()*,  
70431 and *wprintf()* respectively, except that instead of being called with a variable number of  
70432 arguments, they are called with an argument list as defined by **<stdarg.h>**.

70433 These functions shall not invoke the *va\_end* macro. However, as these functions do invoke the  
70434 *va\_arg* macro, the value of *ap* after the return is unspecified.

70435 **RETURN VALUE**70436 Refer to *fwprintf()*.70437 **ERRORS**70438 Refer to *fwprintf()*.70439 **EXAMPLES**

70440 None.

70441 **APPLICATION USAGE**70442 Applications using these functions should call *va\_end(ap)* afterwards to clean up.70443 **RATIONALE**

70444 None.

70445 **FUTURE DIRECTIONS**

70446 None.

70447 **SEE ALSO**70448 [Section 2.5](#) (on page 495), *fwprintf()*70449 XBD **<stdarg.h>**, **<stdio.h>**, **<wchar.h>**70450 **CHANGE HISTORY**

70451 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
70452 (E).

70453 **Issue 6**

70454 The *vfwprintf()*, *vswprintf()*, and *vwprintf()* prototypes are updated for alignment with the  
70455 ISO/IEC 9899:1999 standard.

70456 **Issue 7**  
70457

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0705 [14] is applied.

**NAME**

vfwscanf, vswscanf, wscanf — wide-character formatted input of a stdarg argument list

**SYNOPSIS**

```
#include <stdarg.h>
#include <stdio.h>
#include <wchar.h>

int vfwscanf(FILE *restrict stream, const wchar_t *restrict format,
             va_list arg);
int vswscanf(const wchar_t *restrict ws, const wchar_t *restrict format,
             va_list arg);
int vwscanf(const wchar_t *restrict format, va_list arg);
```

**DESCRIPTION**

**CX** The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *vfwscanf()*, *vswscanf()*, and *vwscanf()* functions shall be equivalent to the *fwscanf()*, *swscanf()*, and *wscanf()* functions, respectively, except that instead of being called with a variable number of arguments, they are called with an argument list as defined in the **<stdarg.h>** header. These functions shall not invoke the *va\_end* macro. As these functions invoke the *va\_arg* macro, the value of *ap* after the return is unspecified.

**RETURN VALUE**

Refer to *fwscanf()*.

**ERRORS**

Refer to *fwscanf()*.

**EXAMPLES**

None.

**APPLICATION USAGE**

Applications using these functions should call *va\_end(ap)* afterwards to clean up.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 2.5](#) (on page 495), *fwscanf()*

XBD **<stdarg.h>**, **<stdio.h>**, **<wchar.h>**

**CHANGE HISTORY**

First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

**Issue 7**

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0706 [14] is applied.

70497 **NAME**

70498       vprintf — format the output of a stdarg argument list

70499 **SYNOPSIS**

70500       #include &lt;stdarg.h&gt;

70501       #include &lt;stdio.h&gt;

70502       int vprintf(const char \*restrict *format*, va\_list *ap*);70503 **DESCRIPTION**70504       Refer to *vfprintf()*.

70505 **NAME**

70506       vscanf — format input of a stdarg argument list

70507 **SYNOPSIS**

70508       #include &lt;stdarg.h&gt;

70509       #include &lt;stdio.h&gt;

70510       int vscanf(const char \*restrict *format*, va\_list *arg*);70511 **DESCRIPTION**70512       Refer to *vfscanf()*.

70513 **NAME**

70514       vsprintf, vsprintf — format output of a stdarg argument list

70515 **SYNOPSIS**

70516       #include &lt;stdarg.h&gt;

70517       #include &lt;stdio.h&gt;

70518       int vsnprintf(char \*restrict *s*, size\_t *n*,70519           const char \*restrict *format*, va\_list *ap*);70520       int vsprintf(char \*restrict *s*, const char \*restrict *format*,70521           va\_list *ap*);70522 **DESCRIPTION**70523       Refer to *fprintf()*.

70524 **NAME**

70525       vsscanf — format input of a stdarg argument list

70526 **SYNOPSIS**

70527       #include &lt;stdarg.h&gt;

70528       #include &lt;stdio.h&gt;

70529       int vsscanf(const char \*restrict *s*, const char \*restrict *format*,70530               va\_list *arg*);70531 **DESCRIPTION**70532       Refer to *vfscanf()*.



70533 **NAME**

70534 vswprintf — wide-character formatted output of a stdarg argument list

70535 **SYNOPSIS**

70536 #include &lt;stdarg.h&gt;

70537 #include &lt;stdio.h&gt;

70538 #include &lt;wchar.h&gt;

```
70539 int vswprintf(wchar_t *restrict ws, size_t n,  
70540             const wchar_t *restrict format, va_list arg);
```

70541 **DESCRIPTION**70542 Refer to *vfwprintf()*.

70543 **NAME**

70544 vswscanf — wide-character formatted input of a stdarg argument list

70545 **SYNOPSIS**

70546 #include &lt;stdarg.h&gt;

70547 #include &lt;stdio.h&gt;

70548 #include &lt;wchar.h&gt;

```
70549 int vswscanf(const wchar_t *restrict ws, const wchar_t *restrict format,  
70550             va_list arg);
```

70551 **DESCRIPTION**70552 Refer to *vfwscanf()*.

70553 **NAME**

70554       vwprintf — wide-character formatted output of a stdarg argument list

70555 **SYNOPSIS**

70556       #include &lt;stdarg.h&gt;

70557       #include &lt;stdio.h&gt;

70558       #include &lt;wchar.h&gt;

70559       int vwprintf(const wchar\_t \*restrict *format*, va\_list *arg*);70560 **DESCRIPTION**70561       Refer to *vwprintf()*.

70562 **NAME**

70563       vwscanf — wide-character formatted input of a stdarg argument list

70564 **SYNOPSIS**

70565       #include &lt;stdarg.h&gt;

70566       #include &lt;stdio.h&gt;

70567       #include &lt;wchar.h&gt;

70568       int vwscanf(const wchar\_t \*restrict *format*, va\_list *arg*);70569 **DESCRIPTION**70570       Refer to *vwscanf()*.

**NAME**

wait, waitpid — wait for a child process to stop or terminate

**SYNOPSIS**

```
#include <sys/wait.h>
```

```
pid_t wait(int *stat_loc);
```

```
pid_t waitpid(pid_t pid, int *stat_loc, int options);
```

**DESCRIPTION**

The *wait()* and *waitpid()* functions shall obtain status information (see [Section 2.13](#), on page 548) pertaining to one of the caller's child processes. The *wait()* function obtains status information for process termination from any child process. The *waitpid()* function obtains status information for process termination, and optionally process stop and/or continue, from a specified subset of the child processes.

The *wait()* function shall cause the calling thread to become blocked until status information generated by child process termination is made available to the thread, or until delivery of a signal whose action is either to execute a signal-catching function or to terminate the process, or an error occurs. If termination status information is available prior to the call to *wait()*, return shall be immediate. If termination status information is available for two or more child processes, the order in which their status is reported is unspecified.

As described in [Section 2.13](#) (on page 548), the *wait()* and *waitpid()* functions consume the status information they obtain.

The behavior when multiple threads are blocked in *wait()*, *waitid()*, or *waitpid()* is described in [Section 2.13](#) (on page 548).

The *waitpid()* function shall be equivalent to *wait()* if the *pid* argument is **(pid\_t)−1** and the *options* argument is 0. Otherwise, its behavior shall be modified by the values of the *pid* and *options* arguments.

The *pid* argument specifies a set of child processes for which *status* is requested. The *waitpid()* function shall only return the status of a child process from this set:

- If *pid* is equal to **(pid\_t)−1**, *status* is requested for any child process. In this respect, *waitpid()* is then equivalent to *wait()*.
- If *pid* is greater than 0, it specifies the process ID of a single child process for which *status* is requested.
- If *pid* is 0, *status* is requested for any child process whose process group ID is equal to that of the calling process.
- If *pid* is less than **(pid\_t)−1**, *status* is requested for any child process whose process group ID is equal to the absolute value of *pid*.

The *options* argument is constructed from the bitwise-inclusive OR of zero or more of the following flags, defined in the **<sys/wait.h>** header:

XSI **WCONTINUED** The *waitpid()* function shall report the status of any continued child process specified by *pid* whose status has not been reported since it continued from a job control stop.

**WNOHANG** The *waitpid()* function shall not suspend execution of the calling thread if *status* is not immediately available for one of the child processes specified by *pid*.

70614 WUNTRACED The status of any child processes specified by *pid* that are stopped, and whose  
 70615 status has not yet been reported since they stopped, shall also be reported to  
 70616 the requesting process.

70617 If *wait()* or *waitpid()* return because the status of a child process is available, these functions -  
 70618 shall return a value equal to the process ID of the child process. In this case, if the value of the  
 70619 argument *stat\_loc* is not a null pointer, information shall be stored in the location pointed to by  
 70620 *stat\_loc*. The value stored at the location pointed to by *stat\_loc* shall be 0 if and only if the status  
 70621 returned is from a terminated child process that terminated by one of the following means:

- 70622 1. The process returned 0 from *main()*.
- 70623 2. The process called *\_exit()* or *exit()* with a *status* argument of 0.
- 70624 3. The process was terminated because the last thread in the process terminated.

70625 Regardless of its value, this information may be interpreted using the following macros, which  
 70626 are defined in **<sys/wait.h>** and evaluate to integral expressions; the *stat\_val* argument is the  
 70627 integer value pointed to by *stat\_loc*.

70628 WIFEXITED(*stat\_val*)  
 70629 Evaluates to a non-zero value if *status* was returned for a child process that terminated  
 70630 normally.

70631 WEXITSTATUS(*stat\_val*)  
 70632 If the value of WIFEXITED(*stat\_val*) is non-zero, this macro evaluates to the low-order 8 bits  
 70633 of the *status* argument that the child process passed to *\_exit()* or *exit()*, or the value the child  
 70634 process returned from *main()*.

70635 WIFSIGNALED(*stat\_val*)  
 70636 Evaluates to a non-zero value if *status* was returned for a child process that terminated due  
 70637 to the receipt of a signal that was not caught (see **<signal.h>**).

70638 WTERMSIG(*stat\_val*)  
 70639 If the value of WIFSIGNALED(*stat\_val*) is non-zero, this macro evaluates to the number of  
 70640 the signal that caused the termination of the child process.

70641 WIFSTOPPED(*stat\_val*)  
 70642 Evaluates to a non-zero value if *status* was returned for a child process that is currently  
 70643 stopped.

70644 WSTOPSIG(*stat\_val*)  
 70645 If the value of WIFSTOPPED(*stat\_val*) is non-zero, this macro evaluates to the number of the  
 70646 signal that caused the child process to stop.

70647 XSI WIFCONTINUED(*stat\_val*)  
 70648 Evaluates to a non-zero value if *status* was returned for a child process that has continued  
 70649 from a job control stop.

70650 SPN It is unspecified whether the *status* value returned by calls to *wait()* or *waitpid()* for processes  
 70651 created by *posix\_spawn()* or *posix\_spawnnp()* can indicate a WIFSTOPPED(*stat\_val*) before  
 70652 subsequent calls to *wait()* or *waitpid()* indicate WIFEXITED(*stat\_val*) as the result of an error  
 70653 detected before the new process image starts executing.

70654 It is unspecified whether the *status* value returned by calls to *wait()* or *waitpid()* for processes  
 70655 created by *posix\_spawn()* or *posix\_spawnnp()* can indicate a WIFSIGNALED(*stat\_val*) if a signal is  
 70656 sent to the parent's process group after *posix\_spawn()* or *posix\_spawnnp()* is called.

70657 If the information pointed to by *stat\_loc* was stored by a call to *waitpid()* that specified the

70658 XSI WUNTRACED flag and did not specify the WCONTINUED flag, exactly one of the macros |  
 70659 WIFEXITED(\*stat\_loc), WIFSIGNALED(\*stat\_loc), and WIFSTOPPED(\*stat\_loc) shall evaluate to a |  
 70660 non-zero value.

70661 XSI If the information pointed to by *stat\_loc* was stored by a call to *waitpid()* that specified the |  
 70662 WUNTRACED and WCONTINUED flags, exactly one of the macros WIFEXITED(\*stat\_loc), |  
 70663 WIFSIGNALED(\*stat\_loc), WIFSTOPPED(\*stat\_loc), and WIFCONTINUED(\*stat\_loc) shall |  
 70664 evaluate to a non-zero value. -

70665 If the information pointed to by *stat\_loc* was stored by a call to *waitpid()* that did not specify the |  
 70666 XSI WUNTRACED or WCONTINUED flags, or by a call to the *wait()* function, exactly one of the |  
 70667 macros WIFEXITED(\*stat\_loc) and WIFSIGNALED(\*stat\_loc) shall evaluate to a non-zero value.

70668 XSI If the information pointed to by *stat\_loc* was stored by a call to *waitpid()* that did not specify the |  
 70669 WUNTRACED flag and specified the WCONTINUED flag, exactly one of the macros |  
 70670 WIFEXITED(\*stat\_loc), WIFSIGNALED(\*stat\_loc), and WIFCONTINUED(\*stat\_loc) shall evaluate |  
 70671 to a non-zero value.

70672 If `_POSIX_REALTIME_SIGNALS` is defined, and the implementation queues the SIGCHLD |  
 70673 signal, then if *wait()* or *waitpid()* returns because the status of a child process is available, any |  
 70674 pending SIGCHLD signal associated with the process ID of the child process shall be discarded. |  
 70675 Any other pending SIGCHLD signals shall remain pending.

70676 Otherwise, if SIGCHLD is blocked, if *wait()* or *waitpid()* return because the status of a child |  
 70677 process is available, any pending SIGCHLD signal shall be cleared unless the status of another |  
 70678 child process is available.

70679 For all other conditions, it is unspecified whether child *status* will be available when a SIGCHLD |  
 70680 signal is delivered.

70681 There may be additional implementation-defined circumstances under which *wait()* or *waitpid()* |  
 70682 report *status*. This shall not occur unless the calling process or one of its child processes |  
 70683 explicitly makes use of a non-standard extension. In these cases the interpretation of the |  
 70684 reported *status* is implementation-defined.

70685 If a parent process terminates without waiting for all of its child processes to terminate, the |  
 70686 remaining child processes shall be assigned a new parent process ID corresponding to an |  
 70687 implementation-defined system process.

## 70688 RETURN VALUE

70689 If *wait()* or *waitpid()* returns because the status of a child process is available, these functions |  
 70690 shall return a value equal to the process ID of the child process for which *status* is reported. If |  
 70691 *wait()* or *waitpid()* returns due to the delivery of a signal to the calling process, -1 shall be |  
 70692 returned and *errno* set to [EINTR]. If *waitpid()* was invoked with WNOHANG set in *options*, it |  
 70693 has at least one child process specified by *pid* for which *status* is not available, and *status* is not |  
 70694 available for any process specified by *pid*, 0 is returned. Otherwise, -1 shall be returned, and |  
 70695 *errno* set to indicate the error.

## 70696 ERRORS

70697 The *wait()* function shall fail if:

70698 [ECHILD] The calling process has no existing unwaited-for child processes.

70699 [EINTR] The function was interrupted by a signal. The value of the location pointed to |  
 70700 by *stat\_loc* is undefined.

70701 The *waitpid()* function shall fail if:

|       |          |                                                                                                                                                                                                                                      |
|-------|----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 70702 | [ECHILD] | The process specified by <i>pid</i> does not exist or is not a child of the calling process, or the process group specified by <i>pid</i> does not exist or does not have any member process that is a child of the calling process. |
| 70703 |          |                                                                                                                                                                                                                                      |
| 70704 |          |                                                                                                                                                                                                                                      |
| 70705 | [EINTR]  | The function was interrupted by a signal. The value of the location pointed to by <i>stat_loc</i> is undefined.                                                                                                                      |
| 70706 |          |                                                                                                                                                                                                                                      |
| 70707 | [EINVAL] | The <i>options</i> argument is not valid.                                                                                                                                                                                            |

## 70708 EXAMPLES

### 70709 Waiting for a Child Process and then Checking its Status

70710 The following example demonstrates the use of *waitpid()*, *fork()*, and the macros used to  
 70711 interpret the status value returned by *waitpid()* (and *wait()*). The code segment creates a child  
 70712 process which does some unspecified work. Meanwhile the parent loops performing calls to  
 70713 *waitpid()* to monitor the status of the child. The loop terminates when child termination is  
 70714 detected.

```

70715 #include <stdio.h>
70716 #include <stdlib.h>
70717 #include <unistd.h>
70718 #include <sys/wait.h>
70719 ...
70720 pid_t child_pid, wpid;
70721 int status;
70722
70723 child_pid = fork();
70724 if (child_pid == -1) {          /* fork() failed */
70725     perror("fork");
70726     exit(EXIT_FAILURE);
70727 }
70728 if (child_pid == 0) {          /* This is the child */
70729     /* Child does some work and then terminates */
70730     ...
70731 } else {                        /* This is the parent */
70732     do {
70733         wpid = waitpid(child_pid, &status, WUNTRACED
70734 #ifdef WCONTINUED          /* Not all implementations support this */
70735         | WCONTINUED
70736 #endif
70737         );
70738         if (wpid == -1) {
70739             perror("waitpid");
70740             exit(EXIT_FAILURE);
70741         }
70742         if (WIFEXITED(status)) {
70743             printf("child exited, status=%d\n", WEXITSTATUS(status));
70744         } else if (WIFSIGNALED(status)) {
70745             printf("child killed (signal %d)\n", WTERMSIG(status));
70746         } else if (WIFSTOPPED(status)) {
70747             printf("child stopped (signal %d)\n", WSTOPSIG(status));

```



```

70747 #ifndef WIFCONTINUED      /* Not all implementations support this */
70748     } else if (WIFCONTINUED(status)) {
70749         printf("child continued\n");
70750 #endif
70751     } else {      /* Non-standard case -- may never happen */
70752         printf("Unexpected status (0x%x)\n", status);
70753     }
70754     } while (!WIFEXITED(status) && !WIFSIGNALED(status));
70755 }

```

### Waiting for a Child Process in a Signal Handler for SIGCHLD

The following example demonstrates how to use *waitpid()* in a signal handler for SIGCHLD without passing -1 as the *pid* argument. (See the APPLICATION USAGE section below for the reasons why passing a *pid* of -1 is not recommended.) The method used here relies on the standard behavior of *waitpid()* when SIGCHLD is blocked. On historical non-conforming systems, the status of some child processes might not be reported.

```

70762 #include <stdlib.h>
70763 #include <stdio.h>
70764 #include <signal.h>
70765 #include <sys/types.h>
70766 #include <sys/wait.h>
70767 #include <unistd.h>
70768
70769 #define CHILDREN 10
70770
70771 static void
70772 handle_sigchld(int signum, siginfo_t *sinfo, void *unused)
70773 {
70774     int sav_errno = errno;
70775     int status;
70776
70777     /*
70778      * Obtain status information for the child which
70779      * caused the SIGCHLD signal and write its exit code
70780      * to stdout.
70781      */
70782     if (sinfo->si_code != CLD_EXITED)
70783     {
70784         static char msg[] = "wrong si_code\n";
70785         write(2, msg, sizeof msg - 1);
70786     }
70787     else if (waitpid(sinfo->si_pid, &status, 0) == -1)
70788     {
70789         static char msg[] = "waitpid() failed\n";
70790         write(2, msg, sizeof msg - 1);
70791     }
70792     else if (!WIFEXITED(status))
70793     {
70794         static char msg[] = "WIFEXITED was false\n";
70795         write(2, msg, sizeof msg - 1);
70796     }
70797     else

```

```

70795     {
70796         int code = WEXITSTATUS(status);
70797         char buf[2];
70798         buf[0] = '0' + code;
70799         buf[1] = '\n';
70800         write(1, buf, 2);
70801     }
70802     errno = sav_errno;
70803 }
70804
70805 int
70806 main(void)
70807 {
70808     int i;
70809     pid_t pid;
70810     struct sigaction sa;
70811
70812     sa.sa_flags = SA_SIGINFO;
70813     sa.sa_sigaction = handle_sigchld;
70814     sigemptyset(&sa.sa_mask);
70815     if (sigaction(SIGCHLD, &sa, NULL) == -1)
70816     {
70817         perror("sigaction");
70818         exit(EXIT_FAILURE);
70819     }
70820
70821     for (i = 0; i < CHILDREN; i++)
70822     {
70823         switch (pid = fork())
70824         {
70825             case -1:
70826                 perror("fork");
70827                 exit(EXIT_FAILURE);
70828             case 0:
70829                 sleep(2);
70830                 _exit(i);
70831         }
70832     }
70833
70834     /* Wait for all the SIGCHLD signals, then terminate on SIGALRM */
70835     alarm(3);
70836     for (;;)
70837         pause();
70838
70839     return 0; /* NOTREACHED */
70840 }

```

## APPLICATION USAGE

Calls to *wait()* will collect information about any child process. This may result in interactions with other interfaces that may be waiting for their own children (such as by use of *system()*). For this and other reasons it is recommended that portable applications not use *wait()*, but instead use *waitpid()*. For these same reasons, the use of *waitpid()* with a *pid* argument of -1, and the use of *waitid()* with the *idtype* argument set to P\_ALL, are also not recommended for portable applications.

70843 XSI As specified in [Consequences of Process Termination](#) (on page 553), if the calling process has +  
 70844 SA\_NOCLDWAIT set or has SIGCHLD set to SIG\_IGN, then the termination of a child process +  
 70845 will not cause status information to become available to a thread blocked in *wait()*, *waitid()*, or +  
 70846 *waitpid()*. Thus, a thread blocked in one of the wait functions will remain blocked unless some +  
 70847 other condition causes the thread to resume execution (such as an [ECHILD] failure due to no +  
 70848 remaining children in the set of waited-for children).

## 70849 RATIONALE

70850 A call to the *wait()* or *waitpid()* function only returns *status* on an immediate child process of the  
 70851 calling process; that is, a child that was produced by a single *fork()* call (perhaps followed by an  
 70852 *exec* or other function calls) from the parent. If a child produces grandchildren by further use of  
 70853 *fork()*, none of those grandchildren nor any of their descendants affect the behavior of a *wait()*  
 70854 from the original parent process. Nothing in this volume of POSIX.1-2008 prevents an  
 70855 implementation from providing extensions that permit a process to get *status* from a grandchild  
 70856 or any other process, but a process that does not use such extensions must be guaranteed to see  
 70857 *status* from only its direct children.

70858 The *waitpid()* function is provided for three reasons:

- 70859 1. To support job control
- 70860 2. To permit a non-blocking version of the *wait()* function
- 70861 3. To permit a library routine, such as *system()* or *pclose()*, to wait for its children without  
 70862 interfering with other terminated children for which the process has not waited

70863 The first two of these facilities are based on the *wait3()* function provided by 4.3 BSD. The  
 70864 function uses the *options* argument, which is equivalent to an argument to *wait3()*. The  
 70865 WUNTRACED flag is used only in conjunction with job control on systems supporting job  
 70866 control. Its name comes from 4.3 BSD and refers to the fact that there are two types of stopped  
 70867 processes in that implementation: processes being traced via the *ptrace()* debugging facility and  
 70868 (untraced) processes stopped by job control signals. Since *ptrace()* is not part of this volume of  
 70869 POSIX.1-2008, only the second type is relevant. The name WUNTRACED was retained because  
 70870 its usage is the same, even though the name is not intuitively meaningful in this context.

70871 The third reason for the *waitpid()* function is to permit independent sections of a process to  
 70872 spawn and wait for children without interfering with each other. For example, the following  
 70873 problem occurs in developing a portable shell, or command interpreter:

```
70874 stream = popen("/bin/true");
70875 (void) system("sleep 100");
70876 (void) pclose(stream);
```

70877 On all historical implementations, the final *pclose()* fails to reap the *wait()* *status* of the *popen()*.

70878 The status values are retrieved by macros, rather than given as specific bit encodings as they are  
 70879 in most historical implementations (and thus expected by existing programs). This was  
 70880 necessary to eliminate a limitation on the number of signals an implementation can support that  
 70881 was inherent in the traditional encodings. This volume of POSIX.1-2008 does require that a *status*  
 70882 value of zero corresponds to a process calling *\_exit(0)*, as this is the most common encoding  
 70883 expected by existing programs. Some of the macro names were adopted from 4.3 BSD.

70884 These macros syntactically operate on an arbitrary integer value. The behavior is undefined  
 70885 unless that value is one stored by a successful call to *wait()* or *waitpid()* in the location pointed to  
 70886 by the *stat\_loc* argument. An early proposal attempted to make this clearer by specifying each  
 70887 argument as *\*stat\_loc* rather than *stat\_val*. However, that did not follow the conventions of other  
 70888 specifications in this volume of POSIX.1-2008 or traditional usage. It also could have implied

that the argument to the macro must literally be *\*stat\_loc*; in fact, that value can be stored or passed as an argument to other functions before being interpreted by these macros.

The extension that affects *wait()* and *waitpid()* and is common in historical implementations is the *ptrace()* function. It is called by a child process and causes that child to stop and return a *status* that appears identical to the *status* indicated by WIFSTOPPED. The *status* of *ptrace()* children is traditionally returned regardless of the WUNTRACED flag (or by the *wait()* function). Most applications do not need to concern themselves with such extensions because they have control over what extensions they or their children use. However, applications, such as command interpreters, that invoke arbitrary processes may see this behavior when those arbitrary processes misuse such extensions.

Implementations that support **core** file creation or other implementation-defined actions on termination of some processes traditionally provide a bit in the *status* returned by *wait()* to indicate that such actions have occurred.

Allowing the *wait()* family of functions to discard a pending SIGCHLD signal that is associated with a successfully waited-for child process puts them into the *sigwait()* and *sigwaitinfo()* category with respect to SIGCHLD.

This definition allows implementations to treat a pending SIGCHLD signal as accepted by the process in *wait()*, with the same meaning of “accepted” as when that word is applied to the *sigwait()* family of functions.

Allowing the *wait()* family of functions to behave this way permits an implementation to be able to deal precisely with SIGCHLD signals.

In particular, an implementation that does accept (discard) the SIGCHLD signal can make the following guarantees regardless of the queuing depth of signals in general (the list of waitable children can hold the SIGCHLD queue):

1. If a SIGCHLD signal handler is established via *sigaction()* without the SA\_RESETHAND flag, SIGCHLD signals can be accurately counted; that is, exactly one SIGCHLD signal will be delivered to or accepted by the process for every child process that terminates.
2. A single *wait()* issued from a SIGCHLD signal handler can be guaranteed to return immediately with status information for a child process.
3. When SA\_SIGINFO is requested, the SIGCHLD signal handler can be guaranteed to receive a non-null pointer to a **siginfo\_t** structure that describes a child process for which a wait via *waitpid()* or *waitid()* will not block or fail.
4. The *system()* function will not cause the SIGCHLD handler of a process to be called as a result of the *fork()/exec* executed within *system()* because *system()* will accept the SIGCHLD signal when it performs a *waitpid()* for its child process. This is a desirable behavior of *system()* so that it can be used in a library without causing side-effects to the application linked with the library.

An implementation that does not permit the *wait()* family of functions to accept (discard) a pending SIGCHLD signal associated with a successfully waited-for child, cannot make the guarantees described above for the following reasons:

#### Guarantee #1

Although it might be assumed that reliable queuing of all SIGCHLD signals generated by the system can make this guarantee, the counter-example is the case of a process that blocks SIGCHLD and performs an indefinite loop of *fork()/wait()* operations. If the implementation supports queued signals, then eventually the system will run out of memory for the queue. The guarantee cannot be made because there must be some limit to

the depth of queuing.

#### Guarantees #2 and #3

These cannot be guaranteed unless the *wait()* family of functions accepts the SIGCHLD signal. Otherwise, a *fork()/wait()* executed while SIGCHLD is blocked (as in the *system()* function) will result in an invocation of the handler when SIGCHLD is unblocked, after the process has disappeared.

#### Guarantee #4

Although possible to make this guarantee, *system()* would have to set the SIGCHLD handler to SIG\_DFL so that the SIGCHLD signal generated by its *fork()* would be discarded (the SIGCHLD default action is to be ignored), then restore it to its previous setting. This would have the undesirable side-effect of discarding all SIGCHLD signals pending to the process.

### FUTURE DIRECTIONS

None.

### SEE ALSO

Section 2.13 (on page 548), *exec*, *exit()*, *fork()*, *system()*, *waitid()*

XBD Section 4.12 (on page 111), *<signal.h>*, *<sys/wait.h>*

### CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.

#### Issue 5

The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

#### Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The requirement to include *<sys/types.h>* has been removed. Although *<sys/types.h>* was required for conforming implementations of previous POSIX specifications, it was not required for UNIX applications.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The processing of the SIGCHLD signal and the [ECHILD] error is clarified.

The semantics of *WIFSTOPPED(stat\_val)*, *WIFEXITED(stat\_val)*, and *WIFSIGNALED(stat\_val)* are defined with respect to *posix\_spawn()* or *posix\_spawnnp()* for alignment with IEEE Std 1003.1d-1999.

The DESCRIPTION is updated for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/145 is applied, adding the example to the EXAMPLES section.

#### Issue 7

SD5-XSH-ERN-202 is applied.

APPLICATION USAGE is added, recommending that the *wait()* function not be used.

An additional example for *waitpid()* is added.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0707 [421], XSH/TC1-2008/0708 [166], XSH/TC1-2008/0709 [166], and XSH/TC1-2008/0710 [69] are applied.

**NAME**

waitid — wait for a child process to change state

**SYNOPSIS**

```
#include <sys/wait.h>
```

```
int waitid(idtype_t idtype, id_t id, siginfo_t *infop, int options);
```

**DESCRIPTION**

The *waitid()* function shall obtain status information (see [Section 2.13](#), on page 548) pertaining to termination, stop, and/or continue events in one of the caller’s child processes.

The *waitid()* function shall cause the calling thread to become blocked until an error occurs or status information becomes available to the calling thread that satisfies all of the following properties (“matching status information”):

- The status information is from one of the child processes in the set of child processes specified by the *idtype* and *id* arguments.
- The state change in the status information matches one of the state change flags set in the *options* argument.

If matching status information is available prior to the call to *waitid()*, return shall be immediate. If matching status information is available for two or more child processes, the order in which their status is reported is unspecified.

As described in [Section 2.13](#) (on page 548), the *waitid()* function consumes the status information it obtains unless the WNOWAIT flag is set in the *options* argument.

The behavior when multiple threads are blocked in *wait()*, *waitid()*, or *waitpid()* is described in [Section 2.13](#) (on page 548).

The *waitid()* function shall record the obtained status information in the structure pointed to by *infop*. The fields of the structure pointed to by *infop* shall be filled in as described under “Pointer to a Function” in [Section 2.4.3](#) (on page 490).

The *idtype* and *id* arguments are used to specify which children *waitid()* waits for.

If *idtype* is P\_PID, *waitid()* shall wait for the child with a process ID equal to (**pid\_t**)*id*.

If *idtype* is P\_PGID, *waitid()* shall wait for any child with a process group ID equal to (**pid\_t**)*id*.

If *idtype* is P\_ALL, *waitid()* shall wait for any children and *id* is ignored.

The *options* argument is used to specify which state changes *waitid()* shall wait for. It is formed by OR’ing together the following flags:

WCONTINUED Status shall be returned for any continued child process whose status either has not been reported since it continued from a job control stop or has been reported only by calls to *waitid()* with the WNOWAIT flag set.

WEXITED Wait for processes that have exited.

WNOHANG Do not hang if no status is available; return immediately.

WNOWAIT Keep the process whose status is returned in *infop* in a waitable state. This shall not affect the state of the process; the process may be waited for again after this call completes.

WSTOPPED Status shall be returned for any child that has stopped upon receipt of a signal, and whose status either has not been reported since it stopped or has been reported only by calls to *waitid()* with the WNOWAIT flag set.



Applications shall specify at least one of the flags WEXITED, WSTOPPED, or WCONTINUED to be OR'ed in with the *options* argument.

The application shall ensure that the *infop* argument points to a **siginfo\_t** structure. If *waitid()* returns because a child process was found that satisfied the conditions indicated by the arguments *idtype* and *options*, then the structure pointed to by *infop* shall be filled in by the system with the status of the process; the *si\_signo* member shall be set equal to SIGCHLD. If *waitid()* returns because WNOHANG was specified and status is not available for any process specified by *idtype* and *id*, then the *si\_signo* and *si\_pid* members of the structure pointed to by *infop* shall be set to zero and the values of other members of the structure are unspecified.

#### 71027 RETURN VALUE

If WNOHANG was specified and status is not available for any process specified by *idtype* and *id*, 0 shall be returned. If *waitid()* returns due to the change of state of one of its children, 0 shall be returned. Otherwise, -1 shall be returned and *errno* set to indicate the error.

#### 71031 ERRORS

The *waitid()* function shall fail if:

|       |          |                                                                                                                         |
|-------|----------|-------------------------------------------------------------------------------------------------------------------------|
| 71033 | [ECHILD] | The calling process has no existing unwaited-for child processes.                                                       |
| 71034 | [EINTR]  | The <i>waitid()</i> function was interrupted by a signal.                                                               |
| 71035 | [EINVAL] | An invalid value was specified for <i>options</i> , or <i>idtype</i> and <i>id</i> specify an invalid set of processes. |

#### 71037 EXAMPLES

71038 None.

#### 71039 APPLICATION USAGE

Calls to *waitid()* with *idtype* equal to P\_ALL will collect information about any child process. This may result in interactions with other interfaces that may be waiting for their own children (such as by use of *system()*). For this reason it is recommended that portable applications not use *waitid()* with *idtype* of P\_ALL. See also APPLICATION USAGE for *wait()*.

71044 XSI As specified in [Consequences of Process Termination](#) (on page 553), if the calling process has +  
 71045 SA\_NOCLDWAIT set or has SIGCHLD set to SIG\_IGN, then the termination of a child process +  
 71046 will not cause status information to become available to a thread blocked in *wait()*, *waitid()*, or +  
 71047 *waitpid()*. Thus, a thread blocked in one of the wait functions will remain blocked unless some +  
 71048 other condition causes the thread to resume execution (such as an [ECHILD] failure due to no +  
 71049 remaining children in the set of waited-for children).

#### 71050 RATIONALE

71051 None.

#### 71052 FUTURE DIRECTIONS

71053 None.

#### 71054 SEE ALSO

71055 [Section 2.4.3](#) (on page 490), [Section 2.13](#) (on page 548), *exec*, *exit()*, *wait()*

71056 XBD [<signal.h>](#), [<sys/wait.h>](#)

#### 71057 CHANGE HISTORY

71058 First released in Issue 4, Version 2.

71059 **Issue 5**

71060 Moved from X/OPEN UNIX extension to BASE.

71061 The DESCRIPTION is updated for alignment with the POSIX Threads Extension.

71062 **Issue 6**

71063 The normative text is updated to avoid use of the term “must” for application requirements.

71064 **Issue 7**

71065 Austin Group Interpretation 1003.1-2001 #060 is applied, updating the DESCRIPTION.

71066 The *waitid()* function is moved from the XSI option to the Base.

71067 APPLICATION USAGE is added, recommending that the *waitid()* function not be used with  
71068 *idtype* equal to P\_ALL.

71069 The description of the WNOHANG flag is updated.

71070 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0711 [154], XSH/TC1-2008/0712 [154],  
71071 and XSH/TC1-2008/0713 [153] are applied.



71072 **NAME**

71073       waitpid — wait for a child process to stop or terminate

71074 **SYNOPSIS**

71075       #include &lt;sys/wait.h&gt;

71076       pid\_t waitpid(pid\_t pid, int \*stat\_loc, int options);

71077 **DESCRIPTION**71078       Refer to *wait()*.

71079 **NAME**

71080       wcpcpy — copy a wide-character string, returning a pointer to its end

71081 **SYNOPSIS**

```
71082 CX      #include <wchar.h>
71083          wchar_t *wcpcpy(wchar_t *restrict ws1, const wchar_t *restrict ws2);
```

71084 **DESCRIPTION**71085       Refer to *wcscpy()*.

71086 **NAME**

71087       wcpncpy — copy a fixed-size wide-character string, returning a pointer to its end

71088 **SYNOPSIS**

```
71089 CX      #include <wchar.h>
71090          wchar_t *wcpncpy(wchar_t restrict *ws1, const wchar_t *restrict ws2,
71091                          size_t n);
```

71092 **DESCRIPTION**71093       Refer to *wcsncpy()*.

71094 **NAME**71095 `wrtomb` — convert a wide-character code to a character (restartable)71096 **SYNOPSIS**71097 `#include <wchar.h>`71098 `size_t wrtomb(char *restrict s, wchar_t wc, mbstate_t *restrict ps);`71099 **DESCRIPTION**

71100 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 71101 conflict between the requirements described here and the ISO C standard is unintentional. This  
 71102 volume of POSIX.1-2008 defers to the ISO C standard.

71103 If *s* is a null pointer, the `wrtomb()` function shall be equivalent to the call:71104 `wrtomb(buf, L'\0', ps)`71105 where *buf* is an internal buffer.

71106 If *s* is not a null pointer, the `wrtomb()` function shall determine the number of bytes needed to  
 71107 represent the character that corresponds to the wide character given by *wc* (including any shift  
 71108 sequences), and store the resulting bytes in the array whose first element is pointed to by *s*. At  
 71109 most {MB\_CUR\_MAX} bytes are stored. If *wc* is a null wide character, a null byte shall be stored,  
 71110 preceded by any shift sequence needed to restore the initial shift state. The resulting state  
 71111 described shall be the initial conversion state.

71112 If *ps* is a null pointer, the `wrtomb()` function shall use its own internal **mbstate\_t** object, which is  
 71113 initialized at program start-up to the initial conversion state. Otherwise, the **mbstate\_t** object  
 71114 pointed to by *ps* shall be used to completely describe the current conversion state of the  
 71115 associated character sequence. The implementation shall behave as if no function defined in this  
 71116 volume of POSIX.1-2008 calls `wrtomb()`.

71117 CX The `wrtomb()` function need not be thread-safe if called with a NULL *ps* argument.71118 The behavior of this function shall be affected by the `LC_CTYPE` category of the current locale.71119 The `wrtomb()` function shall not change the setting of *errno* if successful.71120 **RETURN VALUE**

71121 The `wrtomb()` function shall return the number of bytes stored in the array object (including any  
 71122 shift sequences). When *wc* is not a valid wide character, an encoding error shall occur. In this  
 71123 case, the function shall store the value of the macro [EILSEQ] in *errno* and shall return (**size\_t**)−1;  
 71124 the conversion state shall be undefined.

71125 **ERRORS**71126 The `wrtomb()` function shall fail if:

71127 [EILSEQ] An invalid wide-character code is detected.

71128 The `wrtomb()` function may fail if:71129 CX [EINVAL] *ps* points to an object that contains an invalid conversion state.

71130 **EXAMPLES**

71131 None.

71132 **APPLICATION USAGE**

71133 None.

71134 **RATIONALE**

71135 None.

71136 **FUTURE DIRECTIONS**

71137 None.

71138 **SEE ALSO**71139 *mbsinit()*, *wcsrtombs()*

71140 XBD &lt;wchar.h&gt;

71141 **CHANGE HISTORY**71142 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
71143 (E).71144 **Issue 6**

71145 In the DESCRIPTION, a note on using this function in a threaded application is added.

71146 Extensions beyond the ISO C standard are marked.

71147 The normative text is updated to avoid use of the term “must” for application requirements.

71148 The *wcr tomb()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.71149 **Issue 7**71150 Austin Group Interpretation 1003.1-2001 #148 is applied, clarifying that the *wcr tomb()* function  
71151 need not be thread-safe if called with a NULL *ps* argument.

71152 Austin Group Interpretation 1003.1-2001 #170 is applied.

71153 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0714 [88] and XSH/TC1-2008/0715  
71154 [105] are applied.

## 71155 NAME

71156 `wcscasecmp`, `wcscasecmp_l`, `wcsncasecmp`, `wcsncasecmp_l` — case-insensitive wide-character  
71157 string comparison

## 71158 SYNOPSIS

```
71159 CX #include <wchar.h>
71160
71161 int wcscasecmp(const wchar_t *ws1, const wchar_t *ws2);
71162 int wcscasecmp_l(const wchar_t *ws1, const wchar_t *ws2,
71163                 locale_t locale);
71164 int wcsncasecmp(const wchar_t *ws1, const wchar_t *ws2, size_t n);
71165 int wcsncasecmp_l(const wchar_t *ws1, const wchar_t *ws2,
71166                  size_t n, locale_t locale);
```

## 71166 DESCRIPTION

71167 The `wcscasecmp()` and `wcsncasecmp()` functions are the wide-character equivalent of the  
71168 `strcasecmp()` and `strncasecmp()` functions, respectively.

71169 The `wcscasecmp()` and `wcscasecmp_l()` functions shall compare, while ignoring differences in case,  
71170 the wide-character string pointed to by `ws1` to the wide-character string pointed to by `ws2`.

71171 The `wcsncasecmp()` and `wcsncasecmp_l()` functions shall compare, while ignoring differences in  
71172 case, not more than `n` wide-characters from the wide-character string pointed to by `ws1` to the  
71173 wide-character string pointed to by `ws2`.

71174 The `wcscasecmp()` and `wcsncasecmp()` functions use the current locale to determine the case of the  
71175 wide characters.

71176 The `wcscasecmp_l()` and `wcsncasecmp_l()` functions use the locale represented by `locale` to  
71177 determine the case of the wide characters.

71178 When the `LC_CTYPE` category of the locale being used is from the POSIX locale, these functions  
71179 shall behave as if the wide-character strings had been converted to lowercase and then a  
71180 comparison of wide-character codes performed. Otherwise, the results are unspecified.

71181 The information for `wcscasecmp_l()` and `wcsncasecmp_l()` about the case of the characters comes  
71182 from the locale represented by `locale`.

71183 The behavior is undefined if the `locale` argument to `wcscasecmp_l()` or `wcsncasecmp_l()` is the  
71184 special locale object `LC_GLOBAL_LOCALE` or is not a valid locale object handle.

## 71185 RETURN VALUE

71186 Upon completion, the `wcscasecmp()` and `wcscasecmp_l()` functions shall return an integer greater  
71187 than, equal to, or less than 0 if the wide-character string pointed to by `ws1` is, ignoring case,  
71188 greater than, equal to, or less than the wide-character string pointed to by `ws2`, respectively.

71189 Upon completion, the `wcsncasecmp()` and `wcsncasecmp_l()` functions shall return an integer  
71190 greater than, equal to, or less than 0 if the possibly null wide-character terminated string pointed  
71191 to by `ws1` is, ignoring case, greater than, equal to, or less than the possibly null wide-character  
71192 terminated string pointed to by `ws2`, respectively.

71193 No return values are reserved to indicate an error.

## 71194 ERRORS

71195 No errors are defined.

71196 **EXAMPLES**

71197 None.

71198 **APPLICATION USAGE**

71199 None.

71200 **RATIONALE**

71201 None.

71202 **FUTURE DIRECTIONS**

71203 None.

71204 **SEE ALSO**71205 *strcasecmp()*, *wscmp()*, *wcsncmp()*

71206 XBD &lt;wchar.h&gt;

71207 **CHANGE HISTORY**

71208 First released in Issue 7.

71209 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0716 [294], XSH/TC1-2008/0717 [283],  
71210 and XSH/TC1-2008/0718 [283] are applied.

71211 **NAME**

71212 wscat — concatenate two wide-character strings

71213 **SYNOPSIS**

71214 #include &lt;wchar.h&gt;

71215 wchar\_t \*wscat(wchar\_t \*restrict ws1, const wchar\_t \*restrict ws2);

71216 **DESCRIPTION**

71217 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
71218 conflict between the requirements described here and the ISO C standard is unintentional. This  
71219 volume of POSIX.1-2008 defers to the ISO C standard.

71220 The *wscat()* function shall append a copy of the wide-character string pointed to by *ws2*  
71221 (including the terminating null wide-character code) to the end of the wide-character string  
71222 pointed to by *ws1*. The initial wide-character code of *ws2* shall overwrite the null wide-character  
71223 code at the end of *ws1*. If copying takes place between objects that overlap, the behavior is  
71224 undefined.

71225 **RETURN VALUE**71226 The *wscat()* function shall return *ws1*; no return value is reserved to indicate an error.71227 **ERRORS**

71228 No errors are defined.

71229 **EXAMPLES**

71230 None.

71231 **APPLICATION USAGE**

71232 None.

71233 **RATIONALE**

71234 None.

71235 **FUTURE DIRECTIONS**

71236 None.

71237 **SEE ALSO**71238 [wcsncat\(\)](#)71239 XBD [<wchar.h>](#)71240 **CHANGE HISTORY**

71241 First released in Issue 4. Derived from the MSE working draft.

71242 **Issue 6**

71243 The Open Group Corrigendum U040/2 is applied. In the RETURN VALUE section, *s1* is  
71244 changed to *ws1*.

71245 The *wscat()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.



71246 **NAME**71247        `wcschr` — wide-character string scanning operation71248 **SYNOPSIS**71249        `#include <wchar.h>`71250        `wchar_t *wcschr(const wchar_t *ws, wchar_t wc);`71251 **DESCRIPTION**

71252 **CX**        The functionality described on this reference page is aligned with the ISO C standard. Any  
 71253 conflict between the requirements described here and the ISO C standard is unintentional. This  
 71254 volume of POSIX.1-2008 defers to the ISO C standard.

71255        The `wcschr()` function shall locate the first occurrence of `wc` in the wide-character string pointed  
 71256 to by `ws`. The application shall ensure that the value of `wc` is a character representable as a type  
 71257 **wchar\_t** and a wide-character code corresponding to a valid character in the current locale. The  
 71258 terminating null wide-character code is considered to be part of the wide-character string.

71259 **RETURN VALUE**

71260        Upon completion, `wcschr()` shall return a pointer to the wide-character code, or a null pointer if  
 71261 the wide-character code is not found.

71262 **ERRORS**

71263        No errors are defined.

71264 **EXAMPLES**

71265        None.

71266 **APPLICATION USAGE**

71267        None.

71268 **RATIONALE**

71269        None.

71270 **FUTURE DIRECTIONS**

71271        None.

71272 **SEE ALSO**71273        [\*wcsrchr\(\)\*](#)71274        XBD [\*<wchar.h>\*](#)71275 **CHANGE HISTORY**

71276        First released in Issue 4. Derived from the MSE working draft.

71277 **Issue 6**

71278        The normative text is updated to avoid use of the term “must” for application requirements.

71279 **NAME**71280 `wcscmp` — compare two wide-character strings71281 **SYNOPSIS**71282 `#include <wchar.h>`71283 `int wcscmp(const wchar_t *ws1, const wchar_t *ws2);`71284 **DESCRIPTION**

71285 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
71286 conflict between the requirements described here and the ISO C standard is unintentional. This  
71287 volume of POSIX.1-2008 defers to the ISO C standard.

71288 The `wcscmp()` function shall compare the wide-character string pointed to by `ws1` to the wide-  
71289 character string pointed to by `ws2`.

71290 The sign of a non-zero return value shall be determined by the sign of the difference between the  
71291 values of the first pair of wide-character codes that differ in the objects being compared.

71292 **RETURN VALUE**

71293 Upon completion, `wcscmp()` shall return an integer greater than, equal to, or less than 0, if the  
71294 wide-character string pointed to by `ws1` is greater than, equal to, or less than the wide-character  
71295 string pointed to by `ws2`, respectively.

71296 **ERRORS**

71297 No errors are defined.

71298 **EXAMPLES**

71299 None.

71300 **APPLICATION USAGE**

71301 None.

71302 **RATIONALE**

71303 None.

71304 **FUTURE DIRECTIONS**

71305 None.

71306 **SEE ALSO**71307 [`wscasecmp\(\)`](#), [`wcsncmp\(\)`](#)71308 XBD [`<wchar.h>`](#)71309 **CHANGE HISTORY**

71310 First released in Issue 4. Derived from the MSE working draft.

71311 **NAME**

71312 wscoll, wscoll\_l — wide-character string comparison using collating information

71313 **SYNOPSIS**

```
71314 #include <wchar.h>
71315 int wscoll(const wchar_t *ws1, const wchar_t *ws2);
71316 CX int wscoll_l(const wchar_t *ws1, const wchar_t *ws2,
71317 locale_t locale);
```

71318 **DESCRIPTION**

71319 CX For *wscoll()*: The functionality described on this reference page is aligned with the ISO C  
 71320 standard. Any conflict between the requirements described here and the ISO C standard is  
 71321 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

71322 CX The *wscoll()* and *wscoll\_l()* functions shall compare the wide-character string pointed to by  
 71323 *ws1* to the wide-character string pointed to by *ws2*, both interpreted as appropriate to the  
 71324 CX LC\_COLLATE category of the current locale, or the locale represented by *locale*, respectively.

71325 CX The *wscoll()* and *wscoll\_l()* functions shall not change the setting of *errno* if successful.

71326 CX An application wishing to check for error situations should set *errno* to 0 before calling *wscoll()*  
 71327 or *wscoll\_l()*. If *errno* is non-zero on return, an error has occurred.

71328 CX The behavior is undefined if the *locale* argument to *wscoll\_l()* is the special locale object  
 71329 LC\_GLOBAL\_LOCALE or is not a valid locale object handle.

71330 **RETURN VALUE**

71331 CX Upon successful completion, *wscoll()* and *wscoll\_l()* shall return an integer greater than, equal  
 71332 to, or less than 0, according to whether the wide-character string pointed to by *ws1* is greater  
 71333 than, equal to, or less than the wide-character string pointed to by *ws2*, when both are  
 71334 CX interpreted as appropriate to the current locale, or to the locale represented by *locale*,  
 71335 CX respectively. On error, *wscoll()* and *wscoll\_l()* shall set *errno*, but no return value is reserved  
 71336 to indicate an error.

71337 **ERRORS**

71338 These functions may fail if:

71339 CX [EINVAL] The *ws1* or *ws2* arguments contain wide-character codes outside the domain of  
 71340 the collating sequence.

71341 **EXAMPLES**

71342 None.

71343 **APPLICATION USAGE**

71344 The *wcsxfrm()* and *wscmp()* functions should be used for sorting large lists.

71345 **RATIONALE**

71346 None.

71347 **FUTURE DIRECTIONS**

71348 None.

71349 **SEE ALSO**

71350 *wscmp()*, *wcsxfrm()*

71351 XBD <wchar.h>

71352 **CHANGE HISTORY**

71353 First released in Issue 4. Derived from the MSE working draft.

71354 **Issue 5**

71355 Moved from ENHANCED I18N to BASE and the [ENOSYS] error is removed.

71356 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

71357 **Issue 7**

71358 The *wccoll\_l()* function is added from The Open Group Technical Standard, 2006, Extended API  
71359 Set Part 4.

71360 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0719 [302], XSH/TC1-2008/0720 [283],  
71361 and XSH/TC1-2008/0721 [283] are applied.

71362 **NAME**

71363       wcpcpy, wcscpy — copy a wide-character string, returning a pointer to its end

71364 **SYNOPSIS**

71365       #include &lt;wchar.h&gt;

71366 CX       wchar\_t \*wcpcpy(wchar\_t \*restrict ws1, const wchar\_t \*restrict ws2);

71367       wchar\_t \*wcscpy(wchar\_t \*restrict ws1, const wchar\_t \*restrict ws2);

71368 **DESCRIPTION**71369 CX       For *wcscpy()*: The functionality described on this reference page is aligned with the ISO C  
71370 standard. Any conflict between the requirements described here and the ISO C standard is  
71371 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.71372 CX       The *wcpcpy()* and *wcscpy()* functions shall copy the wide-character string pointed to by *ws2*  
71373 (including the terminating null wide-character code) into the array pointed to by *ws1*.71374       The application shall ensure that there is room for at least *wcslen(ws2)+1* wide characters in the  
71375 *ws1* array, and that the *ws2* and *ws1* arrays do not overlap.

71376       If copying takes place between objects that overlap, the behavior is undefined.

71377 **RETURN VALUE**71378 CX       The *wcpcpy()* function shall return a pointer to the terminating null wide-character code copied  
71379 into the *ws1* buffer.71380       The *wcscpy()* function shall return *ws1*.

71381       No return values are reserved to indicate an error.

71382 **ERRORS**

71383       No errors are defined.

71384 **EXAMPLES**

71385       None.

71386 **APPLICATION USAGE**

71387       None.

71388 **RATIONALE**

71389       None.

71390 **FUTURE DIRECTIONS**

71391       None.

71392 **SEE ALSO**71393       *strcpy()*, *wcsdup()*, *wcsncpy()*

71394       XBD &lt;wchar.h&gt;

71395 **CHANGE HISTORY**

71396       First released in Issue 4. Derived from the MSE working draft.

71397 **Issue 6**71398       The *wcscpy()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.71399 **Issue 7**71400       The *wcpcpy()* function is added from The Open Group Technical Standard, 2006, Extended API  
71401 Set Part 1.

71402 **NAME**

71403       wcscspn — get the length of a complementary wide substring

71404 **SYNOPSIS**

71405       #include &lt;wchar.h&gt;

71406       size\_t wcscspn(const wchar\_t \*ws1, const wchar\_t \*ws2);

71407 **DESCRIPTION**

71408 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
71409 conflict between the requirements described here and the ISO C standard is unintentional. This  
71410 volume of POSIX.1-2008 defers to the ISO C standard.

71411       The *wcscspn()* function shall compute the length (in wide characters) of the maximum initial  
71412 segment of the wide-character string pointed to by *ws1* which consists entirely of wide-character  
71413 codes *not* from the wide-character string pointed to by *ws2*.

71414 **RETURN VALUE**

71415       The *wcscspn()* function shall return the length of the initial substring of *ws1*; no return value is  
71416 reserved to indicate an error.

71417 **ERRORS**

71418       No errors are defined.

71419 **EXAMPLES**

71420       None.

71421 **APPLICATION USAGE**

71422       None.

71423 **RATIONALE**

71424       None.

71425 **FUTURE DIRECTIONS**

71426       None.

71427 **SEE ALSO**71428       *wcsspn()*

71429       XBD &lt;wchar.h&gt;

71430 **CHANGE HISTORY**

71431       First released in Issue 4. Derived from the MSE working draft.

71432 **Issue 5**

71433       The RETURN VALUE section is updated to indicate that *wcscspn()* returns the length of *ws1*,  
71434 rather than *ws1* itself.

71435 **NAME**

71436       wcsdup — duplicate a wide-character string

71437 **SYNOPSIS**

```
71438 CX       #include <wchar.h>
71439       wchar_t *wcsdup(const wchar_t *string);
```

71440 **DESCRIPTION**71441       The *wcsdup()* function is the wide-character equivalent of the *strdup()* function.

71442       The *wcsdup()* function shall return a pointer to a new wide-character string, allocated as if by a  
71443       call to *malloc()*, which is the duplicate of the wide-character string *string*. The returned pointer  
71444       can be passed to *free()*. A null pointer is returned if the new wide-character string cannot be  
71445       created.

71446 **RETURN VALUE**

71447       Upon successful completion, the *wcsdup()* function shall return a pointer to the newly allocated  
71448       wide-character string. Otherwise, it shall return a null pointer and set *errno* to indicate the error.

71449 **ERRORS**71450       The *wcsdup()* function shall fail if:

71451       [ENOMEM]       Memory large enough for the duplicate string could not be allocated.

71452 **EXAMPLES**

71453       None.

71454 **APPLICATION USAGE**

71455       For functions that allocate memory as if by *malloc()*, the application should release such memory  
71456       when it is no longer required by a call to *free()*. For *wcsdup()*, this is the return value.

71457 **RATIONALE**

71458       None.

71459 **FUTURE DIRECTIONS**

71460       None.

71461 **SEE ALSO**71462       *free()*, *strdup()*, *wscpy()*

71463       XBD &lt;wchar.h&gt;

71464 **CHANGE HISTORY**

71465       First released in Issue 7.

71466 **NAME**71467 `wcsftime` — convert date and time to a wide-character string71468 **SYNOPSIS**71469 `#include <wchar.h>`

```
71470 size_t wcsftime(wchar_t *restrict wcs, size_t maxsize,  
71471               const wchar_t *restrict format, const struct tm *restrict timeptr);
```

71472 **DESCRIPTION**

71473 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
71474 conflict between the requirements described here and the ISO C standard is unintentional. This  
71475 volume of POSIX.1-2008 defers to the ISO C standard.

71476 The `wcsftime()` function shall be equivalent to the `strftime()` function, except that:

- 71477 • The argument *wcs* points to the initial element of an array of wide characters into which  
71478 the generated output is to be placed.
- 71479 • The argument *maxsize* indicates the maximum number of wide characters to be placed in  
71480 the output array.
- 71481 • The argument *format* is a wide-character string and the conversion specifications are  
71482 replaced by corresponding sequences of wide characters. It is unspecified whether an +  
71483 encoding error occurs if the format string contains **wchar\_t** values that do not correspond +  
71484 to members of the character set of the current locale. +
- 71485 CX • Field widths specify the number of wide characters instead of the number of bytes.
- 71486 • The return value indicates the number of wide characters placed in the output array.

71487 If copying takes place between objects that overlap, the behavior is undefined.

71488 **RETURN VALUE**

71489 If the total number of resulting wide-character codes including the terminating null wide-  
71490 character code is no more than *maxsize*, `wcsftime()` shall return the number of wide-character  
71491 codes placed into the array pointed to by *wcs*, not including the terminating null wide-character  
71492 code. Otherwise, zero is returned and the contents of the array are unspecified.

71493 **ERRORS**

71494 No errors are defined.

71495 **EXAMPLES**

71496 None.

71497 **APPLICATION USAGE**

71498 None.

71499 **RATIONALE**

71500 None.

71501 **FUTURE DIRECTIONS**

71502 None.

71503 **SEE ALSO**71504 [\*strftime\(\)\*](#)71505 XBD [\*\*<wchar.h>\*\*](#)



71506 **CHANGE HISTORY**

71507 First released in Issue 4.

71508 **Issue 5**

71509 Moved from ENHANCED I18N to BASE and the [ENOSYS] error is removed.

71510 Aligned with ISO/IEC 9899:1990/Amendment 1:1995 (E). Specifically, the type of the *format*  
71511 argument is changed from **const char \*** to **const wchar\_t \***.71512 **Issue 6**71513 The *wcsftime()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

71514 **NAME**

71515       wcslen, wcsnlen — get length of a fixed-sized wide-character string

71516 **SYNOPSIS**

71517       #include &lt;wchar.h&gt;

71518       size\_t wcslen(const wchar\_t \*ws);

71519 CX      size\_t wcsnlen(const wchar\_t \*ws, size\_t maxlen);

71520 **DESCRIPTION**

71521 CX      For *wcslen()*: The functionality described on this reference page is aligned with the ISO C  
 71522 standard. Any conflict between the requirements described here and the ISO C standard is  
 71523 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

71524       The *wcslen()* function shall compute the number of wide-character codes in the wide-character  
 71525 string to which *ws* points, not including the terminating null wide-character code.

71526 CX      The *wcsnlen()* function shall compute the smaller of the number of wide characters in the array  
 71527 to which *ws* points, not including any terminating null wide-character code, and the value of  
 71528 *maxlen*. The *wcsnlen()* function shall never examine more than the first *maxlen* characters of the  
 71529 wide-character array pointed to by *ws*.

71530 **RETURN VALUE**71531       The *wcslen()* function shall return the length of *ws*.

71532 CX      The *wcsnlen()* function shall return the number of wide characters preceding the first null wide-  
 71533 character code in the array to which *ws* points, if *ws* contains a null wide-character code within  
 71534 the first *maxlen* wide characters; otherwise, it shall return *maxlen*.

71535       No return values are reserved to indicate an error.

71536 **ERRORS**

71537       No errors are defined.

71538 **EXAMPLES**

71539       None.

71540 **APPLICATION USAGE**

71541       None.

71542 **RATIONALE**

71543       None.

71544 **FUTURE DIRECTIONS**

71545       None.

71546 **SEE ALSO**71547       *strlen()*

71548       XBD &lt;wchar.h&gt;

71549 **CHANGE HISTORY**

71550       First released in Issue 4. Derived from the MSE working draft.

71551 **Issue 7**

71552       The *wcsnlen()* function is added from The Open Group Technical Standard, 2006, Extended API  
 71553 Set Part 1.

71554 **NAME**

71555       wcsncasecmp, wcsncasecmp\_l — case-insensitive wide-character string comparison

71556 **SYNOPSIS**

```
71557 CX      #include <wchar.h>
71558          int wcsncasecmp(const wchar_t *ws1, const wchar_t *ws2, size_t n);
71559          int wcsncasecmp_l(const wchar_t *ws1, const wchar_t *ws2,
71560                          size_t n, locale_t locale);
```

71561 **DESCRIPTION**71562       Refer to *wcscasecmp()*.

71563 **NAME**

71564       wcsncat — concatenate a wide-character string with part of another

71565 **SYNOPSIS**

71566       #include &lt;wchar.h&gt;

71567       wchar\_t \*wcsncat(wchar\_t \*restrict ws1, const wchar\_t \*restrict ws2,  
71568                   size\_t n);71569 **DESCRIPTION**71570 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
71571       conflict between the requirements described here and the ISO C standard is unintentional. This  
71572       volume of POSIX.1-2008 defers to the ISO C standard.71573       The *wcsncat()* function shall append not more than *n* wide-character codes (a null wide-  
71574       character code and wide-character codes that follow it are not appended) from the array pointed  
71575       to by *ws2* to the end of the wide-character string pointed to by *ws1*. The initial wide-character  
71576       code of *ws2* shall overwrite the null wide-character code at the end of *ws1*. A terminating null  
71577       wide-character code shall always be appended to the result. If copying takes place between  
71578       objects that overlap, the behavior is undefined.71579 **RETURN VALUE**71580       The *wcsncat()* function shall return *ws1*; no return value is reserved to indicate an error.71581 **ERRORS**

71582       No errors are defined.

71583 **EXAMPLES**

71584       None.

71585 **APPLICATION USAGE**

71586       None.

71587 **RATIONALE**

71588       None.

71589 **FUTURE DIRECTIONS**

71590       None.

71591 **SEE ALSO**71592       *wcscat()*

71593       XBD &lt;wchar.h&gt;

71594 **CHANGE HISTORY**

71595       First released in Issue 4. Derived from the MSE working draft.

71596 **Issue 6**71597       The *wcsncat()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

71598 **NAME**

71599       wcsncmp — compare part of two wide-character strings

71600 **SYNOPSIS**

71601       #include &lt;wchar.h&gt;

71602       int wcsncmp(const wchar\_t \*ws1, const wchar\_t \*ws2, size\_t n);

71603 **DESCRIPTION**

71604 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
71605 conflict between the requirements described here and the ISO C standard is unintentional. This  
71606 volume of POSIX.1-2008 defers to the ISO C standard.

71607       The *wcsncmp()* function shall compare not more than *n* wide-character codes (wide-character  
71608 codes that follow a null wide-character code are not compared) from the array pointed to by *ws1*  
71609 to the array pointed to by *ws2*.

71610       The sign of a non-zero return value shall be determined by the sign of the difference between the  
71611 values of the first pair of wide-character codes that differ in the objects being compared.

71612 **RETURN VALUE**

71613       Upon successful completion, *wcsncmp()* shall return an integer greater than, equal to, or less  
71614 than 0, if the possibly null-terminated array pointed to by *ws1* is greater than, equal to, or less  
71615 than the possibly null-terminated array pointed to by *ws2*, respectively.

71616 **ERRORS**

71617       No errors are defined.

71618 **EXAMPLES**

71619       None.

71620 **APPLICATION USAGE**

71621       None.

71622 **RATIONALE**

71623       None.

71624 **FUTURE DIRECTIONS**

71625       None.

71626 **SEE ALSO**71627       *wscasecmp()*, *wscmp()*

71628       XBD &lt;wchar.h&gt;

71629 **CHANGE HISTORY**

71630       First released in Issue 4. Derived from the MSE working draft.

71631 **NAME**

71632       wcpncpy, wcsncpy — copy a fixed-size wide-character string, returning a pointer to its end

71633 **SYNOPSIS**

71634       #include &lt;wchar.h&gt;

71635 CX       wchar\_t \*wcpncpy(wchar\_t restrict \*ws1, const wchar\_t \*restrict ws2,  
71636           size\_t n);71637       wchar\_t \*wcsncpy(wchar\_t \*restrict ws1, const wchar\_t \*restrict ws2,  
71638           size\_t n);71639 **DESCRIPTION**71640 CX       For *wcsncpy()*: The functionality described on this reference page is aligned with the ISO C  
71641 standard. Any conflict between the requirements described here and the ISO C standard is  
71642 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.71643 CX       The *wcpncpy()* and *wcsncpy()* functions shall copy not more than *n* wide-character codes (wide-  
71644 character codes that follow a null wide-character code are not copied) from the array pointed to  
71645 by *ws2* to the array pointed to by *ws1*. If copying takes place between objects that overlap, the  
71646 behavior is undefined.71647       If the array pointed to by *ws2* is a wide-character string that is shorter than *n* wide-character  
71648 codes, null wide-character codes shall be appended to the copy in the array pointed to by *ws1*,  
71649 until *n* wide-character codes in all are written.71650 **RETURN VALUE**71651 CX       If any null wide-character codes were written into the destination, the *wcpncpy()* function shall  
71652 return the address of the first such null wide-character code. Otherwise, it shall return *&ws1[n]*.71653       The *wcsncpy()* function shall return *ws1*.

71654       No return values are reserved to indicate an error.

71655 **ERRORS**

71656       No errors are defined.

71657 **EXAMPLES**

71658       None.

71659 **APPLICATION USAGE**71660       If there is no null wide-character code in the first *n* wide-character codes of the array pointed to  
71661 by *ws2*, the result is not null-terminated.71662 **RATIONALE**

71663       None.

71664 **FUTURE DIRECTIONS**

71665       None.

71666 **SEE ALSO**71667       *strncpy()*, *wcscpy()*

71668       XBD &lt;wchar.h&gt;

71669 **CHANGE HISTORY**

71670       First released in Issue 4. Derived from the MSE working draft.

71671 **Issue 6**

71672 The *wcsncpy()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

71673 **Issue 7**

71674 The *wcpncpy()* function is added from The Open Group Technical Standard, 2006, Extended API  
71675 Set Part 1.

71676 **NAME**

71677       wcsnlen — get length of a fixed-sized wide-character string

71678 **SYNOPSIS**

```
71679 CX       #include <wchar.h>
71680       size_t wcsnlen(const wchar_t *ws, size_t maxlen);
```

71681 **DESCRIPTION**71682       Refer to *wcslen()*.



71683 **NAME**

71684       wcsnrtombs — convert wide-character string to multi-byte string

71685 **SYNOPSIS**

```
71686 CX       #include <wchar.h>
71687       size_t wcsnrtombs(char *restrict dst, const wchar_t **restrict src,
71688                       size_t nwc, size_t len, mbstate_t *restrict ps);
```

71689 **DESCRIPTION**71690       Refer to *wcsrtombs()*.

71691 **NAME**

71692 wcpbrk — scan a wide-character string for a wide-character code

71693 **SYNOPSIS**

71694 #include &lt;wchar.h&gt;

71695 wchar\_t \*wcpbrk(const wchar\_t \*ws1, const wchar\_t \*ws2);

71696 **DESCRIPTION**

71697 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
71698 conflict between the requirements described here and the ISO C standard is unintentional. This  
71699 volume of POSIX.1-2008 defers to the ISO C standard.

71700 The *wcpbrk()* function shall locate the first occurrence in the wide-character string pointed to by  
71701 *ws1* of any wide-character code from the wide-character string pointed to by *ws2*.

71702 **RETURN VALUE**

71703 Upon successful completion, *wcpbrk()* shall return a pointer to the wide-character code or a null  
71704 pointer if no wide-character code from *ws2* occurs in *ws1*.

71705 **ERRORS**

71706 No errors are defined.

71707 **EXAMPLES**

71708 None.

71709 **APPLICATION USAGE**

71710 None.

71711 **RATIONALE**

71712 None.

71713 **FUTURE DIRECTIONS**

71714 None.

71715 **SEE ALSO**71716 *wcschr()*, *wcsrchr()*

71717 XBD &lt;wchar.h&gt;

71718 **CHANGE HISTORY**

71719 First released in Issue 4. Derived from the MSE working draft.

71720 **NAME**

71721 wcsrchr — wide-character string scanning operation

71722 **SYNOPSIS**

71723 #include &lt;wchar.h&gt;

71724 wchar\_t \*wcsrchr(const wchar\_t \*ws, wchar\_t wc);

71725 **DESCRIPTION**

71726 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
71727 conflict between the requirements described here and the ISO C standard is unintentional. This  
71728 volume of POSIX.1-2008 defers to the ISO C standard.

71729 The *wcsrchr()* function shall locate the last occurrence of *wc* in the wide-character string pointed  
71730 to by *ws*. The application shall ensure that the value of *wc* is a character representable as a type  
71731 **wchar\_t** and a wide-character code corresponding to a valid character in the current locale. The  
71732 terminating null wide-character code shall be considered to be part of the wide-character string.

71733 **RETURN VALUE**

71734 Upon successful completion, *wcsrchr()* shall return a pointer to the wide-character code or a null  
71735 pointer if *wc* does not occur in the wide-character string.

71736 **ERRORS**

71737 No errors are defined.

71738 **EXAMPLES**

71739 None.

71740 **APPLICATION USAGE**

71741 None.

71742 **RATIONALE**

71743 None.

71744 **FUTURE DIRECTIONS**

71745 None.

71746 **SEE ALSO**71747 *wcschr()*

71748 XBD &lt;wchar.h&gt;

71749 **CHANGE HISTORY**

71750 First released in Issue 4. Derived from the MSE working draft.

71751 **Issue 6**

71752 The normative text is updated to avoid use of the term “must” for application requirements.

## 71753 NAME

71754 wcsnrtombs, wcsrtombs — convert a wide-character string to a character string (restartable)

## 71755 SYNOPSIS

71756 #include &lt;wchar.h&gt;

```
71757 CX size_t wcsnrtombs(char *restrict dst, const wchar_t **restrict src,
71758 size_t nwc, size_t len, mbstate_t *restrict ps);
71759 size_t wcsrtombs(char *restrict dst, const wchar_t **restrict src,
71760 size_t len, mbstate_t *restrict ps);
```

## 71761 DESCRIPTION

71762 CX For *wcsrtombs()*: The functionality described on this reference page is aligned with the ISO C  
 71763 standard. Any conflict between the requirements described here and the ISO C standard is  
 71764 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

71765 The *wcsrtombs()* function shall convert a sequence of wide characters from the array indirectly  
 71766 pointed to by *src* into a sequence of corresponding characters, beginning in the conversion state  
 71767 described by the object pointed to by *ps*. If *dst* is not a null pointer, the converted characters  
 71768 shall then be stored into the array pointed to by *dst*. Conversion continues up to and including a  
 71769 terminating null wide character, which shall also be stored. Conversion shall stop earlier in the  
 71770 following cases:

- 71771 • When a code is reached that does not correspond to a valid character
- 71772 • When the next character would exceed the limit of *len* total bytes to be stored in the array
- 71773 pointed to by *dst* (and *dst* is not a null pointer)

71774 Each conversion shall take place as if by a call to the *wcrtomb()* function.

71775 If *dst* is not a null pointer, the pointer object pointed to by *src* shall be assigned either a null  
 71776 pointer (if conversion stopped due to reaching a terminating null wide character) or the address  
 71777 just past the last wide character converted (if any). If conversion stopped due to reaching a  
 71778 terminating null wide character, the resulting state described shall be the initial conversion state.

71779 If *ps* is a null pointer, the *wcsrtombs()* function shall use its own internal **mbstate\_t** object, which  
 71780 is initialized at program start-up to the initial conversion state. Otherwise, the **mbstate\_t** object  
 71781 pointed to by *ps* shall be used to completely describe the current conversion state of the  
 71782 associated character sequence.

71783 CX The *wcsnrtombs()* and *wcsrtombs()* functions need not be thread-safe if called with a NULL *ps*  
 71784 argument.

71785 The *wcsnrtombs()* function shall be equivalent to the *wcsrtombs()* function, except that the  
 71786 conversion is limited to the first *nwc* wide characters.

71787 The *wcsrtombs()* function shall not change the setting of *errno* if successful.71788 The behavior of these functions shall be affected by the *LC\_CTYPE* category of the current locale.

71789 The implementation shall behave as if no function defined in System Interfaces volume of  
 71790 POSIX.1-2008 calls these functions.

## 71791 RETURN VALUE

71792 If conversion stops because a code is reached that does not correspond to a valid character, an  
 71793 encoding error occurs. In this case, these functions shall store the value of the macro [EILSEQ] in  
 71794 *errno* and return (**size\_t**)–1; the conversion state is undefined. Otherwise, these functions shall  
 71795 return the number of bytes in the resulting character sequence, not including the terminating  
 71796 null (if any).

**71797 ERRORS**

71798 These functions shall fail if:

71799 [EILSEQ] A wide-character code does not correspond to a valid character.

71800 These functions may fail if:

71801 CX [EINVAL] *ps* points to an object that contains an invalid conversion state.

**71802 EXAMPLES**

71803 None.

**71804 APPLICATION USAGE**

71805 None.

**71806 RATIONALE**

71807 None.

**71808 FUTURE DIRECTIONS**

71809 None.

**71810 SEE ALSO**

71811 *mbsinit()*, *wcrtomb()*

71812 XBD <wchar.h>

**71813 CHANGE HISTORY**

71814 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
71815 (E).

**71816 Issue 6**

71817 In the DESCRIPTION, a note on using this function in a threaded application is added.

71818 Extensions beyond the ISO C standard are marked.

71819 The normative text is updated to avoid use of the term “must” for application requirements.

71820 The *wcsrtombs()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

**71821 Issue 7**

71822 Austin Group Interpretation 1003.1-2001 #148 is applied, clarifying that the *wcsrtombs()* function  
71823 need not be thread-safe if called with a NULL *ps* argument.

71824 Austin Group Interpretation 1003.1-2001 #170 is applied.

71825 The *wcsnrtombs()* function is added from The Open Group Technical Standard, 2006, Extended  
71826 API Set Part 1.

71827 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0722 [109,105] is applied.

71828 **NAME**

71829       wcsspnp — get the length of a wide substring

71830 **SYNOPSIS**

71831       #include &lt;wchar.h&gt;

71832       size\_t wcsspnp(const wchar\_t \*ws1, const wchar\_t \*ws2);

71833 **DESCRIPTION**

71834 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
71835       conflict between the requirements described here and the ISO C standard is unintentional. This  
71836       volume of POSIX.1-2008 defers to the ISO C standard.

71837       The *wcsspnp()* function shall compute the length (in wide characters) of the maximum initial  
71838       segment of the wide-character string pointed to by *ws1* which consists entirely of wide-character  
71839       codes from the wide-character string pointed to by *ws2*.

71840 **RETURN VALUE**

71841       The *wcsspnp()* function shall return the length of the initial substring of *ws1*; no return value is  
71842       reserved to indicate an error.

71843 **ERRORS**

71844       No errors are defined.

71845 **EXAMPLES**

71846       None.

71847 **APPLICATION USAGE**

71848       None.

71849 **RATIONALE**

71850       None.

71851 **FUTURE DIRECTIONS**

71852       None.

71853 **SEE ALSO**71854       *wcscspnp()*

71855       XBD &lt;wchar.h&gt;

71856 **CHANGE HISTORY**

71857       First released in Issue 4. Derived from the MSE working draft.

71858 **Issue 5**

71859       The RETURN VALUE section is updated to indicate that *wcsspnp()* returns the length of *ws1*  
71860       rather than *ws1* itself.

71861 **NAME**71862        `wcsstr` — find a wide-character substring71863 **SYNOPSIS**71864        `#include <wchar.h>`71865        `wchar_t *wcsstr(const wchar_t *restrict ws1,`  
71866                `const wchar_t *restrict ws2);`71867 **DESCRIPTION**71868 **CX**        The functionality described on this reference page is aligned with the ISO C standard. Any  
71869        conflict between the requirements described here and the ISO C standard is unintentional. This  
71870        volume of POSIX.1-2008 defers to the ISO C standard.71871        The `wcsstr()` function shall locate the first occurrence in the wide-character string pointed to by  
71872        `ws1` of the sequence of wide characters (excluding the terminating null wide character) in the  
71873        wide-character string pointed to by `ws2`.71874 **RETURN VALUE**71875        Upon successful completion, `wcsstr()` shall return a pointer to the located wide-character string,  
71876        or a null pointer if the wide-character string is not found.71877        If `ws2` points to a wide-character string with zero length, the function shall return `ws1`.71878 **ERRORS**

71879        No errors are defined.

71880 **EXAMPLES**

71881        None.

71882 **APPLICATION USAGE**

71883        None.

71884 **RATIONALE**

71885        None.

71886 **FUTURE DIRECTIONS**

71887        None.

71888 **SEE ALSO**71889        [\*wcschr\(\)\*](#)71890        XBD [\*<wchar.h>\*](#)71891 **CHANGE HISTORY**71892        First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
71893        (E).71894 **Issue 6**71895        The `wcsstr()` prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

## NAME

wcstod, wcstof, wcstold — convert a wide-character string to a double-precision number

## SYNOPSIS

```
#include <wchar.h>

double wcstod(const wchar_t *restrict nptr, wchar_t **restrict endptr);
float wcstof(const wchar_t *restrict nptr, wchar_t **restrict endptr);
long double wcstold(const wchar_t *restrict nptr,
    wchar_t **restrict endptr);
```

## DESCRIPTION

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

These functions shall convert the initial portion of the wide-character string pointed to by *nptr* to **double**, **float**, and **long double** representation, respectively. First, they shall decompose the input wide-character string into three parts:

1. An initial, possibly empty, sequence of white-space wide-character codes (as specified by *iswspace()*)
2. A subject sequence interpreted as a floating-point constant or representing infinity or NaN
3. A final wide-character string of one or more unrecognized wide-character codes, including the terminating null wide-character code of the input wide-character string

Then they shall attempt to convert the subject sequence to a floating-point number, and return the result.

The expected form of the subject sequence is an optional '+' or '-' sign, then one of the following:

- A non-empty sequence of decimal digits optionally containing a radix character; then an optional exponent part consisting of the wide character 'e' or the wide character 'E', optionally followed by a '+' or '-' wide character, and then followed by one or more decimal digits
- A 0x or 0X, then a non-empty sequence of hexadecimal digits optionally containing a radix character; then an optional binary exponent part consisting of the wide character 'p' or the wide character 'P', optionally followed by a '+' or '-' wide character, and then followed by one or more decimal digits
- One of INF or INFINITY, or any other wide string equivalent except for case
- One of NAN or NAN(*n-wchar-sequence<sub>opt</sub>*), or any other wide string ignoring case in the NAN part, where:

```
n-wchar-sequence:
    digit
    nondigit
    n-wchar-sequence digit
    n-wchar-sequence nondigit
```

The subject sequence is defined as the longest initial subsequence of the input wide string, starting with the first non-white-space wide character, that is of the expected form. The subject sequence contains no wide characters if the input wide string is not of the expected form.



If the subject sequence has the expected form for a floating-point number, the sequence of wide characters starting with the first digit or the radix character (whichever occurs first) shall be interpreted as a floating constant according to the rules of the C language, except that the radix character shall be used in place of a period, and that if neither an exponent part nor a radix character appears in a decimal floating-point number, or if a binary exponent part does not appear in a hexadecimal floating-point number, an exponent part of the appropriate type with value zero shall be assumed to follow the last digit in the string. If the subject sequence begins with a <hyphen-minus>, the sequence shall be interpreted as negated. A wide-character sequence INF or INFINITY shall be interpreted as an infinity, if representable in the return type, else as if it were a floating constant that is too large for the range of the return type. A wide-character sequence NAN or NAN(*n-wchar-sequence<sub>opt</sub>*) shall be interpreted as a quiet NaN, if supported in the return type, else as if it were a subject sequence part that does not have the expected form; the meaning of the *n-wchar* sequences is implementation-defined. A pointer to the final wide string shall be stored in the object pointed to by *endptr*, provided that *endptr* is not a null pointer.

If the subject sequence has the hexadecimal form and FLT\_RADIX is a power of 2, the conversion shall be rounded in an implementation-defined manner.

CX The radix character shall be as defined in the current locale (category *LC\_NUMERIC*). In the POSIX locale, or in a locale where the radix character is not defined, the radix character shall default to a <period> ( ' . ' ).

CX In other than the C or POSIX locale, additional locale-specific subject sequence forms may be accepted.

If the subject sequence is empty or does not have the expected form, no conversion shall be performed; the value of *nptr* shall be stored in the object pointed to by *endptr*, provided that *endptr* is not a null pointer.

These functions shall not change the setting of *errno* if successful.

Since 0 is returned on error and is also a valid return on success, an application wishing to check for error situations should set *errno* to 0, then call *wcstod()*, *wcstof()*, or *wcstold()*, then check *errno*.

## 71969 RETURN VALUE

Upon successful completion, these functions shall return the converted value. If no conversion could be performed, 0 shall be returned and *errno* may be set to [EINVAL].

If the correct value is outside the range of representable values,  $\pm$ HUGE\_VAL,  $\pm$ HUGE\_VALF, or  $\pm$ HUGE\_VALL shall be returned (according to the sign of the value), and *errno* shall be set to [ERANGE].

If the correct value would cause underflow, a value whose magnitude is no greater than the smallest normalized positive number in the return type shall be returned and *errno* set to [ERANGE].

## 71978 ERRORS

The *wcstod()* function shall fail if:

[ERANGE] The value to be returned would cause overflow or underflow.

The *wcstod()* function may fail if:

CX [EINVAL] No conversion could be performed.

**EXAMPLES**

None.

**APPLICATION USAGE**

If the subject sequence has the hexadecimal form and FLT\_RADIX is not a power of 2, and the result is not exactly representable, the result should be one of the two numbers in the appropriate internal format that are adjacent to the hexadecimal floating source value, with the extra stipulation that the error should have a correct sign for the current rounding direction.

If the subject sequence has the decimal form and at most DECIMAL\_DIG (defined in `<float.h>`) significant digits, the result should be correctly rounded. If the subject sequence *D* has the decimal form and more than DECIMAL\_DIG significant digits, consider the two bounding, adjacent decimal strings *L* and *U*, both having DECIMAL\_DIG significant digits, such that the values of *L*, *D*, and *U* satisfy "*L* <= *D* <= *U*". The result should be one of the (equal or adjacent) values that would be obtained by correctly rounding *L* and *U* according to the current rounding direction, with the extra stipulation that the error with respect to *D* should have a correct sign for the current rounding direction.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[\*fscanf\(\)\*](#), [\*iswspace\(\)\*](#), [\*localeconv\(\)\*](#), [\*setlocale\(\)\*](#), [\*wcstol\(\)\*](#)

XBD [Chapter 7](#) (on page 135), `<float.h>`, `<wchar.h>`

**CHANGE HISTORY**

First released in Issue 4. Derived from the MSE working draft.

**Issue 5**

The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

**Issue 6**

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is added if no conversion could be performed.

The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- The *wcstod()* prototype is updated.
- The *wcstof()* and *wcstold()* functions are added.
- If the correct value for *wcstod()* would cause underflow, the return value changed from 0 (as specified in Issue 5) to the smallest normalized positive number.
- The DESCRIPTION, RETURN VALUE, and APPLICATION USAGE sections are extensively updated.

ISO/IEC 9899:1999 standard, Technical Corrigendum 1 is incorporated.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/66 is applied, correcting the second paragraph in the RETURN VALUE section.

72025 **Issue 7**

72026 Austin Group Interpretation 1003.1-2001 #015 is applied.

72027 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0723 [302] and XSH/TC1-2008/0724  
72028 [105] are applied.

72029 **NAME**

72030       wcstoimax, wcstoumax — convert a wide-character string to an integer type

72031 **SYNOPSIS**

```
72032       #include <stddef.h>
72033       #include <inttypes.h>
72034       intmax_t wcstoimax(const wchar_t *restrict nptr,
72035                        wchar_t **restrict endptr, int base);
72036       uintmax_t wcstoumax(const wchar_t *restrict nptr,
72037                        wchar_t **restrict endptr, int base);
```

72038 **DESCRIPTION**

72039 CX       The functionality described on this reference page is aligned with the ISO C standard. Any  
72040       conflict between the requirements described here and the ISO C standard is unintentional. This  
72041       volume of POSIX.1-2008 defers to the ISO C standard.

72042       These functions shall be equivalent to the *wcstol()*, *wcstoll()*, *wcstoul()*, and *wcstoull()* functions,  
72043       respectively, except that the initial portion of the wide string shall be converted to **intmax\_t** and  
72044       **uintmax\_t** representation, respectively.

72045 **RETURN VALUE**

72046       These functions shall return the converted value, if any.

72047       If no conversion could be performed, zero shall be returned. If the correct value is outside the  
72048       range of representable values, {INTMAX\_MAX}, {INTMAX\_MIN}, or {UINTMAX\_MAX} shall  
72049       be returned (according to the return type and sign of the value, if any), and *errno* shall be set to  
72050       [ERANGE].

72051 **ERRORS**

72052       These functions shall fail if:

72053       [EINVAL]       The value of *base* is not supported.

72054       [ERANGE]       The value to be returned is not representable.

72055       These functions may fail if:

72056       [EINVAL]       No conversion could be performed.

72057 **EXAMPLES**

72058       None.

72059 **APPLICATION USAGE**

72060       None.

72061 **RATIONALE**

72062       None.

72063 **FUTURE DIRECTIONS**

72064       None.

72065 **SEE ALSO**72066       *wcstol()*, *wcstoul()*

72067       XBD &lt;inttypes.h&gt;, &lt;stddef.h&gt;

72068 **CHANGE HISTORY**

72069       First released in Issue 6. Derived from the ISO/IEC 9899:1999 standard.

**NAME**

wcstok — split a wide-character string into tokens

**SYNOPSIS**

```
#include <wchar.h>
```

```
wchar_t *wcstok(wchar_t *restrict ws1, const wchar_t *restrict ws2,  
                wchar_t **restrict ptr);
```

**DESCRIPTION**

CX The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

A sequence of calls to *wcstok()* shall break the wide-character string pointed to by *ws1* into a sequence of tokens, each of which shall be delimited by a wide-character code from the wide-character string pointed to by *ws2*. The *ptr* argument points to a caller-provided **wchar\_t** pointer into which the *wcstok()* function shall store information necessary for it to continue scanning the same wide-character string.

The first call in the sequence has *ws1* as its first argument, and is followed by calls with a null pointer as their first argument. The separator string pointed to by *ws2* may be different from call to call.

The first call in the sequence shall search the wide-character string pointed to by *ws1* for the first wide-character code that is *not* contained in the current separator string pointed to by *ws2*. If no such wide-character code is found, then there are no tokens in the wide-character string pointed to by *ws1* and *wcstok()* shall return a null pointer. If such a wide-character code is found, it shall be the start of the first token.

The *wcstok()* function shall then search from there for a wide-character code that *is* contained in the current separator string. If no such wide-character code is found, the current token extends to the end of the wide-character string pointed to by *ws1*, and subsequent searches for a token shall return a null pointer. If such a wide-character code is found, it shall be overwritten by a null wide character, which terminates the current token. The *wcstok()* function shall save a pointer to the following wide-character code, from which the next search for a token shall start.

Each subsequent call, with a null pointer as the value of the first argument, shall start searching from the saved pointer and behave as described above.

The implementation shall behave as if no function calls *wcstok()*.

**RETURN VALUE**

Upon successful completion, the *wcstok()* function shall return a pointer to the first wide-character code of a token. Otherwise, if there is no token, *wcstok()* shall return a null pointer.

**ERRORS**

No errors are defined.

72107 **EXAMPLES**

72108 None.

72109 **APPLICATION USAGE**

72110 None.

72111 **RATIONALE**

72112 None.

72113 **FUTURE DIRECTIONS**

72114 None.

72115 **SEE ALSO**

72116 XBD &lt;wchar.h&gt;

72117 **CHANGE HISTORY**

72118 First released in Issue 4.

72119 **Issue 5**72120 Aligned with ISO/IEC 9899:1990/Amendment 1:1995 (E). Specifically, a third argument is  
72121 added to the definition of *wcstok()* in the SYNOPSIS.72122 **Issue 6**72123 The *wcstok()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

## NAME

wcstol, wcstoll — convert a wide-character string to a long integer

## SYNOPSIS

```
#include <wchar.h>

long wcstol(const wchar_t *restrict nptr, wchar_t **restrict endptr,
            int base);
long long wcstoll(const wchar_t *restrict nptr,
                  wchar_t **restrict endptr, int base);
```

## DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

These functions shall convert the initial portion of the wide-character string pointed to by *nptr* to **long** and **long long**, respectively. First, they shall decompose the input string into three parts:

1. An initial, possibly empty, sequence of white-space wide-character codes (as specified by *iswspace()*)
2. A subject sequence interpreted as an integer represented in some radix determined by the value of *base*
3. A final wide-character string of one or more unrecognized wide-character codes, including the terminating null wide-character code of the input wide-character string

Then they shall attempt to convert the subject sequence to an integer, and return the result.

If *base* is 0, the expected form of the subject sequence is that of a decimal constant, octal constant, or hexadecimal constant, any of which may be preceded by a '+' or '-' sign. A decimal constant begins with a non-zero digit, and consists of a sequence of decimal digits. An octal constant consists of the prefix '0' optionally followed by a sequence of the digits '0' to '7' only. A hexadecimal constant consists of the prefix 0x or 0X followed by a sequence of the decimal digits and letters 'a' (or 'A') to 'f' (or 'F') with values 10 to 15 respectively.

If the value of *base* is between 2 and 36, the expected form of the subject sequence is a sequence of letters and digits representing an integer with the radix specified by *base*, optionally preceded by a '+' or '-' sign, but not including an integer suffix. The letters from 'a' (or 'A') to 'z' (or 'Z') inclusive are ascribed the values 10 to 35; only letters whose ascribed values are less than that of *base* shall be permitted. If the value of *base* is 16, the wide-character code representations of 0x or 0X may optionally precede the sequence of letters and digits, following the sign if present.

The subject sequence is defined as the longest initial subsequence of the input wide-character string, starting with the first non-white-space wide-character code that is of the expected form. The subject sequence contains no wide-character codes if the input wide-character string is empty or consists entirely of white-space wide-character code, or if the first non-white-space wide-character code is other than a sign or a permissible letter or digit.

If the subject sequence has the expected form and *base* is 0, the sequence of wide-character codes starting with the first digit shall be interpreted as an integer constant. If the subject sequence has the expected form and the value of *base* is between 2 and 36, it shall be used as the base for conversion, ascribing to each letter its value as given above. If the subject sequence begins with a <hyphen-minus>, the value resulting from the conversion shall be negated. A pointer to the final wide-character string shall be stored in the object pointed to by *endptr*, provided that *endptr* is not a null pointer.

72170 CX In other than the C or POSIX locale, additional locale-specific subject sequence forms may be  
72171 accepted.

72172 If the subject sequence is empty or does not have the expected form, no conversion shall be  
72173 performed; the value of *nptr* shall be stored in the object pointed to by *endptr*, provided that  
72174 *endptr* is not a null pointer.

72175 These functions shall not change the setting of *errno* if successful.

72176 Since 0, {LONG\_MIN} or {LLONG\_MIN} and {LONG\_MAX} or {LLONG\_MAX} are returned on  
72177 error and are also valid returns on success, an application wishing to check for error situations  
72178 should set *errno* to 0, then call *wcstol()* or *wcstoll()*, then check *errno*.

#### 72179 RETURN VALUE

72180 Upon successful completion, these functions shall return the converted value, if any. If no  
72181 CX conversion could be performed, 0 shall be returned and *errno* may be set to indicate the error. If  
72182 the correct value is outside the range of representable values, {LONG\_MIN}, {LONG\_MAX},  
72183 {LLONG\_MIN}, or {LLONG\_MAX} shall be returned (according to the sign of the value), and  
72184 *errno* set to [ERANGE].

#### 72185 ERRORS

72186 These functions shall fail if:

72187 CX [EINVAL] The value of *base* is not supported.  
72188 [ERANGE] The value to be returned is not representable.

72189 These functions may fail if:

72190 CX [EINVAL] No conversion could be performed.

#### 72191 EXAMPLES

72192 None.

#### 72193 APPLICATION USAGE

72194 None.

#### 72195 RATIONALE

72196 None.

#### 72197 FUTURE DIRECTIONS

72198 None.

#### 72199 SEE ALSO

72200 *fscanf()*, *iswalpha()*, *wcstod()*

72201 XBD <wchar.h>

#### 72202 CHANGE HISTORY

72203 First released in Issue 4. Derived from the MSE working draft.

#### 72204 Issue 5

72205 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

#### 72206 Issue 6

72207 Extensions beyond the ISO C standard are marked.



- 72208 The following new requirements on POSIX implementations derive from alignment with the  
72209 Single UNIX Specification:
- 72210 • In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is  
72211 added if no conversion could be performed.
- 72212 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:
- 72213 • The *wcstol()* prototype is updated.
  - 72214 • The *wcstoll()* function is added.
- 72215 **Issue 7**
- 72216 SD5-XSH-ERN-56 is applied, removing the reference to **unsigned long** and **unsigned long long**  
72217 from the DESCRIPTION.
- 72218 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0725 [105] is applied.

72219 **NAME**

72220       wcstold — convert a wide-character string to a double-precision number

72221 **SYNOPSIS**

72222       #include &lt;wchar.h&gt;

72223       long double wcstold(const wchar\_t \*restrict nptr,

72224                   wchar\_t \*\*restrict endptr);

72225 **DESCRIPTION**72226       Refer to *wcstod()*.

72227 **NAME**

72228       wcstoll — convert a wide-character string to a long integer

72229 **SYNOPSIS**

72230       #include &lt;wchar.h&gt;

72231       long long wcstoll(const wchar\_t \*restrict *nptr*,72232                       wchar\_t \*\*restrict *endptr*, int *base*);72233 **DESCRIPTION**72234       Refer to *wcstol()*.

72235 **NAME**

72236 wcstombs — convert a wide-character string to a character string

72237 **SYNOPSIS**

72238 #include &lt;stdlib.h&gt;

```
72239 size_t wcstombs(char *restrict s, const wchar_t *restrict pwcs,
72240                 size_t n);
```

72241 **DESCRIPTION**

72242 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 72243 conflict between the requirements described here and the ISO C standard is unintentional. This  
 72244 volume of POSIX.1-2008 defers to the ISO C standard.

72245 The *wcstombs()* function shall convert the sequence of wide-character codes that are in the array  
 72246 pointed to by *pwcs* into a sequence of characters that begins in the initial shift state and store  
 72247 these characters into the array pointed to by *s*, stopping if a character would exceed the limit of *n*  
 72248 total bytes or if a null byte is stored. Each wide-character code shall be converted as if by a call to  
 72249 *wctomb()*, except that the shift state of *wctomb()* shall not be affected.

72250 The behavior of this function shall be affected by the *LC\_CTYPE* category of the current locale.

72251 No more than *n* bytes shall be modified in the array pointed to by *s*. If copying takes place  
 72252 CX between objects that overlap, the behavior is undefined. If *s* is a null pointer, *wcstombs()* shall  
 72253 return the length required to convert the entire array regardless of the value of *n*, but no values  
 72254 are stored.

72255 **RETURN VALUE**

72256 If a wide-character code is encountered that does not correspond to a valid character (of one or  
 72257 more bytes each), *wcstombs()* shall return (**size\_t**)–1. Otherwise, *wcstombs()* shall return the  
 72258 number of bytes stored in the character array, not including any terminating null byte. The array  
 72259 shall not be null-terminated if the value returned is *n*.

72260 **ERRORS**72261 The *wcstombs()* function shall fail if:

72262 CX [EILSEQ] A wide-character code does not correspond to a valid character.

72263 **EXAMPLES**

72264 None.

72265 **APPLICATION USAGE**

72266 None.

72267 **RATIONALE**

72268 None.

72269 **FUTURE DIRECTIONS**

72270 None.

72271 **SEE ALSO**72272 *mblen()*, *mbtowc()*, *mbstowcs()*, *wctomb()*

72273 XBD &lt;stdlib.h&gt;

72274 **CHANGE HISTORY**

72275 First released in Issue 4. Derived from the ISO C standard.

72276 **Issue 6**

72277 The following new requirements on POSIX implementations derive from alignment with the  
72278 Single UNIX Specification:

- 72279 • The DESCRIPTION states the effect of when *s* is a null pointer.
- 72280 • The [EILSEQ] error condition is added.

72281 The *wcstombs()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

72282 **Issue 7**

72283 Austin Group Interpretations 1003.1-2001 #156 and #170 are applied.

72284 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0726 [109] is applied.

## NAME

wcstoul, wcstoull — convert a wide-character string to an unsigned long

## SYNOPSIS

```
#include <wchar.h>

unsigned long wcstoul(const wchar_t *restrict nptr,
    wchar_t **restrict endptr, int base);
unsigned long long wcstoull(const wchar_t *restrict nptr,
    wchar_t **restrict endptr, int base);
```

## DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

The *wcstoul()* and *wcstoull()* functions shall convert the initial portion of the wide-character string pointed to by *nptr* to **unsigned long** and **unsigned long long** representation, respectively. First, they shall decompose the input wide-character string into three parts:

1. An initial, possibly empty, sequence of white-space wide-character codes (as specified by *iswspace()*)
2. A subject sequence interpreted as an integer represented in some radix determined by the value of *base*
3. A final wide-character string of one or more unrecognized wide-character codes, including the terminating null wide-character code of the input wide-character string

Then they shall attempt to convert the subject sequence to an unsigned integer, and return the result.

If *base* is 0, the expected form of the subject sequence is that of a decimal constant, octal constant, or hexadecimal constant, any of which may be preceded by a '+' or '-' sign. A decimal constant begins with a non-zero digit, and consists of a sequence of decimal digits. An octal constant consists of the prefix '0' optionally followed by a sequence of the digits '0' to '7' only. A hexadecimal constant consists of the prefix 0x or 0X followed by a sequence of the decimal digits and letters 'a' (or 'A') to 'f' (or 'F') with values 10 to 15 respectively.

If the value of *base* is between 2 and 36, the expected form of the subject sequence is a sequence of letters and digits representing an integer with the radix specified by *base*, optionally preceded by a '+' or '-' sign, but not including an integer suffix. The letters from 'a' (or 'A') to 'z' (or 'Z') inclusive are ascribed the values 10 to 35; only letters whose ascribed values are less than that of *base* shall be permitted. If the value of *base* is 16, the wide-character codes 0x or 0X may optionally precede the sequence of letters and digits, following the sign if present.

The subject sequence is defined as the longest initial subsequence of the input wide-character string, starting with the first wide-character code that is not white space and is of the expected form. The subject sequence contains no wide-character codes if the input wide-character string is empty or consists entirely of white-space wide-character codes, or if the first wide-character code that is not white space is other than a sign or a permissible letter or digit.

If the subject sequence has the expected form and *base* is 0, the sequence of wide-character codes starting with the first digit shall be interpreted as an integer constant. If the subject sequence has the expected form and the value of *base* is between 2 and 36, it shall be used as the base for conversion, ascribing to each letter its value as given above. If the subject sequence begins with a <hyphen-minus>, the value resulting from the conversion shall be negated. A pointer to the final wide-character string shall be stored in the object pointed to by *endptr*, provided that *endptr* is

72331 not a null pointer.

72332 CX In other than the C or POSIX locale, additional locale-specific subject sequence forms may be  
72333 accepted.

72334 If the subject sequence is empty or does not have the expected form, no conversion shall be  
72335 performed; the value of *nptr* shall be stored in the object pointed to by *endptr*, provided that  
72336 *endptr* is not a null pointer.

72337 These functions shall not change the setting of *errno* if successful.

72338 Since 0, {ULONG\_MAX}, and {ULLONG\_MAX} are returned on error and 0 is also a valid return  
72339 on success, an application wishing to check for error situations should set *errno* to 0, then call  
72340 *wcstoul()* or *wcstoull()*, then check *errno*.

#### 72341 RETURN VALUE

72342 Upon successful completion, the *wcstoul()* and *wcstoull()* functions shall return the converted  
72343 CX value, if any. If no conversion could be performed, 0 shall be returned and *errno* may be set to  
72344 indicate the error. If the correct value is outside the range of representable values,  
72345 {ULONG\_MAX} or {ULLONG\_MAX} respectively shall be returned and *errno* set to [ERANGE].

#### 72346 ERRORS

72347 These functions shall fail if:

72348 CX [EINVAL] The value of *base* is not supported.

72349 [ERANGE] The value to be returned is not representable.

72350 These functions may fail if:

72351 CX [EINVAL] No conversion could be performed.

#### 72352 EXAMPLES

72353 None.

#### 72354 APPLICATION USAGE

72355 None.

#### 72356 RATIONALE

72357 None.

#### 72358 FUTURE DIRECTIONS

72359 None.

#### 72360 SEE ALSO

72361 *fscanf()*, *iswalpha()*, *wcstod()*, *wcstol()*

72362 XBD <wchar.h>

#### 72363 CHANGE HISTORY

72364 First released in Issue 4. Derived from the MSE working draft.

#### 72365 Issue 5

72366 The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

#### 72367 Issue 6

72368 Extensions beyond the ISO C standard are marked.

72369 The following new requirements on POSIX implementations derive from alignment with the  
72370 Single UNIX Specification:

- 72371 • The [EINVAL] error condition is added for when the value of *base* is not supported.

72372 In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is  
72373 added if no conversion could be performed.

72374 The following changes are made for alignment with the ISO/IEC 9899:1999 standard:

- 72375 • The *wcstoul()* prototype is updated.
- 72376 • The *wcstoull()* function is added.

72377 **Issue 7**

72378 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0727 [105] is applied.



72379 **NAME**

72380       wcstoumax — convert a wide-character string to an integer type

72381 **SYNOPSIS**

72382       #include &lt;stddef.h&gt;

72383       #include &lt;inttypes.h&gt;

72384       uintmax\_t wcstoumax(const wchar\_t \*restrict nptr,

72385                    wchar\_t \*\*restrict endptr, int base);

72386 **DESCRIPTION**72387       Refer to *wcstoimax()*.

72388 **NAME**

72389       wcswidth — number of column positions of a wide-character string

72390 **SYNOPSIS**

72391 XSI       #include &lt;wchar.h&gt;

72392       int wcswidth(const wchar\_t \*pwcs, size\_t n);

72393 **DESCRIPTION**

72394       The *wcswidth()* function shall determine the number of column positions required for *n* wide-  
72395       character codes (or fewer than *n* wide-character codes if a null wide-character code is  
72396       encountered before *n* wide-character codes are exhausted) in the string pointed to by *pwcs*.

72397 **RETURN VALUE**

72398       The *wcswidth()* function either shall return 0 (if *pwcs* points to a null wide-character code), or  
72399       return the number of column positions to be occupied by the wide-character string pointed to by  
72400       *pwcs*, or return -1 (if any of the first *n* wide-character codes in the wide-character string pointed  
72401       to by *pwcs* is not a printable wide-character code).

72402 **ERRORS**

72403       No errors are defined.

72404 **EXAMPLES**

72405       None.

72406 **APPLICATION USAGE**

72407       This function was removed from the final ISO/IEC 9899:1990/Amendment 1:1995 (E), and the  
72408       return value for a non-printable wide character is not specified.

72409 **RATIONALE**

72410       None.

72411 **FUTURE DIRECTIONS**

72412       None.

72413 **SEE ALSO**72414       *wcwidth()*72415       XBD [Section 3.103](#) (on page 50), [<wchar.h>](#)72416 **CHANGE HISTORY**

72417       First released in Issue 4. Derived from the MSE working draft.

72418 **Issue 6**

72419       The Open Group Corrigendum U021/11 is applied. The function is marked as an extension.

72420 **NAME**72421        `wcsxfrm`, `wcsxfrm_l` — wide-character string transformation72422 **SYNOPSIS**

```

72423     #include <wchar.h>
72424     size_t wcsxfrm(wchar_t *restrict ws1, const wchar_t *restrict ws2,
72425                   size_t n);
72426 CX    size_t wcsxfrm_l(wchar_t *restrict ws1, const wchar_t *restrict ws2,
72427                      size_t n, locale_t locale);

```

72428 **DESCRIPTION**

72429 CX     For `wcsxfrm()`: The functionality described on this reference page is aligned with the ISO C  
72430 standard. Any conflict between the requirements described here and the ISO C standard is  
72431 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

72432 CX     The `wcsxfrm()` and `wcsxfrm_l()` functions shall transform the wide-character string pointed to  
72433 by `ws2` and place the resulting wide-character string into the array pointed to by `ws1`. The  
72434 transformation shall be such that if `wcscmp()` is applied to two transformed wide strings, it shall  
72435 CX     return a value greater than, equal to, or less than 0, corresponding to the result of `wcscoll()` and  
72436 `wcscoll_l()` applied to the same two original wide-character strings, and the same `LC_COLLATE`  
72437 CX     category of the current locale or the locale object `locale`, respectively. No more than `n` wide-  
72438 character codes shall be placed into the resulting array pointed to by `ws1`, including the  
72439 terminating null wide-character code. If `n` is 0, `ws1` is permitted to be a null pointer. If copying  
72440 takes place between objects that overlap, the behavior is undefined.

72441 CX     The `wcsxfrm()` and `wcsxfrm_l()` functions shall not change the setting of `errno` if successful.

72442     Since no return value is reserved to indicate an error, an application wishing to check for error  
72443 situations should set `errno` to 0, then call `wcsxfrm()` or `wcsxfrm_l()`, then check `errno`.

72444     The behavior is undefined if the `locale` argument to `wcsxfrm_l()` is the special locale object  
72445 `LC_GLOBAL_LOCALE` or is not a valid locale object handle.

72446 **RETURN VALUE**

72447 CX     The `wcsxfrm()` and `wcsxfrm_l()` functions shall return the length of the transformed wide-  
72448 character string (not including the terminating null wide-character code). If the value returned is  
72449 `n` or more, the contents of the array pointed to by `ws1` are unspecified.

72450 CX     On error, the `wcsxfrm()` and `wcsxfrm_l()` functions may set `errno`, but no return value is reserved  
72451 to indicate an error.

72452 **ERRORS**

72453     These functions may fail if:

72454 CX     [EINVAL]     The wide-character string pointed to by `ws2` contains wide-character codes  
72455 outside the domain of the collating sequence.

**EXAMPLES**

None.

**APPLICATION USAGE**

The transformation function is such that two transformed wide-character strings can be ordered by *wscmp()* as appropriate to collating sequence information in the current locale (category *LC\_COLLATE*).

The fact that when *n* is 0 *ws1* is permitted to be a null pointer is useful to determine the size of the *ws1* array prior to making the transformation.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*wscmp()*, *wscoll()*

XBD <**wchar.h**>

**CHANGE HISTORY**

First released in Issue 4. Derived from the MSE working draft.

**Issue 5**

Moved from ENHANCED I18N to BASE and the [ENOSYS] error is removed.

The DESCRIPTION is updated to indicate that *errno* is not changed if the function is successful.

**Issue 6**

In earlier versions, this function was required to return -1 on error.

Extensions beyond the ISO C standard are marked.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the RETURN VALUE and ERRORS sections, the [EINVAL] optional error condition is added if no conversion could be performed.

The *wcsxfrm()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

**Issue 7**

The *wcsxfrm\_l()* function is added from The Open Group Technical Standard, 2006, Extended API Set Part 4.

POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0728 [302], XSH/TC1-2008/0729 [283], XSH/TC1-2008/0730 [283], and XSH/TC1-2008/0731 [302] are applied.

72489 **NAME**

72490 wctob — wide-character to single-byte conversion

72491 **SYNOPSIS**

72492 #include &lt;stdio.h&gt;

72493 #include &lt;wchar.h&gt;

72494 int wctob(wint\_t c);

72495 **DESCRIPTION**

72496 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 72497 conflict between the requirements described here and the ISO C standard is unintentional. This  
 72498 volume of POSIX.1-2008 defers to the ISO C standard.

72499 The *wctob()* function shall determine whether *c* corresponds to a member of the extended  
 72500 character set whose character representation is a single byte when in the initial shift state.

72501 The behavior of this function shall be affected by the *LC\_CTYPE* category of the current locale.

72502 **RETURN VALUE**

72503 The *wctob()* function shall return EOF if *c* does not correspond to a character with length one in  
 72504 the initial shift state. Otherwise, it shall return the single-byte representation of that character as  
 72505 an **unsigned char** converted to **int**.

72506 **ERRORS**

72507 No errors are defined.

72508 **EXAMPLES**

72509 None.

72510 **APPLICATION USAGE**

72511 None.

72512 **RATIONALE**

72513 None.

72514 **FUTURE DIRECTIONS**

72515 None.

72516 **SEE ALSO**72517 *btowc()*

72518 XBD &lt;stdio.h&gt;, &lt;wchar.h&gt;

72519 **CHANGE HISTORY**

72520 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
 72521 (E).

72522 **NAME**

72523 wctomb — convert a wide-character code to a character

72524 **SYNOPSIS**

72525 #include &lt;stdlib.h&gt;

72526 int wctomb(char \*s, wchar\_t wchar);

72527 **DESCRIPTION**

72528 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 72529 conflict between the requirements described here and the ISO C standard is unintentional. This  
 72530 volume of POSIX.1-2008 defers to the ISO C standard.

72531 The *wctomb()* function shall determine the number of bytes needed to represent the character  
 72532 corresponding to the wide-character code whose value is *wchar* (including any change in the  
 72533 shift state). It shall store the character representation (possibly multiple bytes and any special  
 72534 bytes to change shift state) in the array object pointed to by *s* (if *s* is not a null pointer). At most  
 72535 {MB\_CUR\_MAX} bytes shall be stored. If *wchar* is 0, a null byte shall be stored, preceded by any  
 72536 shift sequence needed to restore the initial shift state, and *wctomb()* shall be left in the initial shift  
 72537 state.

72538 CX The behavior of this function is affected by the *LC\_CTYPE* category of the current locale. For a  
 72539 state-dependent encoding, this function shall be placed into its initial state by a call for which its  
 72540 character pointer argument, *s*, is a null pointer. Subsequent calls with *s* as other than a null  
 72541 pointer shall cause the internal state of the function to be altered as necessary. A call with *s* as a  
 72542 null pointer shall cause this function to return a non-zero value if encodings have state  
 72543 dependency, and 0 otherwise. Changing the *LC\_CTYPE* category causes the shift state of this  
 72544 function to be unspecified.

72545 The *wctomb()* function need not be thread-safe.

72546 The implementation shall behave as if no function defined in this volume of POSIX.1-2008 calls  
 72547 *wctomb()*.

72548 **RETURN VALUE**

72549 If *s* is a null pointer, *wctomb()* shall return a non-zero or 0 value, if character encodings,  
 72550 respectively, do or do not have state-dependent encodings. If *s* is not a null pointer, *wctomb()*  
 72551 shall return -1 if the value of *wchar* does not correspond to a valid character, or return the  
 72552 number of bytes that constitute the character corresponding to the value of *wchar*.

72553 In no case shall the value returned be greater than the value of the {MB\_CUR\_MAX} macro.

72554 **ERRORS**72555 The *wctomb()* function shall fail if:

72556 CX [EILSEQ] An invalid wide-character code is detected.

72557 **EXAMPLES**

72558 None.

72559 **APPLICATION USAGE**

72560 None.

72561 **RATIONALE**

72562 None.

72563 **FUTURE DIRECTIONS**

72564 None.

72565 **SEE ALSO**72566 *mblen()*, *mbtowc()*, *mbstowcs()*, *wcstombs()*72567 XBD **<stdlib.h>**72568 **CHANGE HISTORY**

72569 First released in Issue 4. Derived from the ANSI C standard.

72570 **Issue 6**

72571 Extensions beyond the ISO C standard are marked.

72572 A note indicating that this function need not be reentrant is added to the DESCRIPTION.

72573 **Issue 7**

72574 Austin Group Interpretations 1003.1-2001 #156 and #170 are applied.

72575 **NAME**

72576 wctrans, wctrans\_l — define character mapping

72577 **SYNOPSIS**

72578 #include &lt;wctype.h&gt;

72579 wctrans\_t wctrans(const char \*charclass);

72580 CX wctrans\_t wctrans\_l(const char \*charclass, locale\_t locale);

72581 **DESCRIPTION**

72582 CX For *wctrans()*: The functionality described on this reference page is aligned with the ISO C  
 72583 standard. Any conflict between the requirements described here and the ISO C standard is  
 72584 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

72585 CX The *wctrans()* and *wctrans\_l()* functions are defined for valid character mapping names  
 72586 identified in the current locale. The *charclass* is a string identifying a generic character mapping  
 72587 name for which codeset-specific information is required. The following character mapping  
 72588 names are defined in all locales: **tolower** and **toupper**.

72589 These functions shall return a value of type **wctrans\_t**, which can be used as the second  
 72590 CX argument to subsequent calls of *towctrans()* and *towctrans\_l()*.

72591 CX The *wctrans()* and *wctrans\_l()* functions shall determine values of **wctrans\_t** according to the  
 72592 CX rules of the coded character set defined by character mapping information in the current locale  
 72593 or in the locale represented by *locale*, respectively (category *LC\_CTYPE*).

72594 The values returned by *wctrans()* shall be valid until a call to *setlocale()* that modifies the  
 72595 category *LC\_CTYPE*.

72596 CX The values returned by *wctrans\_l()* shall be valid only in calls to *towctrans\_l()* with a locale  
 72597 represented by *locale* with the same *LC\_CTYPE* category value.

72598 The behavior is undefined if the *locale* argument to *wctrans\_l()* is the special locale object  
 72599 *LC\_GLOBAL\_LOCALE* or is not a valid locale object handle.

72600 **RETURN VALUE**

72601 CX The *wctrans()* and *wctrans\_l()* functions shall return 0 and may set *errno* to indicate the error if  
 72602 the given character mapping name is not valid for the current locale (category *LC\_CTYPE*);  
 72603 otherwise, they shall return a non-zero object of type **wctrans\_t** that can be used in calls to  
 72604 CX *towctrans()* and *towctrans\_l()*.

72605 **ERRORS**

72606 These functions may fail if:

72607 CX **[EINVAL]** The character mapping name pointed to by *charclass* is not valid in the current  
 72608 locale.

72609 **EXAMPLES**

72610 None.

72611 **APPLICATION USAGE**

72612 None.

72613 **RATIONALE**

72614 None.



72615 **FUTURE DIRECTIONS**

72616 None.

72617 **SEE ALSO**72618 *towctrans()*

72619 XBD &lt;wctype.h&gt;

72620 **CHANGE HISTORY**

72621 First released in Issue 5. Derived from ISO/IEC 9899:1990/Amendment 1:1995 (E).

72622 **Issue 7**72623 The *wctrans\_l()* function is added from The Open Group Technical Standard, 2006, Extended  
72624 API Set Part 4.72625 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0732 [302], XSH/TC1-2008/0733 [289],  
72626 XSH/TC1-2008/0734 [283], and XSH/TC1-2008/0735 [283] are applied.

72627 **NAME**

72628       wctype, wctype\_l — define character class

72629 **SYNOPSIS**

72630       #include &lt;wctype.h&gt;

72631       wctype\_t wctype(const char \*property);

72632 CX     wctype\_t wctype\_l(const char \*property, locale\_t locale);

72633 **DESCRIPTION**

72634 CX     For *wctype()*: The functionality described on this reference page is aligned with the ISO C  
 72635 standard. Any conflict between the requirements described here and the ISO C standard is  
 72636 unintentional. This volume of POSIX.1-2008 defers to the ISO C standard.

72637 CX     The *wctype()* and *wctype\_l()* functions are defined for valid character class names as defined in  
 72638 CX     the current locale or in the locale represented by *locale*, respectively.

72639     The *property* argument is a string identifying a generic character class for which codeset-specific  
 72640 type information is required. The following character class names shall be defined in all locales:

|       |              |              |               |
|-------|--------------|--------------|---------------|
| 72641 | <b>alnum</b> | <b>digit</b> | <b>punct</b>  |
| 72642 | <b>alpha</b> | <b>graph</b> | <b>space</b>  |
| 72643 | <b>blank</b> | <b>lower</b> | <b>upper</b>  |
| 72644 | <b>cntrl</b> | <b>print</b> | <b>xdigit</b> |

72645     Additional character class names defined in the locale definition file (category *LC\_CTYPE*) can  
 72646 also be specified.

72647     These functions shall return a value of type **wctype\_t**, which can be used as the second  
 72648 CX     argument to subsequent calls of *iswctype()* and *iswctype\_l()*.

72649 CX     The *wctype()* and *wctype\_l()* functions shall determine values of **wctype\_t** according to the  
 72650 CX     rules of the coded character set defined by character type information in the current locale or in  
 72651 the locale represented by *locale*, respectively (category *LC\_CTYPE*).

72652     The values returned by *wctype()* shall be valid until a call to *setlocale()* that modifies the category  
 72653 *LC\_CTYPE*.

72654 CX     The values returned by *wctype\_l()* shall be valid only in calls to *iswctype\_l()* with a locale  
 72655 represented by *locale* with the same *LC\_CTYPE* category value.

72656     The behavior is undefined if the *locale* argument to *wctype\_l()* is the special locale object  
 72657 *LC\_GLOBAL\_LOCALE* or is not a valid locale object handle.

72658 **RETURN VALUE**

72659 CX     The *wctype()* and *wctype\_l()* functions shall return 0 if the given character class name is not  
 72660 valid for the current locale (category *LC\_CTYPE*); otherwise, they shall return an object of type  
 72661 CX     **wctype\_t** that can be used in calls to *iswctype()* and *iswctype\_l()*.

72662 **ERRORS**

72663     No errors are defined.

72664 **EXAMPLES**

72665 None.

72666 **APPLICATION USAGE**

72667 None.

72668 **RATIONALE**

72669 None.

72670 **FUTURE DIRECTIONS**

72671 None.

72672 **SEE ALSO**72673 *iswctype()*

72674 XBD &lt;wctype.h&gt;

72675 **CHANGE HISTORY**

72676 First released in Issue 4.

72677 **Issue 5**72678 The following change has been made in this version for alignment with  
72679 ISO/IEC 9899:1990/Amendment 1:1995 (E):

- 72680 • The SYNOPSIS has been changed to indicate that this function and associated data types  
72681 are now made visible by inclusion of the <wctype.h> header rather than <wchar.h>.

72682 **Issue 7**72683 The *wctype\_l()* function is added from The Open Group Technical Standard, 2006, Extended API  
72684 Set Part 4.72685 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0736 [302], XSH/TC1-2008/0737 [283],  
72686 and XSH/TC1-2008/0738 [283] are applied.

72687 **NAME**

72688       wctype — number of column positions of a wide-character code

72689 **SYNOPSIS**

```
72690 XSI      #include <wctype.h>
72691          int wctype(wchar_t wc);
```

72692 **DESCRIPTION**

72693       The *wctype()* function shall determine the number of column positions required for the wide  
72694       character *wc*. The application shall ensure that the value of *wc* is a character representable as a  
72695       *wchar\_t*, and is a wide-character code corresponding to a valid character in the current locale.

72696 **RETURN VALUE**

72697       The *wctype()* function shall either return 0 (if *wc* is a null wide-character code), or return the  
72698       number of column positions to be occupied by the wide-character code *wc*, or return -1 (if *wc*  
72699       does not correspond to a printable wide-character code).

72700 **ERRORS**

72701       No errors are defined.

72702 **EXAMPLES**

72703       None.

72704 **APPLICATION USAGE**

72705       This function was removed from the final ISO/IEC 9899:1990/Amendment 1:1995 (E), and the  
72706       return value for a non-printable wide character is not specified.

72707 **RATIONALE**

72708       None.

72709 **FUTURE DIRECTIONS**

72710       None.

72711 **SEE ALSO**72712       *wcswidth()*

72713       XBD &lt;wctype.h&gt;

72714 **CHANGE HISTORY**

72715       First released as a World-wide Portability Interface in Issue 4. Derived from the MSE working  
72716       draft.

72717 **Issue 6**

72718       The Open Group Corrigendum U021/12 is applied. This function is marked as an extension.

72719       The normative text is updated to avoid use of the term “must” for application requirements.

72720 **NAME**

72721 wmemchr — find a wide character in memory

72722 **SYNOPSIS**

72723 #include &lt;wchar.h&gt;

72724 wchar\_t \*wmemchr(const wchar\_t \*ws, wchar\_t wc, size\_t n);

72725 **DESCRIPTION**

72726 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 72727 conflict between the requirements described here and the ISO C standard is unintentional. This  
 72728 volume of POSIX.1-2008 defers to the ISO C standard.

72729 The *wmemchr()* function shall locate the first occurrence of *wc* in the initial *n* wide characters of  
 72730 the object pointed to by *ws*. This function shall not be affected by locale and all **wchar\_t** values  
 72731 shall be treated identically. The null wide character and **wchar\_t** values not corresponding to  
 72732 valid characters shall not be treated specially.

72733 If *n* is zero, the application shall ensure that *ws* is a valid pointer and the function behaves as if  
 72734 no valid occurrence of *wc* is found.

72735 **RETURN VALUE**

72736 The *wmemchr()* function shall return a pointer to the located wide character, or a null pointer if  
 72737 the wide character does not occur in the object.

72738 **ERRORS**

72739 No errors are defined.

72740 **EXAMPLES**

72741 None.

72742 **APPLICATION USAGE**

72743 None.

72744 **RATIONALE**

72745 None.

72746 **FUTURE DIRECTIONS**

72747 None.

72748 **SEE ALSO**72749 *wmemcmp()*, *wmemcpy()*, *wmemmove()*, *wmemset()*

72750 XBD &lt;wchar.h&gt;

72751 **CHANGE HISTORY**

72752 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
 72753 (E).

72754 **Issue 6**

72755 The normative text is updated to avoid use of the term “must” for application requirements.

72756 **NAME**

72757 wmemcmp — compare wide characters in memory

72758 **SYNOPSIS**

72759 #include &lt;wchar.h&gt;

72760 int wmemcmp(const wchar\_t \*ws1, const wchar\_t \*ws2, size\_t n);

72761 **DESCRIPTION**

72762 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
72763 conflict between the requirements described here and the ISO C standard is unintentional. This  
72764 volume of POSIX.1-2008 defers to the ISO C standard.

72765 The *wmemcmp()* function shall compare the first *n* wide characters of the object pointed to by  
72766 *ws1* to the first *n* wide characters of the object pointed to by *ws2*. This function shall not be  
72767 affected by locale and all **wchar\_t** values shall be treated identically. The null wide character and  
72768 **wchar\_t** values not corresponding to valid characters shall not be treated specially.

72769 If *n* is zero, the application shall ensure that *ws1* and *ws2* are valid pointers, and the function  
72770 shall behave as if the two objects compare equal.

72771 **RETURN VALUE**

72772 The *wmemcmp()* function shall return an integer greater than, equal to, or less than zero,  
72773 respectively, as the object pointed to by *ws1* is greater than, equal to, or less than the object  
72774 pointed to by *ws2*.

72775 **ERRORS**

72776 No errors are defined.

72777 **EXAMPLES**

72778 None.

72779 **APPLICATION USAGE**

72780 None.

72781 **RATIONALE**

72782 None.

72783 **FUTURE DIRECTIONS**

72784 None.

72785 **SEE ALSO**72786 *wmemchr()*, *wmemcpy()*, *wmemmove()*, *wmemset()*

72787 XBD &lt;wchar.h&gt;

72788 **CHANGE HISTORY**

72789 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
72790 (E).

72791 **Issue 6**

72792 The normative text is updated to avoid use of the term “must” for application requirements.

72793 **NAME**

72794       wmemcpy — copy wide characters in memory

72795 **SYNOPSIS**

72796       #include &lt;wchar.h&gt;

```
72797     wchar_t *wmemcpy(wchar_t *restrict ws1, const wchar_t *restrict ws2,
72798                     size_t n);
```

72799 **DESCRIPTION**

72800 CX     The functionality described on this reference page is aligned with the ISO C standard. Any  
 72801       conflict between the requirements described here and the ISO C standard is unintentional. This  
 72802       volume of POSIX.1-2008 defers to the ISO C standard.

72803       The *wmemcpy()* function shall copy *n* wide characters from the object pointed to by *ws2* to the  
 72804       object pointed to by *ws1*. This function shall not be affected by locale and all **wchar\_t** values  
 72805       shall be treated identically. The null wide character and **wchar\_t** values not corresponding to  
 72806       valid characters shall not be treated specially.

72807       If *n* is zero, the application shall ensure that *ws1* and *ws2* are valid pointers, and the function  
 72808       shall copy zero wide characters.

72809 **RETURN VALUE**72810       The *wmemcpy()* function shall return the value of *ws1*.72811 **ERRORS**

72812       No errors are defined.

72813 **EXAMPLES**

72814       None.

72815 **APPLICATION USAGE**

72816       None.

72817 **RATIONALE**

72818       None.

72819 **FUTURE DIRECTIONS**

72820       None.

72821 **SEE ALSO**72822       *wmemchr()*, *wmemcmp()*, *wmemmove()*, *wmemset()*

72823       XBD &lt;wchar.h&gt;

72824 **CHANGE HISTORY**

72825       First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
 72826       (E).

72827 **Issue 6**

72828       The normative text is updated to avoid use of the term “must” for application requirements.

72829       The *wmemcpy()* prototype is updated for alignment with the ISO/IEC 9899:1999 standard.

72830 **NAME**

72831 `wmemmove` — copy wide characters in memory with overlapping areas

72832 **SYNOPSIS**

72833 `#include <wchar.h>`

72834 `wchar_t *wmemmove(wchar_t *ws1, const wchar_t *ws2, size_t n);`

72835 **DESCRIPTION**

72836 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
72837 conflict between the requirements described here and the ISO C standard is unintentional. This  
72838 volume of POSIX.1-2008 defers to the ISO C standard.

72839 The `wmemmove()` function shall copy *n* wide characters from the object pointed to by *ws2* to the  
72840 object pointed to by *ws1*. Copying shall take place as if the *n* wide characters from the object  
72841 pointed to by *ws2* are first copied into a temporary array of *n* wide characters that does not  
72842 overlap the objects pointed to by *ws1* or *ws2*, and then the *n* wide characters from the temporary  
72843 array are copied into the object pointed to by *ws1*.

72844 This function shall not be affected by locale and all **wchar\_t** values shall be treated identically.  
72845 The null wide character and **wchar\_t** values not corresponding to valid characters shall not be  
72846 treated specially.

72847 If *n* is zero, the application shall ensure that *ws1* and *ws2* are valid pointers, and the function  
72848 shall copy zero wide characters.

72849 **RETURN VALUE**

72850 The `wmemmove()` function shall return the value of *ws1*.

72851 **ERRORS**

72852 No errors are defined

72853 **EXAMPLES**

72854 None.

72855 **APPLICATION USAGE**

72856 None.

72857 **RATIONALE**

72858 None.

72859 **FUTURE DIRECTIONS**

72860 None.

72861 **SEE ALSO**

72862 [\*wmemchr\(\)\*](#), [\*wmemcmp\(\)\*](#), [\*wmemcpy\(\)\*](#), [\*wmemset\(\)\*](#)

72863 XBD [\*\*<wchar.h>\*\*](#)

72864 **CHANGE HISTORY**

72865 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
72866 (E).

72867 **Issue 6**

72868 The normative text is updated to avoid use of the term “must” for application requirements.



72869 **NAME**

72870 wmemset — set wide characters in memory

72871 **SYNOPSIS**

72872 #include &lt;wchar.h&gt;

72873 wchar\_t \*wmemset(wchar\_t \*ws, wchar\_t wc, size\_t n);

72874 **DESCRIPTION**

72875 CX The functionality described on this reference page is aligned with the ISO C standard. Any  
 72876 conflict between the requirements described here and the ISO C standard is unintentional. This  
 72877 volume of POSIX.1-2008 defers to the ISO C standard.

72878 The *wmemset()* function shall copy the value of *wc* into each of the first *n* wide characters of the  
 72879 object pointed to by *ws*. This function shall not be affected by locale and all **wchar\_t** values shall  
 72880 be treated identically. The null wide character and **wchar\_t** values not corresponding to valid  
 72881 characters shall not be treated specially.

72882 If *n* is zero, the application shall ensure that *ws* is a valid pointer, and the function shall copy  
 72883 zero wide characters.

72884 **RETURN VALUE**72885 The *wmemset()* functions shall return the value of *ws*.72886 **ERRORS**

72887 No errors are defined.

72888 **EXAMPLES**

72889 None.

72890 **APPLICATION USAGE**

72891 None.

72892 **RATIONALE**

72893 None.

72894 **FUTURE DIRECTIONS**

72895 None.

72896 **SEE ALSO**72897 *wmemchr()*, *wmemcmp()*, *wmemcpy()*, *wmemmove()*

72898 XBD &lt;wchar.h&gt;

72899 **CHANGE HISTORY**

72900 First released in Issue 5. Included for alignment with ISO/IEC 9899:1990/Amendment 1:1995  
 72901 (E).

72902 **Issue 6**

72903 The normative text is updated to avoid use of the term “must” for application requirements.

**NAME**

wordexp, wordfree — perform word expansions

**SYNOPSIS**

```
#include <wordexp.h>

int wordexp(const char *restrict words, wordexp_t *restrict pwordexp,
            int flags);
void wordfree(wordexp_t *pwordexp);
```

**DESCRIPTION**

The *wordexp()* function shall perform word expansions as described in XCU [Section 2.6](#) (on page 2343), subject to quoting as described in XCU [Section 2.2](#) (on page 2336), and place the list of expanded words into the structure pointed to by *pwordexp*.

The *words* argument is a pointer to a string containing one or more words to be expanded. The expansions shall be the same as would be performed by the command line interpreter if *words* were the part of a command line representing the arguments to a utility. Therefore, the application shall ensure that *words* does not contain an unquoted <newline> character or any of the unquoted shell special characters ' | ', '&', ';', '<', '>' except in the context of command substitution as specified in XCU [Section 2.6.3](#) (on page 2347). It also shall not contain unquoted parentheses or braces, except in the context of command or variable substitution. The application shall ensure that every member of *words* which it expects to have expanded by *wordexp()* does not contain an unquoted initial comment character. The application shall also ensure that any words which it intends to be ignored (because they begin or continue a comment) are deleted from *words*. If the argument *words* contains an unquoted comment character (<number-sign>) that is the beginning of a token, *wordexp()* shall either treat the comment character as a regular character, or interpret it as a comment indicator and ignore the remainder of *words*.

The structure type **wordexp\_t** is defined in the **<wordexp.h>** header and includes at least the following members:

| Member Type | Member Name     | Description                                                         |
|-------------|-----------------|---------------------------------------------------------------------|
| size_t      | <i>we_wordc</i> | Count of words matched by <i>words</i> .                            |
| char **     | <i>we_wordv</i> | Pointer to list of expanded words.                                  |
| size_t      | <i>we_offs</i>  | Slots to reserve at the beginning of <i>pwordexp-&gt;we_wordv</i> . |

The *wordexp()* function shall store the number of generated words into *pwordexp->we\_wordc* and a pointer to a list of pointers to words in *pwordexp->we\_wordv*. Each individual field created during field splitting (see XCU [Section 2.6.5](#), on page 2349) or pathname expansion (see XCU [Section 2.6.6](#), on page 2350) shall be a separate word in the *pwordexp->we\_wordv* list. The words shall be in order as described in XCU [Section 2.6](#) (on page 2343). The first pointer after the last word pointer shall be a null pointer. The expansion of special parameters described in XCU [Section 2.5.2](#) (on page 2340) is unspecified.

It is the caller's responsibility to allocate the storage pointed to by *pwordexp*. The *wordexp()* function shall allocate other space as needed, including memory pointed to by *pwordexp->we\_wordv*. The *wordfree()* function frees any memory associated with *pwordexp* from a previous call to *wordexp()*.

The *flags* argument is used to control the behavior of *wordexp()*. The value of *flags* is the bitwise-inclusive OR of zero or more of the following constants, which are defined in **<wordexp.h>**:

|       |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 72949 | WRDE_APPEND  | Append words generated to the ones from a previous call to <i>wordexp()</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 72950 | WRDE_DOOFFS  | Make use of <i>pwordexp-&gt;we_offs</i> . If this flag is set, <i>pwordexp-&gt;we_offs</i> is used to specify how many null pointers to add to the beginning of <i>pwordexp-&gt;we_wordv</i> . In other words, <i>pwordexp-&gt;we_wordv</i> shall point to <i>pwordexp-&gt;we_offs</i> null pointers, followed by <i>pwordexp-&gt;we_wordc</i> word pointers, followed by a null pointer.                                                                                                                                                                                                                               |
| 72951 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72952 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72953 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72954 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72955 | WRDE_NOCMD   | If the implementation supports the utilities defined in the Shell and Utilities volume of POSIX.1-2008, fail if command substitution, as specified in XCU Section 2.6.3 (on page 2347), is requested.                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 72956 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72957 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72958 | WRDE_REUSE   | The <i>pwordexp</i> argument was passed to a previous successful call to <i>wordexp()</i> , and has not been passed to <i>wordfree()</i> . The result shall be the same as if the application had called <i>wordfree()</i> and then called <i>wordexp()</i> without WRDE_REUSE.                                                                                                                                                                                                                                                                                                                                         |
| 72959 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72960 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72961 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72962 | WRDE_SHOWERR | Do not redirect <i>stderr</i> to <i>/dev/null</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 72963 | WRDE_UNDEF   | Report error on an attempt to expand an undefined shell variable.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 72964 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72965 |              | The WRDE_APPEND flag can be used to append a new set of words to those generated by a previous call to <i>wordexp()</i> . The following rules apply to applications when two or more calls to <i>wordexp()</i> are made with the same value of <i>pwordexp</i> and without intervening calls to <i>wordfree()</i> :                                                                                                                                                                                                                                                                                                     |
| 72966 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72967 |              | 1. The first such call shall not set WRDE_APPEND. All subsequent calls shall set it.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 72968 |              | 2. All of the calls shall set WRDE_DOOFFS, or all shall not set it.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 72969 |              | 3. After the second and each subsequent call, <i>pwordexp-&gt;we_wordv</i> shall point to a list containing the following:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 72970 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72971 |              | a. Zero or more null pointers, as specified by WRDE_DOOFFS and <i>pwordexp-&gt;we_offs</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 72972 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72973 |              | b. Pointers to the words that were in the <i>pwordexp-&gt;we_wordv</i> list before the call, in the same order as before                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 72974 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72975 |              | c. Pointers to the new words generated by the latest call, in the specified order                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 72976 |              | 4. The count returned in <i>pwordexp-&gt;we_wordc</i> shall be the total number of words from all of the calls.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72977 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72978 |              | 5. The application can change any of the fields after a call to <i>wordexp()</i> , but if it does it shall reset them to the original value before a subsequent call, using the same <i>pwordexp</i> value, to <i>wordfree()</i> or <i>wordexp()</i> with the WRDE_APPEND or WRDE_REUSE flag.                                                                                                                                                                                                                                                                                                                           |
| 72979 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72980 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72981 |              | If the implementation supports the utilities defined in the Shell and Utilities volume of POSIX.1-2008, and <i>words</i> contains an unquoted character—<newline>, '   ', '&', ';', '<', '>', '(', ')', '{', '}'—in an inappropriate context, <i>wordexp()</i> shall fail, and the number of expanded words shall be 0.                                                                                                                                                                                                                                                                                                 |
| 72982 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72983 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72984 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72985 |              | Unless WRDE_SHOWERR is set in <i>flags</i> , <i>wordexp()</i> shall redirect <i>stderr</i> to <i>/dev/null</i> for any utilities executed as a result of command substitution while expanding <i>words</i> . If WRDE_SHOWERR is set, <i>wordexp()</i> may write messages to <i>stderr</i> if syntax errors are detected while expanding <i>words</i> , unless the <i>stderr</i> stream has wide orientation in which case the behavior is undefined. It is unspecified whether any write errors encountered while outputting such messages will affect the <i>stderr</i> error indicator or the value of <i>errno</i> . |
| 72986 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72987 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72988 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72989 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72990 |              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 72991 |              | The application shall ensure that if WRDE_DOOFFS is set, then <i>pwordexp-&gt;we_offs</i> has the same                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

72992 value for each *wordexp()* call and *wordfree()* call using a given *pwordexp*.

72993 The results are unspecified if WRDE\_APPEND and WRDE\_REUSE are both specified. +

72994 The following constants are defined as error return values:

72995 WRDE\_BADCHAR One of the unquoted characters—<newline>, ' | ', '&', ';', '<', '>',  
72996 ' ( ', ' ) ', ' { ', ' } '—appears in *words* in an inappropriate context.

72997 WRDE\_BADVAL Reference to undefined shell variable when WRDE\_UNDEF is set in *flags*.

72998 WRDE\_CMDSUB Command substitution requested when WRDE\_NOCMD was set in *flags*.

72999 WRDE\_NOSPACE Attempt to allocate memory failed.

73000 WRDE\_SYNTAX Shell syntax error, such as unbalanced parentheses or unterminated  
73001 string.

## 73002 RETURN VALUE

73003 Upon successful completion, *wordexp()* shall return 0. Otherwise, a non-zero value, as described  
73004 in <**wordexp.h**>, shall be returned to indicate an error. If *wordexp()* returns the value  
73005 WRDE\_NOSPACE, then *pwordexp->we\_wordc* and *pwordexp->we\_wordv* shall be updated to  
73006 reflect any words that were successfully expanded. In other error cases, if the WRDE\_APPEND  
73007 flag was specified, *pwordexp->we\_wordc* and *pwordexp->we\_wordv* shall not be modified. |

73008 The *wordfree()* function shall not return a value.

## 73009 ERRORS

73010 No errors are defined.

## 73011 EXAMPLES

73012 None.

## 73013 APPLICATION USAGE

73014 The *wordexp()* function is intended to be used by an application that wants to do all of the shell's  
73015 expansions on a word or words obtained from a user. For example, if the application prompts  
73016 for a pathname (or list of pathnames) and then uses *wordexp()* to process the input, the user  
73017 could respond with anything that would be valid as input to the shell.

73018 The WRDE\_NOCMD flag is provided for applications that, for security or other reasons, want to  
73019 prevent a user from executing shell commands. Disallowing unquoted shell special characters  
73020 also prevents unwanted side-effects, such as executing a command or writing a file.

73021 POSIX.1-2008 does not require the *wordexp()* function to be thread-safe if passed an expression  
73022 referencing an environment variable while any other thread is concurrently modifying any  
73023 environment variable; see *exec* (on page 781).

73024 Even though the WRDE\_SHOWERR flag allows the implementation to write messages to *stderr*  
73025 during command substitution or syntax errors, this standard does not provide any way to detect  
73026 write failures during the output of such messages.

73027 Applications which use wide-character output functions with *stderr* should ensure that any calls +  
73028 to *wordexp()* do not write to *stderr*, by avoiding use of the WRDE\_SHOWERR flag.

## 73029 RATIONALE

73030 This function was included as an alternative to *glob()*. There had been continuing controversy  
73031 over exactly what features should be included in *glob()*. It is hoped that by providing *wordexp()*  
73032 (which provides all of the shell word expansions, but which may be slow to execute) and *glob()*  
73033 (which is faster, but which only performs pathname expansion, without tilde or parameter  
73034 expansion) this will satisfy the majority of applications.

While *wordexp()* could be implemented entirely as a library routine, it is expected that most implementations run a shell in a subprocess to do the expansion.

Two different approaches have been proposed for how the required information might be presented to the shell and the results returned. They are presented here as examples.

One proposal is to extend the *echo* utility by adding a *-q* option. This option would cause *echo* to add a *<backslash>* before each *<backslash>* and *<blank>* that occurs within an argument. The *wordexp()* function could then invoke the shell as follows:

```
(void) strcpy(buffer, "echo -q");
(void) strcat(buffer, words);
if ((flags & WRDE_SHOWERR) == 0)
    (void) strcat(buffer, "2>/dev/null");
f = popen(buffer, "r");
```

The *wordexp()* function would read the resulting output, remove unquoted *<backslash>* characters, and break into words at unquoted *<blank>* characters. If the *WRDE\_NOCMD* flag was set, *wordexp()* would have to scan *words* before starting the subshell to make sure that there would be no command substitution. In any case, it would have to scan *words* for unquoted special characters.

Another proposal is to add the following options to *sh*:

**-w** *wordlist*

This option provides a wordlist expansion service to applications. The words in *wordlist* shall be expanded and the following written to standard output:

1. The count of the number of words after expansion, in decimal, followed by a null byte
2. The number of bytes needed to represent the expanded words (not including null separators), in decimal, followed by a null byte
3. The expanded words, each terminated by a null byte

If an error is encountered during word expansion, *sh* exits with a non-zero status after writing the former to report any words successfully expanded

**-P** Run in “protected” mode. If specified with the *-w* option, no command substitution shall be performed.

With these options, *wordexp()* could be implemented fairly simply by creating a subprocess using *fork()* and executing *sh* using the line:

```
execl(<shell path>, "sh", "-P", "-w", words, (char *)0);
```

after directing standard error to */dev/null*.

It seemed objectionable for a library routine to write messages to standard error, unless explicitly requested, so *wordexp()* is required to redirect standard error to */dev/null* to ensure that no messages are generated, even for commands executed for command substitution. The *WRDE\_SHOWERR* flag can be specified to request that error messages be written.

The *WRDE\_REUSE* flag allows the implementation to avoid the expense of freeing and reallocating memory, if that is possible. A minimal implementation can call *wordfree()* when *WRDE\_REUSE* is set.

73076 **FUTURE DIRECTIONS**

73077 None.

73078 **SEE ALSO**73079 *exec*, *fnmatch()*, *glob()*

73080 XBD &lt;wordexp.h&gt;

73081 XCU Chapter 2 (on page 2335)

73082 **CHANGE HISTORY**

73083 First released in Issue 4. Derived from the ISO POSIX-2 standard.

73084 **Issue 5**

73085 Moved from POSIX2 C-language Binding to BASE.

73086 **Issue 6**

73087 The normative text is updated to avoid use of the term “must” for application requirements.

73088 The **restrict** keyword is added to the *wordexp()* prototype for alignment with the  
73089 ISO/IEC 9899:1999 standard.73090 **Issue 7**

73091 Austin Group Interpretation 1003.1-2001 #148 is applied, adding APPLICATION USAGE.

73092 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0739 [460], XSH/TC1-2008/0740 [291],  
73093 and XSH/TC1-2008/0741 [460] are applied.

73094 **NAME**

73095       wprintf — print formatted wide-character output

73096 **SYNOPSIS**

73097       #include &lt;stdio.h&gt;

73098       #include &lt;wchar.h&gt;

73099       int wprintf(const wchar\_t \*restrict *format*, ...);73100 **DESCRIPTION**73101       Refer to *fwprintf()*.



## 73102 NAME

73103 pwrite, write — write on a file

## 73104 SYNOPSIS

```
73105 #include <unistd.h>
73106 ssize_t pwrite(int fildes, const void *buf, size_t nbyte,
73107               off_t offset);
73108 ssize_t write(int fildes, const void *buf, size_t nbyte);
```

## 73109 DESCRIPTION

73110 The *write()* function shall attempt to write *nbyte* bytes from the buffer pointed to by *buf* to the  
73111 file associated with the open file descriptor, *fildes*.

73112 Before any action described below is taken, and if *nbyte* is zero and the file is a regular file, the  
73113 *write()* function may detect and return errors as described below. In the absence of errors, or if  
73114 error detection is not performed, the *write()* function shall return zero and have no other results.  
73115 If *nbyte* is zero and the file is not a regular file, the results are unspecified.

73116 On a regular file or other file capable of seeking, the actual writing of data shall proceed from  
73117 the position in the file indicated by the file offset associated with *fildes*. Before successful return  
73118 from *write()*, the file offset shall be incremented by the number of bytes actually written. On a  
73119 regular file, if the position of the last byte written is greater than or equal to the length of the file,  
73120 the length of the file shall be set to this position plus one.

73121 On a file not capable of seeking, writing shall always take place starting at the current position.  
73122 The value of a file offset associated with such a device is undefined.

73123 If the O\_APPEND flag of the file status flags is set, the file offset shall be set to the end of the file  
73124 prior to each write and no intervening file modification operation shall occur between changing  
73125 the file offset and the write operation.

73126 XSI If a *write()* requests that more bytes be written than there is room for (for example, the file size  
73127 limit of the process or the physical end of a medium), only as many bytes as there is room for  
73128 shall be written. For example, suppose there is space for 20 bytes more in a file before reaching a  
73129 limit. A write of 512 bytes will return 20. The next write of a non-zero number of bytes would  
73130 give a failure return (except as noted below).

73131 XSI If the request would cause the file size to exceed the soft file size limit for the process and there  
73132 is no room for any bytes to be written, the request shall fail and the implementation shall  
73133 generate the SIGXFSZ signal for the thread.

73134 If *write()* is interrupted by a signal before it writes any data, it shall return -1 with *errno* set to  
73135 [EINTR].

73136 If *write()* is interrupted by a signal after it successfully writes some data, it shall return the  
73137 number of bytes written.

73138 If the value of *nbyte* is greater than {SSIZE\_MAX}, the result is implementation-defined.

73139 After a *write()* to a regular file has successfully returned:

- 73140 • Any successful *read()* from each byte position in the file that was modified by that write  
73141 shall return the data specified by the *write()* for that position until such byte positions are  
73142 again modified.
- 73143 • Any subsequent successful *write()* to the same byte position in the file shall overwrite that  
73144 file data.



73145 Write requests to a pipe or FIFO shall be handled in the same way as a regular file with the  
73146 following exceptions:

- 73147 • There is no file offset associated with a pipe, hence each write request shall append to the  
73148 end of the pipe.
- 73149 • Write requests of {PIPE\_BUF} bytes or less shall not be interleaved with data from other  
73150 processes doing writes on the same pipe. Writes of greater than {PIPE\_BUF} bytes may  
73151 have data interleaved, on arbitrary boundaries, with writes by other processes, whether or  
73152 not the O\_NONBLOCK flag of the file status flags is set.
- 73153 • If the O\_NONBLOCK flag is clear, a write request may cause the thread to block, but on  
73154 normal completion it shall return *nbyte*.
- 73155 • If the O\_NONBLOCK flag is set, *write()* requests shall be handled differently, in the  
73156 following ways:
  - 73157 — The *write()* function shall not block the thread.
  - 73158 — A write request for {PIPE\_BUF} or fewer bytes shall have the following effect: if there  
73159 is sufficient space available in the pipe, *write()* shall transfer all the data and return  
73160 the number of bytes requested. Otherwise, *write()* shall transfer no data and return  
73161  $-1$  with *errno* set to [EAGAIN].
  - 73162 — A write request for more than {PIPE\_BUF} bytes shall cause one of the following:
    - 73163 — When at least one byte can be written, transfer what it can and return the  
73164 number of bytes written. When all data previously written to the pipe is read, it  
73165 shall transfer at least {PIPE\_BUF} bytes.
    - 73166 — When no data can be written, transfer no data, and return  $-1$  with *errno* set to  
73167 [EAGAIN].

73168 When attempting to write to a file descriptor (other than a pipe or FIFO) that supports non-  
73169 blocking writes and cannot accept the data immediately:

- 73170 • If the O\_NONBLOCK flag is clear, *write()* shall block the calling thread until the data can  
73171 be accepted.
- 73172 • If the O\_NONBLOCK flag is set, *write()* shall not block the thread. If some data can be  
73173 written without blocking the thread, *write()* shall write what it can and return the number  
73174 of bytes written. Otherwise, it shall return  $-1$  and set *errno* to [EAGAIN].

73175 Upon successful completion, where *nbyte* is greater than 0, *write()* shall mark for update the last  
73176 data modification and last file status change timestamps of the file, and if the file is a regular file,  
73177 the S\_ISUID and S\_ISGID bits of the file mode may be cleared.

73178 For regular files, no data transfer shall occur past the offset maximum established in the open  
73179 file description associated with *filides*.

73180 If *filides* refers to a socket, *write()* shall be equivalent to *send()* with no flags set.

73181 SIO If the O\_DSYNC bit has been set, write I/O operations on the file descriptor shall complete as  
73182 defined by synchronized I/O data integrity completion.

73183 If the O\_SYNC bit has been set, write I/O operations on the file descriptor shall complete as  
73184 defined by synchronized I/O file integrity completion.

73185 SHM If *filides* refers to a shared memory object, the result of the *write()* function is unspecified.

|          |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|------------------------------------------------------------------------------|---------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 73186    | TYM                                                                                                                                                                                     | If <i>filides</i> refers to a typed memory object, the result of the <i>write()</i> function is unspecified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73187    | OB XSR                                                                                                                                                                                  | <p>If <i>filides</i> refers to a STREAM, the operation of <i>write()</i> shall be determined by the values of the minimum and maximum <i>nbyte</i> range (packet size) accepted by the STREAM. These values are determined by the topmost STREAM module. If <i>nbyte</i> falls within the packet size range, <i>nbyte</i> bytes shall be written. If <i>nbyte</i> does not fall within the range and the minimum packet size value is 0, <i>write()</i> shall break the buffer into maximum packet size segments prior to sending the data downstream (the last segment may contain less than the maximum packet size). If <i>nbyte</i> does not fall within the range and the minimum value is non-zero, <i>write()</i> shall fail with <i>errno</i> set to [ERANGE]. Writing a zero-length buffer (<i>nbyte</i> is 0) to a STREAMS device sends 0 bytes with 0 returned. However, writing a zero-length buffer to a STREAMS-based pipe or FIFO sends no message and 0 is returned. The process may issue <code>I_SWROPT ioctl()</code> to enable zero-length messages to be sent across the pipe or FIFO.</p> <p>When writing to a STREAM, data messages are created with a priority band of 0. When writing to a STREAM that is not a pipe or FIFO:</p> <ul style="list-style-type: none"><li>• If <code>O_NONBLOCK</code> is clear, and the STREAM cannot accept data (the STREAM write queue is full due to internal flow control conditions), <i>write()</i> shall block until data can be accepted.</li><li>• If <code>O_NONBLOCK</code> is set and the STREAM cannot accept data, <i>write()</i> shall return <code>-1</code> and set <i>errno</i> to [EAGAIN].</li><li>• If <code>O_NONBLOCK</code> is set and part of the buffer has been written while a condition in which the STREAM cannot accept additional data occurs, <i>write()</i> shall terminate and return the number of bytes written.</li></ul> <p>In addition, <i>write()</i> shall fail if the STREAM head has processed an asynchronous error before the call. In this case, the value of <i>errno</i> does not reflect the result of <i>write()</i>, but reflects the prior error.</p> <p>The <i>pwrite()</i> function shall be equivalent to <i>write()</i>, except that it writes into a given position and does not change the file offset (regardless of whether <code>O_APPEND</code> is set). The first three arguments to <i>pwrite()</i> are the same as <i>write()</i> with the addition of a fourth argument <i>offset</i> for the desired position inside the file. An attempt to perform a <i>pwrite()</i> on a file that is incapable of seeking shall result in an error.</p> |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73188    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73189    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73190    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73191    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73192    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73193    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73194    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73195    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73196    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73197    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73198    |                                                                                                                                                                                         | <p><b>RETURN VALUE</b></p> <p>Upon successful completion, these functions shall return the number of bytes actually written to the file associated with <i>filides</i>. This number shall never be greater than <i>nbyte</i>. Otherwise, <code>-1</code> shall be returned and <i>errno</i> set to indicate the error.</p> <p><b>ERRORS</b></p> <p>These functions shall fail if:</p> <table><tr><td>[EAGAIN]</td><td>The file is neither a pipe, nor a FIFO, nor a socket, the <code>O_NONBLOCK</code> flag is set for the file descriptor, and the thread would be delayed in the <i>write()</i> operation.</td></tr><tr><td>[EBADF]</td><td>The <i>filides</i> argument is not a valid file descriptor open for writing.</td></tr><tr><td>[EFBIG]</td><td>An attempt was made to write a file that exceeds the implementation-defined maximum file size or the file size limit of the process, and there was no room for any bytes to be written.</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | [EAGAIN] | The file is neither a pipe, nor a FIFO, nor a socket, the <code>O_NONBLOCK</code> flag is set for the file descriptor, and the thread would be delayed in the <i>write()</i> operation. | [EBADF] | The <i>filides</i> argument is not a valid file descriptor open for writing. | [EFBIG] | An attempt was made to write a file that exceeds the implementation-defined maximum file size or the file size limit of the process, and there was no room for any bytes to be written. |
| [EAGAIN] | The file is neither a pipe, nor a FIFO, nor a socket, the <code>O_NONBLOCK</code> flag is set for the file descriptor, and the thread would be delayed in the <i>write()</i> operation. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| [EBADF]  | The <i>filides</i> argument is not a valid file descriptor open for writing.                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| [EFBIG]  | An attempt was made to write a file that exceeds the implementation-defined maximum file size or the file size limit of the process, and there was no room for any bytes to be written. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73199    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73200    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73201    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73202    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73203    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73204    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73205    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73206    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73207    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73208    |                                                                                                                                                                                         | <p><b>RETURN VALUE</b></p> <p>Upon successful completion, these functions shall return the number of bytes actually written to the file associated with <i>filides</i>. This number shall never be greater than <i>nbyte</i>. Otherwise, <code>-1</code> shall be returned and <i>errno</i> set to indicate the error.</p> <p><b>ERRORS</b></p> <p>These functions shall fail if:</p> <table><tr><td>[EAGAIN]</td><td>The file is neither a pipe, nor a FIFO, nor a socket, the <code>O_NONBLOCK</code> flag is set for the file descriptor, and the thread would be delayed in the <i>write()</i> operation.</td></tr><tr><td>[EBADF]</td><td>The <i>filides</i> argument is not a valid file descriptor open for writing.</td></tr><tr><td>[EFBIG]</td><td>An attempt was made to write a file that exceeds the implementation-defined maximum file size or the file size limit of the process, and there was no room for any bytes to be written.</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | [EAGAIN] | The file is neither a pipe, nor a FIFO, nor a socket, the <code>O_NONBLOCK</code> flag is set for the file descriptor, and the thread would be delayed in the <i>write()</i> operation. | [EBADF] | The <i>filides</i> argument is not a valid file descriptor open for writing. | [EFBIG] | An attempt was made to write a file that exceeds the implementation-defined maximum file size or the file size limit of the process, and there was no room for any bytes to be written. |
| [EAGAIN] | The file is neither a pipe, nor a FIFO, nor a socket, the <code>O_NONBLOCK</code> flag is set for the file descriptor, and the thread would be delayed in the <i>write()</i> operation. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| [EBADF]  | The <i>filides</i> argument is not a valid file descriptor open for writing.                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| [EFBIG]  | An attempt was made to write a file that exceeds the implementation-defined maximum file size or the file size limit of the process, and there was no room for any bytes to be written. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73209    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73210    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73211    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73212    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73213    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73214    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73215    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73216    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73217    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73218    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73219    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73220    |                                                                                                                                                                                         | <p><b>RETURN VALUE</b></p> <p>Upon successful completion, these functions shall return the number of bytes actually written to the file associated with <i>filides</i>. This number shall never be greater than <i>nbyte</i>. Otherwise, <code>-1</code> shall be returned and <i>errno</i> set to indicate the error.</p> <p><b>ERRORS</b></p> <p>These functions shall fail if:</p> <table><tr><td>[EAGAIN]</td><td>The file is neither a pipe, nor a FIFO, nor a socket, the <code>O_NONBLOCK</code> flag is set for the file descriptor, and the thread would be delayed in the <i>write()</i> operation.</td></tr><tr><td>[EBADF]</td><td>The <i>filides</i> argument is not a valid file descriptor open for writing.</td></tr><tr><td>[EFBIG]</td><td>An attempt was made to write a file that exceeds the implementation-defined maximum file size or the file size limit of the process, and there was no room for any bytes to be written.</td></tr></table>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | [EAGAIN] | The file is neither a pipe, nor a FIFO, nor a socket, the <code>O_NONBLOCK</code> flag is set for the file descriptor, and the thread would be delayed in the <i>write()</i> operation. | [EBADF] | The <i>filides</i> argument is not a valid file descriptor open for writing. | [EFBIG] | An attempt was made to write a file that exceeds the implementation-defined maximum file size or the file size limit of the process, and there was no room for any bytes to be written. |
| [EAGAIN] | The file is neither a pipe, nor a FIFO, nor a socket, the <code>O_NONBLOCK</code> flag is set for the file descriptor, and the thread would be delayed in the <i>write()</i> operation. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| [EBADF]  | The <i>filides</i> argument is not a valid file descriptor open for writing.                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| [EFBIG]  | An attempt was made to write a file that exceeds the implementation-defined maximum file size or the file size limit of the process, and there was no room for any bytes to be written. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73221    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73222    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73223    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73224    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73225    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73226    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73227    | XSI                                                                                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |
| 73228    |                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |          |                                                                                                                                                                                         |         |                                                                              |         |                                                                                                                                                                                         |

|       |                           |                                                                                                                                                                                                                                                                                                                                   |
|-------|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 73229 | [EFBIG]                   | The file is a regular file, <i>nbyte</i> is greater than 0, and the starting position is greater than or equal to the offset maximum established in the open file description associated with <i>fildest</i> .                                                                                                                    |
| 73230 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73231 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73232 | [EINTR]                   | The write operation was terminated due to the receipt of a signal, and no data was transferred.                                                                                                                                                                                                                                   |
| 73233 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73234 | [EIO]                     | The process is a member of a background process group attempting to write to its controlling terminal, TOSTOP is set, the calling thread is not blocking SIGTTOU, the process is not ignoring SIGTTOU, and the process group of the process is orphaned. This error may also be returned under implementation-defined conditions. |
| 73235 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73236 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73237 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73238 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73239 | [ENOSPC]                  | There was no free space remaining on the device containing the file.                                                                                                                                                                                                                                                              |
| 73240 | OB XSR [ERANGE]           | The transfer request size was outside the range supported by the STREAMS file associated with <i>fildest</i> .                                                                                                                                                                                                                    |
| 73241 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73242 |                           | The <i>pwrite()</i> function shall fail if:                                                                                                                                                                                                                                                                                       |
| 73243 | [EINVAL]                  | The file is a regular file or block special file, and the <i>offset</i> argument is negative. The file offset shall remain unchanged.                                                                                                                                                                                             |
| 73244 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73245 | [ESPIPE]                  | The file is incapable of seeking.                                                                                                                                                                                                                                                                                                 |
| 73246 |                           | The <i>write()</i> function shall fail if:                                                                                                                                                                                                                                                                                        |
| 73247 | [EAGAIN]                  | The file is a pipe or FIFO, the O_NONBLOCK flag is set for the file descriptor, and the thread would be delayed in the write operation.                                                                                                                                                                                           |
| 73248 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73249 | [EAGAIN] or [EWOULDBLOCK] |                                                                                                                                                                                                                                                                                                                                   |
| 73250 |                           | The file is a socket, the O_NONBLOCK flag is set for the file descriptor, and the thread would be delayed in the write operation.                                                                                                                                                                                                 |
| 73251 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73252 | [ECONNRESET]              | A write was attempted on a socket that is not connected.                                                                                                                                                                                                                                                                          |
| 73253 | [EPIPE]                   | An attempt is made to write to a pipe or FIFO that is not open for reading by any process, or that only has one end open. A SIGPIPE signal shall also be sent to the thread.                                                                                                                                                      |
| 73254 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73255 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73256 | [EPIPE]                   | A write was attempted on a socket that is shut down for writing, or is no longer connected. In the latter case, if the socket is of type SOCK_STREAM, a SIGPIPE signal shall also be sent to the thread.                                                                                                                          |
| 73257 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73258 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73259 |                           | These functions may fail if:                                                                                                                                                                                                                                                                                                      |
| 73260 | OB XSR [EINVAL]           | The STREAM or multiplexer referenced by <i>fildest</i> is linked (directly or indirectly) downstream from a multiplexer.                                                                                                                                                                                                          |
| 73261 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73262 | [EIO]                     | A physical I/O error has occurred.                                                                                                                                                                                                                                                                                                |
| 73263 | [ENOBUFFS]                | Insufficient resources were available in the system to perform the operation.                                                                                                                                                                                                                                                     |
| 73264 | [ENXIO]                   | A request was made of a nonexistent device, or the request was outside the capabilities of the device.                                                                                                                                                                                                                            |
| 73265 |                           |                                                                                                                                                                                                                                                                                                                                   |
| 73266 | OB XSR [ENXIO]            | A hangup occurred on the STREAM being written to.                                                                                                                                                                                                                                                                                 |
| 73267 | OB XSR                    | A write to a STREAMS file may fail if an error message has been received at the STREAM head.                                                                                                                                                                                                                                      |
| 73268 |                           | In this case, <i>errno</i> is set to the value included in the error message.                                                                                                                                                                                                                                                     |

- 73269 The *write()* function may fail if:
- 73270 [EACCES] A write was attempted on a socket and the calling process does not have  
73271 appropriate privileges.
- 73272 [ENETDOWN] A write was attempted on a socket and the local network interface used to  
73273 reach the destination is down.
- 73274 [ENETUNREACH]  
73275 A write was attempted on a socket and no route to the network is present.

## 73276 EXAMPLES

### 73277 Writing from a Buffer

73278 The following example writes data from the buffer pointed to by *buf* to the file associated with  
73279 the file descriptor *fd*.

```
73280 #include <sys/types.h>
73281 #include <string.h>
73282 ...
73283 char buf[20];
73284 size_t nbytes;
73285 ssize_t bytes_written;
73286 int fd;
73287 ...
73288 strcpy(buf, "This is a test\n");
73289 nbytes = strlen(buf);
73290 bytes_written = write(fd, buf, nbytes);
73291 ...
```

## 73292 APPLICATION USAGE

73293 None.

## 73294 RATIONALE

73295 See also the RATIONALE section in *read()*.

73296 An attempt to write to a pipe or FIFO has several major characteristics:

- 73297 • *Atomic/non-atomic*: A write is atomic if the whole amount written in one operation is not  
73298 interleaved with data from any other process. This is useful when there are multiple  
73299 writers sending data to a single reader. Applications need to know how large a write  
73300 request can be expected to be performed atomically. This maximum is called {PIPE\_BUF}.  
73301 This volume of POSIX.1-2008 does not say whether write requests for more than  
73302 {PIPE\_BUF} bytes are atomic, but requires that writes of {PIPE\_BUF} or fewer bytes shall  
73303 be atomic.
- 73304 • *Blocking/immediate*: Blocking is only possible with O\_NONBLOCK clear. If there is enough  
73305 space for all the data requested to be written immediately, the implementation should do  
73306 so. Otherwise, the calling thread may block; that is, pause until enough space is available  
73307 for writing. The effective size of a pipe or FIFO (the maximum amount that can be written  
73308 in one operation without blocking) may vary dynamically, depending on the  
73309 implementation, so it is not possible to specify a fixed value for it.
- 73310 • *Complete/partial/deferred*: A write request:  
73311 

```
int fildes;
```

  
73312 

```
size_t nbyte;
```

```

73313     ssize_t ret;
73314     char *buf;

73315     ret = write(fildes, buf, nbyte);

```

73316 may return:

```

73317     Complete    ret=nbyte
73318     Partial     ret<nbyte

```

73319 This shall never happen if  $nbyte \leq \{PIPE\_BUF\}$ . If it does happen (with  
73320  $nbyte > \{PIPE\_BUF\}$ ), this volume of POSIX.1-2008 does not guarantee  
73321 atomicity, even if  $ret \leq \{PIPE\_BUF\}$ , because atomicity is guaranteed according  
73322 to the amount *requested*, not the amount *written*.

```

73323     Deferred:   ret=-1, errno=[EAGAIN]

```

73324 This error indicates that a later request may succeed. It does not indicate that  
73325 it *shall* succeed, even if  $nbyte \leq \{PIPE\_BUF\}$ , because if no process reads from  
73326 the pipe or FIFO, the write never succeeds. An application could usefully  
73327 count the number of times [EAGAIN] is caused by a particular value of  
73328  $nbyte > \{PIPE\_BUF\}$  and perhaps do later writes with a smaller value, on the  
73329 assumption that the effective size of the pipe may have decreased.

73330 Partial and deferred writes are only possible with O\_NONBLOCK set.

73331 The relations of these properties are shown in the following tables:

| Write to a Pipe or FIFO with O_NONBLOCK clear |                                 |                                 |                                  |
|-----------------------------------------------|---------------------------------|---------------------------------|----------------------------------|
| Immediately Writable:                         | None                            | Some                            | <i>nbyte</i>                     |
| $nbyte \leq \{PIPE\_BUF\}$                    | Atomic blocking<br><i>nbyte</i> | Atomic blocking<br><i>nbyte</i> | Atomic immediate<br><i>nbyte</i> |
| $nbyte > \{PIPE\_BUF\}$                       | Blocking <i>nbyte</i>           | Blocking <i>nbyte</i>           | Blocking <i>nbyte</i>            |

73337 If the O\_NONBLOCK flag is clear, a write request shall block if the amount writable  
73338 immediately is less than that requested. If the flag is set (by *fcntl()*), a write request shall never  
73339 block.

| Write to a Pipe or FIFO with O_NONBLOCK set |              |                                   |                                 |
|---------------------------------------------|--------------|-----------------------------------|---------------------------------|
| Immediately Writable:                       | None         | Some                              | <i>nbyte</i>                    |
| $nbyte \leq \{PIPE\_BUF\}$                  | -1, [EAGAIN] | -1, [EAGAIN]                      | Atomic <i>nbyte</i>             |
| $nbyte > \{PIPE\_BUF\}$                     | -1, [EAGAIN] | < <i>nbyte</i> or -1,<br>[EAGAIN] | $\leq nbyte$ or -1,<br>[EAGAIN] |

73345 There is no exception regarding partial writes when O\_NONBLOCK is set. With the exception  
73346 of writing to an empty pipe, this volume of POSIX.1-2008 does not specify exactly when a partial  
73347 write is performed since that would require specifying internal details of the implementation.  
73348 Every application should be prepared to handle partial writes when O\_NONBLOCK is set and  
73349 the requested amount is greater than {PIPE\_BUF}, just as every application should be prepared  
73350 to handle partial writes on other kinds of file descriptors.

73351 The intent of forcing writing at least one byte if any can be written is to assure that each write  
73352 makes progress if there is any room in the pipe. If the pipe is empty, {PIPE\_BUF} bytes must be  
73353 written; if not, at least some progress must have been made.

73354 Where this volume of POSIX.1-2008 requires -1 to be returned and *errno* set to [EAGAIN], most

historical implementations return zero (with the `O_NDELAY` flag set, which is the historical predecessor of `O_NONBLOCK`, but is not itself in this volume of POSIX.1-2008). The error indications in this volume of POSIX.1-2008 were chosen so that an application can distinguish these cases from end-of-file. While `write()` cannot receive an indication of end-of-file, `read()` can, and the two functions have similar return values. Also, some existing systems (for example, Eighth Edition) permit a write of zero bytes to mean that the reader should get an end-of-file indication; for those systems, a return value of zero from `write()` indicates a successful write of an end-of-file indication.

Implementations are allowed, but not required, to perform error checking for `write()` requests of zero bytes.

The concept of a `{PIPE_MAX}` limit (indicating the maximum number of bytes that can be written to a pipe in a single operation) was considered, but rejected, because this concept would unnecessarily limit application writing.

See also the discussion of `O_NONBLOCK` in `read()`.

Writes can be serialized with respect to other reads and writes. If a `read()` of file data can be proven (by any means) to occur after a `write()` of the data, it must reflect that `write()`, even if the calls are made by different processes. A similar requirement applies to multiple write operations to the same file position. This is needed to guarantee the propagation of data from `write()` calls to subsequent `read()` calls. This requirement is particularly significant for networked file systems, where some caching schemes violate these semantics.

Note that this is specified in terms of `read()` and `write()`. The XSI extensions `readv()` and `writew()` also obey these semantics. A new “high-performance” write analog that did not follow these serialization requirements would also be permitted by this wording. This volume of POSIX.1-2008 is also silent about any effects of application-level caching (such as that done by `stdio`).

This volume of POSIX.1-2008 does not specify the value of the file offset after an error is returned; there are too many cases. For programming errors, such as `[EBADF]`, the concept is meaningless since no file is involved. For errors that are detected immediately, such as `[EAGAIN]`, clearly the pointer should not change. After an interrupt or hardware error, however, an updated value would be very useful and is the behavior of many implementations.

This volume of POSIX.1-2008 does not specify the behavior of concurrent writes to a regular file from multiple threads, except that each write is atomic (see [Section 2.9.7](#), on page 522). Applications should use some form of concurrency control.

This volume of POSIX.1-2008 intentionally does not specify any `pwrite()` errors related to pipes, FIFOs, and sockets other than `[ESPIPE]`.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[`chmod\(\)`](#), [`creat\(\)`](#), [`dup\(\)`](#), [`fcntl\(\)`](#), [`getrlimit\(\)`](#), [`lseek\(\)`](#), [`open\(\)`](#), [`pipe\(\)`](#), [`read\(\)`](#), [`ulimit\(\)`](#), [`writew\(\)`](#)

XBD [`<limits.h>`](#), [`<stropts.h>`](#), [`<sys/uio.h>`](#), [`<unistd.h>`](#)

## CHANGE HISTORY

First released in Issue 1. Derived from Issue 1 of the SVID.



**Issue 5**

The DESCRIPTION is updated for alignment with the POSIX Realtime Extension and the POSIX Threads Extension.

Large File Summit extensions are added.

The *pwrite()* function is added.

**Issue 6**

The DESCRIPTION states that the *write()* function does not block the thread. Previously this said “process” rather than “thread”.

The DESCRIPTION and ERRORS sections are updated so that references to STREAMS are marked as part of the XSI STREAMS Option Group.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The DESCRIPTION now states that if *write()* is interrupted by a signal after it has successfully written some data, it returns the number of bytes written. In the POSIX.1-1988 standard, it was optional whether *write()* returned the number of bytes written, or whether it returned -1 with *errno* set to [EINTR]. This is a FIPS requirement.
- The following changes are made to support large files:
  - For regular files, no data transfer occurs past the offset maximum established in the open file description associated with the *files*.
  - A second [EFBIG] error condition is added.
- The [EIO] error condition is added.
- The [EPIPE] error condition is added for when a pipe has only one end open.
- The [ENXIO] optional error condition is added.

Text referring to sockets is added to the DESCRIPTION.

The following changes were made to align with the IEEE P1003.1a draft standard:

- The effect of reading zero bytes is clarified.

The DESCRIPTION is updated for alignment with IEEE Std 1003.1j-2000 by specifying that *write()* results are unspecified for typed memory objects.

The following error conditions are added for operations on sockets: [EAGAIN], [EWOULDBLOCK], [ECONNRESET], [ENOTCONN], and [EPIPE].

The [EIO] error is made optional.

The [ENOBUFFS] error is added for sockets.

The following error conditions are added for operations on sockets: [EACCES], [ENETDOWN], and [ENETUNREACH].

The *writetv()* function is split out into a separate reference page.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/146 is applied, updating text in the ERRORS section from “a SIGPIPE signal is generated to the calling process” to “a SIGPIPE signal shall also be sent to the thread”.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/147 is applied, making a correction to the RATIONALE.

73437 **Issue 7**

73438 The *pwrite()* function is moved from the XSI option to the Base.

73439 Functionality relating to the XSI STREAMS option is marked obsolescent.

73440 SD5-XSH-ERN-160 is applied, updating the DESCRIPTION to clarify the requirements for the  
73441 *pwrite()* function, and to change the use of the phrase “file pointer” to “file offset”.

73442 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0742 [219], XSH/TC1-2008/0743 [215],  
73443 XSH/TC1-2008/0744 [79], and XSH/TC1-2008/0745 [215] are applied.



73444 **NAME**

73445 writev — write a vector

73446 **SYNOPSIS**

```
73447 XSI    #include <sys/uio.h>
73448      ssize_t writev(int fildes, const struct iovec *iov, int iovcnt);
```

73449 **DESCRIPTION**

73450 The *writev()* function shall be equivalent to *write()*, except as described below. The *writev()*  
 73451 function shall gather output data from the *iovcnt* buffers specified by the members of the *iov*  
 73452 array: *iov*[0], *iov*[1], ..., *iov*[*iovcnt*−1]. The *iovcnt* argument is valid if greater than 0 and less than  
 73453 or equal to {IOV\_MAX}, as defined in <limits.h>.

73454 Each *iovec* entry specifies the base address and length of an area in memory from which data  
 73455 should be written. The *writev()* function shall always write a complete area before proceeding to  
 73456 the next.

73457 If *fildes* refers to a regular file and all of the *iov\_len* members in the array pointed to by *iov* are 0,  
 73458 *writev()* shall return 0 and have no other effect. For other file types, the behavior is unspecified.

73459 If the sum of the *iov\_len* values is greater than {SSIZE\_MAX}, the operation shall fail and no data  
 73460 shall be transferred.

73461 **RETURN VALUE**

73462 Upon successful completion, *writev()* shall return the number of bytes actually written.  
 73463 Otherwise, it shall return a value of −1, the file-pointer shall remain unchanged, and *errno* shall  
 73464 be set to indicate an error.

73465 **ERRORS**73466 Refer to *write()*.73467 In addition, the *writev()* function shall fail if:73468 [EINVAL] The sum of the *iov\_len* values in the *iov* array would overflow an *ssize\_t*.73469 The *writev()* function may fail and set *errno* to:73470 [EINVAL] The *iovcnt* argument was less than or equal to 0, or greater than {IOV\_MAX}.73471 **EXAMPLES**73472 **Writing Data from an Array**

73473 The following example writes data from the buffers specified by members of the *iov* array to the  
 73474 file associated with the file descriptor *fd*.

```
73475 #include <sys/types.h>
73476 #include <sys/uio.h>
73477 #include <unistd.h>
73478 ...
73479 ssize_t bytes_written;
73480 int fd;
73481 char *buf0 = "short string\n";
73482 char *buf1 = "This is a longer string\n";
73483 char *buf2 = "This is the longest string in this example\n";
73484 int iocnt;
73485 struct iovec iov[3];
```

```
73486     iov[0].iov_base = buf0;
73487     iov[0].iov_len = strlen(buf0);
73488     iov[1].iov_base = buf1;
73489     iov[1].iov_len = strlen(buf1);
73490     iov[2].iov_base = buf2;
73491     iov[2].iov_len = strlen(buf2);
73492     ...
73493     iovcnt = sizeof(iov) / sizeof(struct iovec);
73494     bytes_written = writev(fd, iov, iovcnt);
73495     ...
```

#### 73496 APPLICATION USAGE

73497 None.

#### 73498 RATIONALE

73499 Refer to *write()*.

#### 73500 FUTURE DIRECTIONS

73501 None.

#### 73502 SEE ALSO

73503 *readv()*, *write()*

73504 XBD [\*<limits.h>\*](#), [\*<sys/uio.h>\*](#)

#### 73505 CHANGE HISTORY

73506 First released in Issue 4, Version 2.

#### 73507 Issue 6

73508 Split out from the *write()* reference page.

73509 **NAME**

73510       wscanf — convert formatted wide-character input

73511 **SYNOPSIS**

73512       #include &lt;stdio.h&gt;

73513       #include &lt;wchar.h&gt;

73514       int wscanf(const wchar\_t \*restrict *format*, ...);73515 **DESCRIPTION**73516       Refer to *fwscanf()*.

73517 **NAME**

73518 y0, y1, yn — Bessel functions of the second kind

73519 **SYNOPSIS**

```
73520 XSI      #include <math.h>
73521          double y0(double x);
73522          double y1(double x);
73523          double yn(int n, double x);
```

73524 **DESCRIPTION**

73525 The *y0()*, *y1()*, and *yn()* functions shall compute Bessel functions of *x* of the second kind of  
 73526 orders 0, 1, and *n*, respectively.

73527 An application wishing to check for error situations should set *errno* to zero and call  
 73528 *feclearexcept*(FE\_ALL\_EXCEPT) before calling these functions. On return, if *errno* is non-zero or  
 73529 *fetestexcept*(FE\_INVALID | FE\_DIVBYZERO | FE\_OVERFLOW | FE\_UNDERFLOW) is non-  
 73530 zero, an error has occurred.

73531 **RETURN VALUE**

73532 Upon successful completion, these functions shall return the relevant Bessel value of *x* of the  
 73533 second kind.

73534 MXX If *x* is NaN, NaN shall be returned.

73535 If the *x* argument to these functions is negative, *-HUGE\_VAL* or NaN shall be returned, and a  
 73536 domain error may occur.

73537 If *x* is 0.0, *-HUGE\_VAL* shall be returned and a pole error may occur.

73538 If the correct result would cause underflow, 0.0 shall be returned and a range error may occur.

73539 If the correct result would cause overflow, *-HUGE\_VAL* or 0.0 shall be returned and a range  
 73540 error may occur.

73541 **ERRORS**

73542 These functions may fail if:

73543 Domain Error     The value of *x* is negative.

73544 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 73545 then *errno* shall be set to [EDOM]. If the integer expression (*math\_errhandling*  
 73546 & MATH\_ERREXCEPT) is non-zero, then the invalid floating-point exception  
 73547 shall be raised.

73548 Pole Error       The value of *x* is zero.

73549 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 73550 then *errno* shall be set to [ERANGE]. If the integer expression  
 73551 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the divide-by-zero  
 73552 floating-point exception shall be raised.

73553 Range Error      The correct result would cause overflow.

73554 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
 73555 then *errno* shall be set to [ERANGE]. If the integer expression  
 73556 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the overflow  
 73557 floating-point exception shall be raised.

73558 Range Error The value of  $x$  is too large in magnitude, or the correct result would cause  
73559 underflow.

73560 If the integer expression (*math\_errhandling* & MATH\_ERRNO) is non-zero,  
73561 then *errno* shall be set to [ERANGE]. If the integer expression  
73562 (*math\_errhandling* & MATH\_ERREXCEPT) is non-zero, then the underflow  
73563 floating-point exception shall be raised.

#### 73564 EXAMPLES

73565 None.

#### 73566 APPLICATION USAGE

73567 On error, the expressions (*math\_errhandling* & MATH\_ERRNO) and (*math\_errhandling* &  
73568 MATH\_ERREXCEPT) are independent of each other, but at least one of them must be non-zero.

#### 73569 RATIONALE

73570 None.

#### 73571 FUTURE DIRECTIONS

73572 None.

#### 73573 SEE ALSO

73574 *feclearexcept()*, *fetestexcept()*, *isnan()*, *j0()*

73575 XBD Section 4.20 (on page 117), <math.h>

#### 73576 CHANGE HISTORY

73577 First released in Issue 1. Derived from Issue 1 of the SVID.

#### 73578 Issue 5

73579 The DESCRIPTION is updated to indicate how an application should check for an error. This  
73580 text was previously published in the APPLICATION USAGE section.

#### 73581 Issue 6

73582 The normative text is updated to avoid use of the term “must” for application requirements.

73583 The RETURN VALUE and ERRORS sections are reworked for alignment of the error handling  
73584 with the ISO/IEC 9899:1999 standard.

73585 IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/148 is applied, updating the RETURN  
73586 VALUE and ERRORS sections. The changes are made for consistency with the general rules  
73587 stated in “Treatment of Error Conditions for Mathematical Functions” in the Base Definitions  
73588 volume of POSIX.1-2008.

#### 73589 Issue 7

73590 POSIX.1-2008, Technical Corrigendum 1, XSH/TC1-2008/0746 [68] is applied.



73591



73592

**Vol. 3:**

73593

**Shell and Utilities, Issue 7**

73594

*The Open Group*

73595

*The Institute of Electrical and Electronics Engineers, Inc.*





73596

Chapter 1

73597

# Introduction

73598

73599

The Shell and Utilities volume of POSIX.1-2008 describes the commands and utilities offered to application programs by POSIX-conformant systems.

73600

## 1.1 Relationship to Other Documents

73601

### 1.1.1 System Interfaces

73602

73603

73604

73605

73606

73607

This subsection describes some of the features provided by the System Interfaces volume of POSIX.1-2008 that are assumed to be globally available on all systems conforming to this volume of POSIX.1-2008. This subsection does not attempt to detail all of the features defined in the System Interfaces volume of POSIX.1-2008 that are required by all of the utilities defined in this volume of POSIX.1-2008; the utility and function descriptions point out additional functionality required to provide the corresponding specific features needed by each.

73608

73609

73610

The following subsections describe frequently used concepts. Many of these concepts are described in the Base Definitions volume of POSIX.1-2008. Utility and function description statements override these defaults when appropriate.

73611

#### 1.1.1.1 Process Attributes

73612

73613

The following process attributes, as described in the System Interfaces volume of POSIX.1-2008, are assumed to be supported for all processes in this volume of POSIX.1-2008:

73614

73615

73616

73617

73618

73619

73620

73621

|                           |                         |
|---------------------------|-------------------------|
| Controlling Terminal      | Real Group ID           |
| Current Working Directory | Real User ID            |
| Effective Group ID        | Root Directory          |
| Effective User ID         | Saved Set-Group-ID      |
| File Descriptors          | Saved Set-User-ID       |
| File Mode Creation Mask   | Session Membership      |
| Process Group ID          | Supplementary Group IDs |
| Process ID                |                         |

73622

A conforming implementation may include additional process attributes.

73623

#### 1.1.1.2 Concurrent Execution of Processes

73624

73625

The following functionality of the *fork()* function defined in the System Interfaces volume of POSIX.1-2008 shall be available on all systems conforming to this volume of POSIX.1-2008:

73626

73627

1. Independent processes shall be capable of executing independently without either process terminating.

73628 2. A process shall be able to create a new process with all of the attributes referenced in  
 73629 [Section 1.1.1.1](#) (on page 2317), determined according to the semantics of a call to the *fork()*  
 73630 function defined in the System Interfaces volume of POSIX.1-2008 followed by a call in  
 73631 the child process to one of the *exec* functions defined in the System Interfaces volume of  
 73632 POSIX.1-2008.

#### 73633 1.1.1.3 File Access Permissions

73634 The file access control mechanism described by XBD [Section 4.5](#) (on page 108) shall apply to all  
 73635 files on an implementation conforming to this volume of POSIX.1-2008.

#### 73636 1.1.1.4 File Read, Write, and Creation

73637 If a file that does not exist is to be written, it shall be created as described below, unless the  
 73638 utility description states otherwise.

73639 When a file that does not exist is created, the following features defined in the System Interfaces  
 73640 volume of POSIX.1-2008 shall apply unless the utility or function description states otherwise:

- 73641 1. The user ID of the file shall be set to the effective user ID of the calling process.
- 73642 2. The group ID of the file shall be set to the effective group ID of the calling process or the  
 73643 group ID of the directory in which the file is being created.

- 73644 3. If the file is a regular file, the permission bits of the file shall be set to:

73645 S\_IROTH | S\_IWOTH | S\_IRGRP | S\_IWGRP | S\_IRUSR | S\_IWUSR

73646 (see the description of *File Modes* in XBD [Chapter 13](#) (on page 219), `<sys/stat.h>`) except  
 73647 that the bits specified by the file mode creation mask of the process shall be cleared. If the  
 73648 file is a directory, the permission bits shall be set to:

73649 S\_IRWXU | S\_IRWXG | S\_IRWXO

73650 except that the bits specified by the file mode creation mask of the process shall be  
 73651 cleared.

- 73652 4. The last data access, last data modification, and last file status change timestamps of the  
 73653 file shall be updated as specified in XBD [Section 4.9](#) (on page 109).

- 73654 5. If the file is a directory, it shall be an empty directory; otherwise, the file shall have length  
 73655 zero.

- 73656 6. If the file is a symbolic link, the effect shall be undefined unless the {POSIX2\_SYMLINKS}  
 73657 variable is in effect for the directory in which the symbolic link would be created.

- 73658 7. Unless otherwise specified, the file created shall be a regular file.

73659 When an attempt is made to create a file that already exists, the utility shall take the action  
 73660 indicated in [Table 1-1](#) (on page 2319) corresponding to the type of the file the utility is trying to  
 73661 create and the type of the existing file, unless the utility description states otherwise.

**Table 1-1** Actions when Creating a File that Already Exists

| Existing Type                 | New Type |   |   |   |   |   |   |   |    |   |   | Function Creating New |
|-------------------------------|----------|---|---|---|---|---|---|---|----|---|---|-----------------------|
|                               | B        | C | D | F | L | M | P | Q | R  | S | T |                       |
| A <i>fattach()</i> -ed STREAM | F        | F | F | F | F | — | — | — | OF | — | U | N/A                   |
| B Block Special               | F        | F | F | F | F | U | U | U | OF | U | U | <i>mknod()</i> **     |
| C Character Special           | F        | F | F | F | F | U | U | U | OF | U | U | <i>mknod()</i> **     |
| D Directory                   | F        | F | F | F | F | — | — | — | F  | — | U | <i>mkdir()</i>        |
| F FIFO Special File           | F        | F | F | F | F | — | — | — | O  | — | U | <i>mkfifo()</i>       |
| L Symbolic Link               | F        | F | F | F | F | — | — | — | FL | — | U | <i>symlink()</i>      |
| M Shared Memory               | F        | F | F | F | F | — | — | — | —  | — | U | <i>shm_open()</i>     |
| P Semaphore                   | F        | F | F | F | F | — | — | — | —  | — | U | <i>sem_open()</i>     |
| Q Message Queue               | F        | F | F | F | F | — | — | — | —  | — | U | <i>mq_open()</i>      |
| R Regular File                | F        | F | F | F | F | — | — | — | RF | — | U | <i>open()</i>         |
| S Socket                      | F        | F | F | F | F | — | — | — | —  | — | U | <i>bind()</i>         |
| T Typed Memory                | F        | F | F | F | F | U | U | U | U  | U | U | *                     |

The following codes are used in [Table 1-1](#):

**F** Fail. The attempt to create the new file shall fail and the utility shall either continue with its operation or exit immediately with a non-zero exit status, depending on the description of the utility.

**FL** Follow link. Unless otherwise specified, the symbolic link shall be followed as specified for pathname resolution, and the operation performed shall be as if the target of the symbolic link (after all resolution) had been named. If the target of the symbolic link does not exist, it shall be as if that nonexistent target had been named directly.

**O** Open FIFO. When attempting to create a regular file, and the existing file is a FIFO special file:

1. If the FIFO is not already open for reading, the attempt shall block until the FIFO is opened for reading.
2. Once the FIFO is open for reading, the utility shall open the FIFO for writing and continue with its operation.

**OF** The named file shall be opened with the consequences defined for that file type.

**RF** Regular file. When attempting to create a regular file, and the existing file is a regular file:

1. The user ID, group ID, and permission bits of the file shall not be changed.
2. The file shall be truncated to zero length.
3. The last data modification and last file status change timestamps shall be marked for update.

**—** The effect is implementation-defined unless specified by the utility description.

**U** The effect is unspecified unless specified by the utility description.

**\*** There is no portable way to create a file of this type.

**\*\*** Not portable.

When a file is to be appended, the file shall be opened in a manner equivalent to using the `O_APPEND` flag, without the `O_TRUNC` flag, in the *open()* function defined in the System Interfaces volume of POSIX.1-2008.

73704 When a file is to be read or written, the file shall be opened with an access mode corresponding  
 73705 to the operation to be performed. If file access permissions deny access, the requested operation  
 73706 shall fail.

#### 73707 1.1.1.5 File Removal

73708 When a directory that is the root directory or current working directory of any process is  
 73709 removed, the effect is implementation-defined. If file access permissions deny access, the  
 73710 requested operation shall fail. Otherwise, when a file is removed:

- 73711 1. Its directory entry shall be removed from the file system.
- 73712 2. The link count of the file shall be decremented.
- 73713 3. If the file is an empty directory (see XBD [Section 3.144](#), on page 56):
  - 73714 a. If no process has the directory open, the space occupied by the directory shall be  
 73715 freed and the directory shall no longer be accessible.
  - 73716 b. If one or more processes have the directory open, the directory contents shall be  
 73717 preserved until all references to the file have been closed.
- 73718 4. If the file is a directory that is not empty, the last file status change timestamp shall be  
 73719 marked for update.
- 73720 5. If the file is not a directory:
  - 73721 a. If the link count becomes zero:
    - 73722 i. If no process has the file open, the space occupied by the file shall be freed  
 73723 and the file shall no longer be accessible.
    - 73724 ii. If one or more processes have the file open, the file contents shall be  
 73725 preserved until all references to the file have been closed.
  - 73726 b. If the link count is not reduced to zero, the last file status change timestamp shall  
 73727 be marked for update.
- 73728 6. The last data modification and last file status change timestamps of the containing  
 73729 directory shall be marked for update.

#### 73730 1.1.1.6 File Time Values

73731 All files shall have the three time values described by XBD [Section 4.9](#) (on page 109).

#### 73732 1.1.1.7 File Contents

73733 When a reference is made to the contents of a file, *pathname*, this means the equivalent of all of  
 73734 the data placed in the space pointed to by *buf* when performing the *read()* function calls in the  
 73735 following operations defined in the System Interfaces volume of POSIX.1-2008:

```
73736 while (read (fildes, buf, nbytes) > 0)
73737     ;
```

73738 If the file is indicated by a pathname *pathname*, the file descriptor shall be determined by the  
 73739 equivalent of the following operation defined in the System Interfaces volume of POSIX.1-2008:

```
73740 fildes = open (pathname, O_RDONLY);
```

73741 The value of *nbytes* in the above sequence is unspecified; if the file is of a type where the data

73742 returned by *read()* would vary with different values, the value shall be one that results in the  
73743 most data being returned.

73744 If the *read()* function calls would return an error, it is unspecified whether the contents of the file  
73745 are considered to include any data from offsets in the file beyond where the error would be  
73746 returned.

#### 73747 1.1.1.8 Pathname Resolution

73748 The pathname resolution algorithm, described by XBD [Section 4.13](#) (on page 111), shall be used  
73749 by implementations conforming to this volume of POSIX.1-2008; see also XBD [Section 4.6](#) (on  
73750 page 109).

#### 73751 1.1.1.9 Changing the Current Working Directory

73752 When the current working directory (see XBD [Section 3.122](#), on page 53) is to be changed, unless  
73753 the utility or function description states otherwise, the operation shall succeed unless a call to  
73754 the *chdir()* function defined in the System Interfaces volume of POSIX.1-2008 would fail when  
73755 invoked with the new working directory pathname as its argument.

#### 73756 1.1.1.10 Establish the Locale

73757 The functionality of the *setlocale()* function defined in the System Interfaces volume of  
73758 POSIX.1-2008 shall be available on all systems conforming to this volume of POSIX.1-2008; that  
73759 is, utilities that require the capability of establishing an international operating environment  
73760 shall be permitted to set the specified category of the international environment.

#### 73761 1.1.1.11 Actions Equivalent to Functions

73762 Some utility descriptions specify that a utility performs actions equivalent to a function defined  
73763 in the System Interfaces volume of POSIX.1-2008. Such specifications require only that the  
73764 external effects be equivalent, not that any effect within the utility and visible only to the utility  
73765 be equivalent.

### 73766 1.1.2 Concepts Derived from the ISO C Standard

73767 Some of the standard utilities perform complex data manipulation using their own procedure  
73768 and arithmetic languages, as defined in their EXTENDED DESCRIPTION or OPERANDS  
73769 sections. Unless otherwise noted, the arithmetic and semantic concepts (precision, type  
73770 conversion, control flow, and so on) shall be equivalent to those defined in the ISO C standard,  
73771 as described in the following sections. Note that there is no requirement that the standard  
73772 utilities be implemented in any particular programming language.

#### 73773 1.1.2.1 Arithmetic Precision and Operations

73774 Integer variables and constants, including the values of operands and option-arguments, used  
73775 by the standard utilities listed in this volume of POSIX.1-2008 shall be implemented as  
73776 equivalent to the ISO C standard **signed long** data type; floating point shall be implemented as  
73777 equivalent to the ISO C standard **double** type. Conversions between types shall be as described  
73778 in the ISO C standard. All variables shall be initialized to zero if they are not otherwise assigned  
73779 by the input to the application.

Arithmetic operators and control flow keywords shall be implemented as equivalent to those in the cited ISO C standard section, as listed in [Table 1-2](#).

**Table 1-2** Selected ISO C Standard Operators and Control Flow Keywords

| Operation                                                          | ISO C Standard Equivalent Reference           |
|--------------------------------------------------------------------|-----------------------------------------------|
| ()                                                                 | Section 6.5.1, Primary Expressions            |
| postfix ++<br>postfix --                                           | Section 6.5.2, Postfix Operators              |
| unary +<br>unary -<br>prefix ++<br>prefix --<br>~<br>!<br>sizeof() | Section 6.5.3, Unary Operators                |
| *<br>/<br>%                                                        | Section 6.5.5, Multiplicative Operators       |
| +<br>-                                                             | Section 6.5.6, Additive Operators             |
| <<<br>>>                                                           | Section 6.5.7, Bitwise Shift Operators        |
| <, <=<br>>, >=                                                     | Section 6.5.8, Relational Operators           |
| ==<br>!=                                                           | Section 6.5.9, Equality Operators             |
| &                                                                  | Section 6.5.10, Bitwise AND Operator          |
| ^                                                                  | Section 6.5.11, Bitwise Exclusive OR Operator |
|                                                                    | Section 6.5.12, Bitwise Inclusive OR Operator |
| &&                                                                 | Section 6.5.13, Logical AND Operator          |
|                                                                    | Section 6.5.14, Logical OR Operator           |
| expr?expr:expr                                                     | Section 6.5.15, Conditional Operator          |
| =, *=, /=, %=, +=, -=<br><<=, >>=, &=, ^=,  =                      | Section 6.5.16, Assignment Operators          |
| if ()<br>if () ... else<br>switch ()                               | Section 6.8.4, Selection Statements           |
| while ()<br>do ... while ()<br>for ()                              | Section 6.8.5, Iteration Statements           |
| goto<br>continue<br>break<br>return                                | Section 6.8.6, Jump Statements                |

73823 The evaluation of arithmetic expressions shall be equivalent to that described in Section 6.5,  
73824 Expressions, of the ISO C standard.

#### 73825 1.1.2.2 Mathematical Functions

73826 Any mathematical functions with the same names as those in the following sections of the ISO C  
73827 standard:

- 73828 • Section 7.12, Mathematics, <math.h>
- 73829 • Section 7.20.2, Pseudo-Random Sequence Generation Functions

73830 shall be implemented to return the results equivalent to those returned from a call to the  
73831 corresponding function described in the ISO C standard.

## 73832 1.2 Utility Limits

73833 This section lists magnitude limitations imposed by a specific implementation. The braces  
73834 notation, {LIMIT}, is used in this volume of POSIX.1-2008 to indicate these values, but the braces  
73835 are not part of the name.

73836 **Table 1-3** Utility Limit Minimum Values

| Name                      | Description                                                                                                                                                                                                                                | Value |
|---------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| {POSIX2_BC_BASE_MAX}      | The maximum <i>obase</i> value allowed by the <i>bc</i> utility.                                                                                                                                                                           | 99    |
| {POSIX2_BC_DIM_MAX}       | The maximum number of elements permitted in an array by the <i>bc</i> utility.                                                                                                                                                             | 2 048 |
| {POSIX2_BC_SCALE_MAX}     | The maximum <i>scale</i> value allowed by the <i>bc</i> utility.                                                                                                                                                                           | 99    |
| {POSIX2_BC_STRING_MAX}    | The maximum length of a string constant accepted by the <i>bc</i> utility.                                                                                                                                                                 | 1 000 |
| {POSIX2_COLL_WEIGHTS_MAX} | The maximum number of weights that can be assigned to an entry of the <i>LC_COLLATE</i> <b>order</b> keyword in the locale definition file; see the <b>border_start</b> keyword in XBD <a href="#">Section 7.3.2</a> (on page 147).        | 2     |
| {POSIX2_EXPR_NEST_MAX}    | The maximum number of expressions that can be nested within parentheses by the <i>expr</i> utility.                                                                                                                                        | 32    |
| {POSIX2_LINE_MAX}         | Unless otherwise noted, the maximum length, in bytes, of the input line of a utility (either standard input or another file), when the utility is described as processing text files. The length includes room for the trailing <newline>. | 2 048 |
| {POSIX_RE_DUP_MAX}        | Maximum number of repeated occurrences of a BRE or ERE interval expression; see XBD <a href="#">Section 9.3.6</a> (on page 186) and <a href="#">Section 9.4.6</a> (on page 190).                                                           | 255   |

73862 The values specified in [Table 1-3](#) represent the lowest values conforming implementations shall  
73863 provide and, consequently, the largest values on which an application can rely without further



enquiries, as described below. These values shall be accessible to applications via the *getconf* utility (see *getconf*, on page 2820).

Implementations may provide more liberal, or less restrictive, values than shown in [Table 1-3](#) (on page 2323). These possibly more liberal values are accessible using the symbols in [Table 1-4](#).

The *sysconf()* function defined in the System Interfaces volume of POSIX.1-2008 or the *getconf* utility return the value of each symbol on each specific implementation. The value so retrieved is the largest, or most liberal, value that is available throughout the session lifetime, as determined at session creation. The literal names shown in the table apply only to the *getconf* utility; the high-level language binding describes the exact form of each name to be used by the interfaces in that binding.

All numeric limits defined by the System Interfaces volume of POSIX.1-2008, such as {PATH\_MAX}, shall also apply to this volume of POSIX.1-2008. All the utilities defined by this volume of POSIX.1-2008 are implicitly limited by these values, unless otherwise noted in the utility descriptions.

It is not guaranteed that the application can actually reach the specified limit of an implementation in any given case, or at all, as a lack of virtual memory or other resources may prevent this. The limit value indicates only that the implementation does not specifically impose any arbitrary, more restrictive limit.

**Table 1-4** Symbolic Utility Limits

| Name               | Description                                                                                                                                                                                                                        | Minimum Value             |
|--------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|
| {BC_BASE_MAX}      | The maximum <i>obase</i> value allowed by the <i>bc</i> utility.                                                                                                                                                                   | {POSIX2_BC_BASE_MAX}      |
| {BC_DIM_MAX}       | The maximum number of elements permitted in an array by the <i>bc</i> utility.                                                                                                                                                     | {POSIX2_BC_DIM_MAX}       |
| {BC_SCALE_MAX}     | The maximum <i>scale</i> value allowed by the <i>bc</i> utility.                                                                                                                                                                   | {POSIX2_BC_SCALE_MAX}     |
| {BC_STRING_MAX}    | The maximum length of a string constant accepted by the <i>bc</i> utility.                                                                                                                                                         | {POSIX2_BC_STRING_MAX}    |
| {COLL_WEIGHTS_MAX} | The maximum number of weights that can be assigned to an entry of the <i>LC_COLLATE</i> <b>order</b> keyword in the locale definition file; see the <b>order_start</b> keyword in XBD <a href="#">Section 7.3.2</a> (on page 147). | {POSIX2_COLL_WEIGHTS_MAX} |
| {EXPR_NEST_MAX}    | The maximum number of expressions that can be nested within parentheses by the <i>expr</i> utility.                                                                                                                                | {POSIX2_EXPR_NEST_MAX}    |



| Name         | Description                                                                                                                                                                                                                                | Minimum Value      |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------|
| {LINE_MAX}   | Unless otherwise noted, the maximum length, in bytes, of the input line of a utility (either standard input or another file), when the utility is described as processing text files. The length includes room for the trailing <newline>. | {POSIX2_LINE_MAX}  |
| {RE_DUP_MAX} | Maximum number of repeated occurrences of a BRE or ERE interval expression; see XBD <a href="#">Section 9.3.6</a> (on page 186) and <a href="#">Section 9.4.6</a> (on page 190).                                                           | {POSIX_RE_DUP_MAX} |

The following value may be a constant within an implementation or may vary from one pathname to another.

{POSIX2\_SYMLINKS}

When referring to a directory, the system supports the creation of symbolic links within that directory; for non-directory files, the meaning of {POSIX2\_SYMLINKS} is undefined.

## 1.3 Grammar Conventions

Portions of this volume of POSIX.1-2008 are expressed in terms of a special grammar notation. It is used to portray the complex syntax of certain program input. The grammar is based on the syntax used by the *yacc* utility. However, it does not represent fully functional *yacc* input, suitable for program use; the lexical processing and all semantic requirements are described only in textual form. The grammar is not based on source used in any traditional implementation and has not been tested with the semantic code that would normally be required to accompany it. Furthermore, there is no implication that the partial *yacc* code presented represents the most efficient, or only, means of supporting the complex syntax within the utility. Implementations may use other programming languages or algorithms, as long as the syntax supported is the same as that represented by the grammar.

The following typographical conventions are used in the grammar; they have no significance except to aid in reading.

- The identifiers for the reserved words of the language are shown with a leading capital letter. (These are terminals in the grammar; for example, **While**, **Case**.)
- The identifiers for terminals in the grammar are all named with uppercase letters and underscores; for example, **NEWLINE**, **ASSIGN\_OP**, **NAME**.
- The identifiers for non-terminals are all lowercase.

## 1.4 Utility Description Defaults

This section describes all of the subsections used within the utility descriptions, including:

- Intended usage of the section
- Global defaults that affect all the standard utilities
- The meanings of notations used in this volume of POSIX.1-2008 that are specific to individual utility sections

### NAME

This section gives the name or names of the utility and briefly states its purpose.

### SYNOPSIS

The SYNOPSIS section summarizes the syntax of the calling sequence for the utility, including options, option-arguments, and operands. Standards for utility naming are described in XBD [Section 12.2](#) (on page 216); for describing the utility's arguments in XBD [Section 12.1](#) (on page 213).

### DESCRIPTION

The DESCRIPTION section describes the actions of the utility. If the utility has a very complex set of subcommands or its own procedural language, an EXTENDED DESCRIPTION section is also provided. Most explanations of optional functionality are omitted here, as they are usually explained in the OPTIONS section.

As stated in [Section 1.1.1.11](#) (on page 2321), some functions are described in terms of equivalent functionality. When specific functions are cited, the implementation shall provide equivalent functionality including side-effects associated with successful execution of the function. The treatment of errors and intermediate results from the individual functions cited is generally not specified by this volume of POSIX.1-2008. See the utility's EXIT STATUS and CONSEQUENCES OF ERRORS sections for all actions associated with errors encountered by the utility.

### OPTIONS

The OPTIONS section describes the utility options and option-arguments, and how they modify the actions of the utility. Standard utilities that have options either fully comply with XBD [Section 12.2](#) (on page 216) or describe all deviations. Apparent disagreements between functionality descriptions in the OPTIONS and DESCRIPTION (or EXTENDED DESCRIPTION) sections are always resolved in favor of the OPTIONS section.

Each OPTIONS section that uses the phrase "The ... utility shall conform to the Utility Syntax Guidelines ..." refers only to the use of the utility as specified by this volume of POSIX.1-2008; implementation extensions should also conform to the guidelines, but may allow exceptions for historical practice.

Unless otherwise stated in the utility description, when given an option unrecognized by the implementation, or when a required option-argument is not provided, standard utilities shall issue a diagnostic message to standard error and exit with a non-zero exit status.

All utilities in this volume of POSIX.1-2008 shall be capable of processing arguments using eight-bit transparency.

**Default Behavior:** When this section is listed as "None.", it means that the implementation need not support any options. Standard utilities that do not accept options, but that do accept operands, shall recognize "--" as a first argument to be discarded.

The requirement for recognizing "--" is because conforming applications need a way to shield their operands from any arbitrary options that the implementation may provide as an extension. For example, if the standard utility *foo* is listed as taking no options, and the application needed to give it a pathname with a leading <hyphen-minus>, it could safely do it as:

```
foo -- -myfile
```

and avoid any problems with **-m** used as an extension.

## OPERANDS

The OPERANDS section describes the utility operands, and how they affect the actions of the utility. Apparent disagreements between functionality descriptions in the OPERANDS and DESCRIPTION (or EXTENDED DESCRIPTION) sections shall be resolved in favor of the OPERANDS section.

If an operand naming a file can be specified as '-', which means to use the standard input instead of a named file, this is explicitly stated in this section. Unless otherwise stated, the use of multiple instances of '-' to mean standard input in a single command produces unspecified results.

Unless otherwise stated, the standard utilities that accept operands shall process those operands in the order specified in the command line.

**Default Behavior:** When this section is listed as "None.", it means that the implementation need not support any operands.

## STDIN

The STDIN section describes the standard input of the utility. This section is frequently merely a reference to the following section, as many utilities treat standard input and input files in the same manner. Unless otherwise stated, all restrictions described in the INPUT FILES section shall apply to this section as well.

Use of a terminal for standard input can cause any of the standard utilities that read standard input to stop when used in the background. For this reason, applications should not use interactive features in scripts to be placed in the background.

The specified standard input format of the standard utilities shall not depend on the existence or value of the environment variables defined in this volume of POSIX.1-2008, except as provided by this volume of POSIX.1-2008.

**Default Behavior:** When this section is listed as "Not used.", it means that the standard input shall not be read when the utility is used as described by this volume of POSIX.1-2008.

## INPUT FILES

The INPUT FILES section describes the files, other than the standard input, used as input by the utility. It includes files named as operands and option-arguments as well as other files that are referred to, such as start-up and initialization files, databases, and so on. Commonly-used files are generally described in one place and cross-referenced by other utilities.

All utilities in this volume of POSIX.1-2008 shall be capable of processing input files using eight-bit transparency.

When a standard utility reads a seekable input file and terminates without an error before it reaches end-of-file, the utility shall ensure that the file offset in the open file description is properly positioned just past the last byte processed by the utility. For files that are not seekable, the state of the file offset in the open file description for that

file is unspecified. A conforming application shall not assume that the following three commands are equivalent:

```
tail -n +2 file
(sed -n 1q; cat) < file
cat file | (sed -n 1q; cat)
```

The second command is equivalent to the first only when the file is seekable. The third command leaves the file offset in the open file description in an unspecified state. Other utilities, such as *head*, *read*, and *sh*, have similar properties.

Some of the standard utilities, such as filters, process input files a line or a block at a time and have no restrictions on the maximum input file size. Some utilities may have size limitations that are not as obvious as file space or memory limitations. Such limitations should reflect resource limitations of some sort, not arbitrary limits set by implementors. Implementations shall document those utilities that are limited by constraints other than file system space, available memory, and other limits specifically cited by this volume of POSIX.1-2008, and identify what the constraint is and indicate a way of estimating when the constraint would be reached. Similarly, some utilities descend the directory tree (recursively). Implementations shall also document any limits that they may have in descending the directory tree that are beyond limits cited by this volume of POSIX.1-2008.

When an input file is described as a “text file”, the utility produces undefined results if given input that is not from a text file, unless otherwise stated. Some utilities (for example, *make*, *read*, *sh*) allow for continued input lines using an escaped <newline> convention; unless otherwise stated, the utility need not be able to accumulate more than {LINE\_MAX} bytes from a set of multiple, continued input lines. Thus, for a conforming application the total of all the continued lines in a set cannot exceed {LINE\_MAX}. If a utility using the escaped <newline> convention detects an end-of-file condition immediately after an escaped <newline>, the results are unspecified.

Record formats are described in a notation similar to that used by the C-language function, *printf()*. See XBD Chapter 5 (on page 121) for a description of this notation. The format description is intended to be sufficiently rigorous to allow other applications to generate these input files. However, since <blank>s can legitimately be included in some of the fields described by the standard utilities, particularly in locales other than the POSIX locale, this intent is not always realized.

**Default Behavior:** When this section is listed as “None.”, it means that no input files are required to be supplied when the utility is used as described by this volume of POSIX.1-2008.

## ENVIRONMENT VARIABLES

The ENVIRONMENT VARIABLES section lists what variables affect the utility’s execution.

The entire manner in which environment variables described in this volume of POSIX.1-2008 affect the behavior of each utility is described in the ENVIRONMENT VARIABLES section for that utility, in conjunction with the global effects of the *LANG*, *LC\_ALL*, and *NLSPATH* environment variables described in XBD Chapter 8 (on page 173). The existence or value of environment variables described in this volume of POSIX.1-2008 shall not otherwise affect the specified behavior of the standard utilities. Any effects of the existence or value of environment variables not described by this volume of POSIX.1-2008 upon the standard utilities are unspecified.

For those standard utilities that use environment variables as a means for selecting a

utility to execute (such as CC in *make*), the string provided to the utility is subjected to the path search described for *PATH* in XBD [Chapter 8](#) (on page 173).

All utilities in this volume of POSIX.1-2008 shall be capable of processing environment variable names and values using eight-bit transparency.

**Default Behavior:** When this section is listed as “None.”, it means that the behavior of the utility is not directly affected by environment variables described by this volume of POSIX.1-2008 when the utility is used as described by this volume of POSIX.1-2008.

## ASYNCHRONOUS EVENTS

The ASYNCHRONOUS EVENTS section lists how the utility reacts to such events as signals and what signals are caught.

**Default Behavior:** When this section is listed as “Default.”, or it refers to “the standard action for all other signals; see [Section 1.4](#) (on page 2326)” it means that the action taken as a result of the signal shall be one of the following:

1. The action shall be that inherited from the parent according to the rules of inheritance of signal actions defined in the System Interfaces volume of POSIX.1-2008.
2. When no action has been taken to change the default, the default action shall be that specified by the System Interfaces volume of POSIX.1-2008.
3. The result of the utility’s execution is as if default actions had been taken.

A utility is permitted to catch a signal, perform some additional processing (such as deleting temporary files), restore the default signal action (or action inherited from the parent process), and resignal itself.

## STDOUT

The STDOUT section completely describes the standard output of the utility. This section is frequently merely a reference to the following section, OUTPUT FILES, because many utilities treat standard output and output files in the same manner.

Use of a terminal for standard output may cause any of the standard utilities that write standard output to stop when used in the background. For this reason, applications should not use interactive features in scripts to be placed in the background.

Record formats are described in a notation similar to that used by the C-language function, *printf()*. See XBD [Chapter 5](#) (on page 121) for a description of this notation.

The specified standard output of the standard utilities shall not depend on the existence or value of the environment variables defined in this volume of POSIX.1-2008, except as provided by this volume of POSIX.1-2008.

Some of the standard utilities describe their output using the verb *display*, defined in XBD [Section 3.133](#) (on page 54). Output described in the STDOUT sections of such utilities may be produced using means other than standard output. When standard output is directed to a terminal, the output described shall be written directly to the terminal. Otherwise, the results are undefined.

**Default Behavior:** When this section is listed as “Not used.”, it means that the standard output shall not be written when the utility is used as described by this volume of POSIX.1-2008.

## STDERR

The STDERR section describes the standard error output of the utility. Only those messages that are purposely sent by the utility are described.



74130 Use of a terminal for standard error may cause any of the standard utilities that write  
 74131 standard error output to stop when used in the background. For this reason,  
 74132 applications should not use interactive features in scripts to be placed in the  
 74133 background.

74134 The format of diagnostic messages for most utilities is unspecified, but the language  
 74135 and cultural conventions of diagnostic and informative messages whose format is  
 74136 unspecified by this volume of POSIX.1-2008 should be affected by the setting of  
 74137 XSI `LC_MESSAGES` and `NLSPATH`.

74138 The specified standard error output of standard utilities shall not depend on the  
 74139 existence or value of the environment variables defined in this volume of POSIX.1-2008,  
 74140 except as provided by this volume of POSIX.1-2008.

74141 **Default Behavior:** When this section is listed as “The standard error shall be used only  
 74142 for diagnostic messages.”, it means that, unless otherwise stated, the diagnostic  
 74143 messages shall be sent to the standard error only when the exit status indicates that an  
 74144 error occurred and the utility is used as described by this volume of POSIX.1-2008.

74145 When this section is listed as “Not used.”, it means that the standard error shall not be  
 74146 used when the utility is used as described in this volume of POSIX.1-2008.

## 74147 OUTPUT FILES

74148 The OUTPUT FILES section completely describes the files created or modified by the  
 74149 utility. Temporary or system files that are created for internal usage by this utility or  
 74150 other parts of the implementation (for example, spool, log, and audit files) are not  
 74151 described in this, or any, section. The utilities creating such files and the names of such  
 74152 files are unspecified. If applications are written to use temporary or intermediate files,  
 74153 they should use the `TMPDIR` environment variable, if it is set and represents an  
 74154 accessible directory, to select the location of temporary files.

74155 Implementations shall ensure that temporary files, when used by the standard utilities,  
 74156 are named so that different utilities or multiple instances of the same utility can operate  
 74157 simultaneously without regard to their working directories, or any other process  
 74158 characteristic other than process ID. There are two exceptions to this rule:

- 74159 1. Resources for temporary files other than the name space (for example, disk  
 74160 space, available directory entries, or number of processes allowed) are not  
 74161 guaranteed.
- 74162 2. Certain standard utilities generate output files that are intended as input for  
 74163 other utilities (for example, *lex* generates `lex.yy.c`), and these cannot have unique  
 74164 names. These cases are explicitly identified in the descriptions of the respective  
 74165 utilities.

74166 Any temporary file created by the implementation shall be removed by the  
 74167 implementation upon a utility’s successful exit, exit because of errors, or before  
 74168 termination by any of the `SIGHUP`, `SIGINT`, or `SIGTERM` signals, unless specified  
 74169 otherwise by the utility description.

74170 Receipt of the `SIGQUIT` signal should generally cause termination (unless in some  
 74171 debugging mode) that would bypass any attempted recovery actions.

74172 Record formats are described in a notation similar to that used by the C-language  
 74173 function, `printf()`; see XBD Chapter 5 (on page 121) for a description of this notation.

74174 **Default Behavior:** When this section is listed as “None.”, it means that no files are  
 74175 created or modified as a consequence of direct action on the part of the utility when the  
 74176 utility is used as described by this volume of POSIX.1-2008. However, the utility may

create or modify system files, such as log files, that are outside the utility's normal execution environment.

## EXTENDED DESCRIPTION

The EXTENDED DESCRIPTION section provides a place for describing the actions of very complicated utilities, such as text editors or language processors, which typically have elaborate command languages.

**Default Behavior:** When this section is listed as "None.", no further description is necessary.

## EXIT STATUS

The EXIT STATUS section describes the values the utility shall return to the calling program, or shell, and the conditions that cause these values to be returned. Usually, utilities return zero for successful completion and values greater than zero for various error conditions. If specific numeric values are listed in this section, the system shall use those values for the errors described. In some cases, status values are listed more loosely, such as >0. A strictly conforming application shall not rely on any specific value in the range shown and shall be prepared to receive any value in the range.

For example, a utility may list zero as a successful return, 1 as a failure for a specific reason, and >1 as "an error occurred". In this case, unspecified conditions may cause a 2 or 3, or other value, to be returned. A conforming application should be written so that it tests for successful exit status values (zero in this case), rather than relying upon the single specific error value listed in this volume of POSIX.1-2008. In that way, it has maximum portability, even on implementations with extensions.

Unspecified error conditions may be represented by specific values not listed in this volume of POSIX.1-2008.

## CONSEQUENCES OF ERRORS

The CONSEQUENCES OF ERRORS section describes the effects on the environment, file systems, process state, and so on, when error conditions occur. It does not describe error messages produced or exit status values used.

The many reasons for failure of a utility are generally not specified by the utility descriptions. Utilities may terminate prematurely if they encounter: invalid usage of options, arguments, or environment variables; invalid usage of the complex syntaxes expressed in EXTENDED DESCRIPTION sections; resource exhaustion; difficulties accessing, creating, reading, or writing files; or difficulties associated with the privileges of the process.

The following shall apply to each utility, unless otherwise stated:

- If the requested action cannot be performed on an operand representing a file, directory, user, process, and so on, the utility shall issue a diagnostic message to standard error and continue processing the next operand in sequence, but the final exit status shall be returned as non-zero.

For a utility that recursively traverses a file hierarchy (such as *find* or *chown -R*), if the requested action cannot be performed on a file or directory encountered in the hierarchy, the utility shall issue a diagnostic message to standard error and continue processing the remaining files in the hierarchy, but the final exit status shall be returned as non-zero.

- If the requested action characterized by an option or option-argument cannot be performed, the utility shall issue a diagnostic message to standard error and the exit status returned shall be non-zero.

- When an unrecoverable error condition is encountered, the utility shall exit with a non-zero exit status.
- A diagnostic message shall be written to standard error whenever an error condition occurs.

When a utility encounters an error condition several actions are possible, depending on the severity of the error and the state of the utility. Included in the possible actions of various utilities are: deletion of temporary or intermediate work files; deletion of incomplete files; validity checking of the file system or directory.

**Default Behavior:** When this section is listed as “Default.”, it means that any changes to the environment, file systems, process state, and so on are unspecified.

## APPLICATION USAGE

This section is informative.

The APPLICATION USAGE section gives advice to the application programmer or user about the way the utility should be used.

## EXAMPLES

This section is informative.

The EXAMPLES section gives one or more examples of usage, where appropriate. In the event of conflict between an example and a normative part of the specification, the normative material is to be taken as correct.

In all examples, quoting has been used, showing how sample commands (utility names combined with arguments) could be passed correctly to a shell (see *sh*) or as a string to the *system()* function defined in the System Interfaces volume of POSIX.1-2008. Such quoting would not be used if the utility is invoked using one of the *exec* functions defined in the System Interfaces volume of POSIX.1-2008.

## RATIONALE

This section is informative.

This section contains historical information concerning the contents of this volume of POSIX.1-2008 and why features were included or discarded by the standard developers.

## FUTURE DIRECTIONS

This section is informative.

The FUTURE DIRECTIONS section should be used as a guide to current thinking; there is not necessarily a commitment to implement all of these future directions in their entirety.

## SEE ALSO

This section is informative.

The SEE ALSO section lists related entries.

## CHANGE HISTORY

This section is informative.

This section shows the derivation of the entry and any significant changes that have been made to it.

Certain of the standard utilities describe how they can invoke other utilities or applications, such as by passing a command string to the command interpreter. The external influences (STDIN, ENVIRONMENT VARIABLES, and so on) and external effects (STDOUT, CONSEQUENCES OF



74268 ERRORS, and so on) of such invoked utilities are not described in the section concerning the  
 74269 standard utility that invokes them.

## 74270 1.5 Considerations for Utilities in Support of Files of Arbitrary Size

74271 The following utilities support files of any size up to the maximum that can be created by the  
 74272 implementation. This support includes correct writing of file size-related values (such as file  
 74273 sizes and offsets, line numbers, and block counts) and correct interpretation of command line  
 74274 arguments that contain such values.

|       |                 |                                                   |
|-------|-----------------|---------------------------------------------------|
| 74275 | <i>basename</i> | Return non-directory portion of pathname.         |
| 74276 | <i>cat</i>      | Concatenate and print files.                      |
| 74277 | <i>cd</i>       | Change working directory.                         |
| 74278 | <i>chgrp</i>    | Change file group ownership.                      |
| 74279 | <i>chmod</i>    | Change file modes.                                |
| 74280 | <i>chown</i>    | Change file ownership.                            |
| 74281 | <i>cksum</i>    | Write file checksums and sizes.                   |
| 74282 | <i>cmp</i>      | Compare two files.                                |
| 74283 | <i>cp</i>       | Copy files.                                       |
| 74284 | <i>dd</i>       | Convert and copy a file.                          |
| 74285 | <i>df</i>       | Report free disk space.                           |
| 74286 | <i>dirname</i>  | Return directory portion of pathname.             |
| 74287 | <i>du</i>       | Estimate file space usage.                        |
| 74288 | <i>find</i>     | Find files.                                       |
| 74289 | <i>ln</i>       | Link files.                                       |
| 74290 | <i>ls</i>       | List directory contents.                          |
| 74291 | <i>mkdir</i>    | Make directories.                                 |
| 74292 | <i>mv</i>       | Move files.                                       |
| 74293 | <i>pathchk</i>  | Check pathnames.                                  |
| 74294 | <i>pwd</i>      | Return working directory name.                    |
| 74295 | <i>rm</i>       | Remove directory entries.                         |
| 74296 | <i>rmdir</i>    | Remove directories.                               |
| 74297 | <i>sh</i>       | Shell, the standard command language interpreter. |
| 74298 | <i>sum</i>      | Print checksum and block or byte count of a file. |
| 74299 | <i>test</i>     | Evaluate expression.                              |
| 74300 | <i>touch</i>    | Change file access and modification times.        |
| 74301 | <i>ulimit</i>   | Set or report file size limit.                    |

Exceptions to the requirement that utilities support files of any size up to the maximum are as follows:

1. Uses of files as command scripts, or for configuration or control, are exempt. For example, it is not required that *sh* be able to read an arbitrarily large **.profile**.
2. Shell input and output redirection are exempt. For example, it is not required that the redirections *sum < file* or *echo foo > file* succeed for an arbitrarily large existing file.

## 1.6 Built-In Utilities

Any of the standard utilities may be implemented as regular built-in utilities within the command language interpreter. This is usually done to increase the performance of frequently used utilities or to achieve functionality that would be more difficult in a separate environment. The utilities named in Table 1-5 are frequently provided in built-in form. All of the utilities named in the table have special properties in terms of command search order within the shell, as described in Section 2.9.1.1 (on page 2357).

**Table 1-5** Regular Built-In Utilities

|                |                |               |             |                |
|----------------|----------------|---------------|-------------|----------------|
| <i>alias</i>   | <i>false</i>   | <i>hash</i>   | <i>pwd</i>  | <i>ulimit</i>  |
| <i>bg</i>      | <i>fc</i>      | <i>jobs</i>   | <i>read</i> | <i>umask</i>   |
| <i>cd</i>      | <i>fg</i>      | <i>kill</i>   | <i>true</i> | <i>unalias</i> |
| <i>command</i> | <i>getopts</i> | <i>newgrp</i> | <i>type</i> | <i>wait</i>    |

However, all of the standard utilities, including the regular built-ins in the table, but not the special built-ins described in Section 2.14 (on page 2374), shall be implemented in a manner so that they can be accessed via the *exec* family of functions as defined in the System Interfaces volume of POSIX.1-2008 and can be invoked directly by those standard utilities that require it - (*env*, *find*, *nice*, *nohup*, *time*, *xargs*).

74325

Chapter 2

74326

# Shell Command Language

74327

This chapter contains the definition of the Shell Command Language.

74328

## 2.1 Shell Introduction

74329

74330

74331

The shell is a command language interpreter. This chapter describes the syntax of that command language as it is used by the *sh* utility and the *system()* and *popen()* functions defined in the System Interfaces volume of POSIX.1-2008.

74332

74333

The shell operates according to the following general overview of operations. The specific details are included in the cited sections of this chapter.

74334

74335

74336

74337

1. The shell reads its input from a file (see *sh*), from the `-c` option or from the *system()* and *popen()* functions defined in the System Interfaces volume of POSIX.1-2008. If the first line of a file of shell commands starts with the characters "#!", the results are unspecified.

74338

2. The shell breaks the input into tokens: words and operators; see [Section 2.3](#).

74339

74340

3. The shell parses the input into simple commands (see [Section 2.9.1](#)) and compound commands (see [Section 2.9.4](#)).

74341

74342

74343

4. The shell performs various expansions (separately) on different parts of each command, resulting in a list of pathnames and fields to be treated as a command and arguments; see [Section 2.6](#).

74344

74345

5. The shell performs redirection (see [Section 2.7](#)) and removes redirection operators and their operands from the parameter list.

74346

74347

74348

74349

6. The shell executes a function (see [Section 2.9.5](#)), built-in (see [Section 2.14](#)), executable file, or script, giving the names of the arguments as positional parameters numbered 1 to *n*, and the name of the command (or in the case of a function within a script, the name of the script) as the positional parameter numbered 0 (see [Section 2.9.1.1](#)).

74350

74351

7. The shell optionally waits for the command to complete and collects the exit status (see [Section 2.8.2](#)).

## 2.2 Quoting

Quoting is used to remove the special meaning of certain characters or words to the shell. Quoting can be used to preserve the literal meaning of the special characters in the next paragraph, prevent reserved words from being recognized as such, and prevent parameter expansion and command substitution within here-document processing (see [Section 2.7.4](#)).

The application shall quote the following characters if they are to represent themselves:

| & ; < > ( ) \$ ' \ " ' <space> <tab> <newline>

and the following may need to be quoted under certain circumstances. That is, these characters may be special depending on conditions described elsewhere in this volume of POSIX.1-2008:

\* ? [ # ~ = %

The various quoting mechanisms are the escape character, single-quotes, and double-quotes. The here-document represents another form of quoting; see [Section 2.7.4](#).

### 2.2.1 Escape Character (Backslash)

A <backslash> that is not quoted shall preserve the literal value of the following character, with the exception of a <newline>. If a <newline> follows the <backslash>, the shell shall interpret this as line continuation. The <backslash> and <newline> shall be removed before splitting the input into tokens. Since the escaped <newline> is removed entirely from the input and is not replaced by any white space, it cannot serve as a token separator.

### 2.2.2 Single-Quotes

Enclosing characters in single-quotes ( ' ' ) shall preserve the literal value of each character within the single-quotes. A single-quote cannot occur within single-quotes.

### 2.2.3 Double-Quotes

Enclosing characters in double-quotes ( " " ) shall preserve the literal value of all characters within the double-quotes, with the exception of the characters backquote, <dollar-sign>, and <backslash>, as follows:

\$ The <dollar-sign> shall retain its special meaning introducing parameter expansion (see [Section 2.6.2](#)), a form of command substitution (see [Section 2.6.3](#)), and arithmetic expansion (see [Section 2.6.4](#)).

The input characters within the quoted string that are also enclosed between "\$ ( " and the matching ' ) ' shall not be affected by the double-quotes, but rather shall define that command whose output replaces the "\$ ( . . . ) " when the word is expanded. The tokenizing rules in [Section 2.3](#), not including the alias substitutions in [Section 2.3.1](#), shall be applied recursively to find the matching ' ) '.

Within the string of characters from an enclosed "\$ { " to the matching ' } ' , an even number of unescaped double-quotes or single-quotes, if any, shall occur. A preceding <backslash> character shall be used to escape a literal ' { ' or ' } ' . The rule in [Section 2.6.2](#) shall be used to determine the matching ' } ' .

74389     ``` The backquote shall retain its special meaning introducing the other form of command  
 74390 substitution (see [Section 2.6.3](#)). The portion of the quoted string from the initial backquote  
 74391 and the characters up to the next backquote that is not preceded by a `<backslash>`, having  
 74392 escape characters removed, defines that command whose output replaces `"`...`"` when  
 74393 the word is expanded. Either of the following cases produces undefined results:

- 74394         • A single-quoted or double-quoted string that begins, but does not end, within the  
 74395         `"`...`"` sequence
- 74396         • A `"`...`"` sequence that begins, but does not end, within the same double-quoted  
 74397         string

74398     `\` The `<backslash>` shall retain its special meaning as an escape character (see [Section 2.2.1](#))  
 74399     only when followed by one of the following characters when considered special:

74400         `$   `   "   \   <newline>`

74401     The application shall ensure that a double-quote is preceded by a `<backslash>` to be included  
 74402     within double-quotes. The parameter `'@'` has special meaning inside double-quotes and is  
 74403     described in [Section 2.5.2](#).

## 74404   2.3   Token Recognition

74405     The shell shall read its input in terms of lines. (For details about how the shell reads its input, see  
 74406     the description of [sh](#).) The input lines can be of unlimited length. These lines shall be parsed  
 74407     using two major modes: ordinary token recognition and processing of here-documents.

74408     When an **io\_here** token has been recognized by the grammar (see [Section 2.10](#)), one or more of  
 74409     the subsequent lines immediately following the next **NEWLINE** token form the body of one or  
 74410     more here-documents and shall be parsed according to the rules of [Section 2.7.4](#).

74411     When it is not processing an **io\_here**, the shell shall break its input into tokens by applying the  
 74412     first applicable rule below to the next character in its input. The token shall be from the current  
 74413     position in the input until a token is delimited according to one of the rules below; the characters  
 74414     forming the token are exactly those in the input, including any quoting characters. If it is  
 74415     indicated that a token is delimited, and no characters have been included in a token, processing  
 74416     shall continue until an actual token is delimited.

- 74417         1. If the end of input is recognized, the current token (if any) shall be delimited. |
- 74418         2. If the previous character was used as part of an operator and the current character is not |  
 74419         quoted and can be used with the previous characters to form an operator, it shall be used  
 74420         as part of that (operator) token.
- 74421         3. If the previous character was used as part of an operator and the current character cannot |  
 74422         be used with the previous characters to form an operator, the operator containing the  
 74423         previous character shall be delimited.
- 74424         4. If the current character is `<backslash>`, single-quote, or double-quote and it is not quoted,  
 74425         it shall affect quoting for subsequent characters up to the end of the quoted text. The rules  
 74426         for quoting are as described in [Section 2.2](#). During token recognition no substitutions  
 74427         shall be actually performed, and the result token shall contain exactly the characters that  
 74428         appear in the input (except for `<newline>` joining), unmodified, including any embedded  
 74429         or enclosing quotes or substitution operators, between the `<quotation-mark>` and the end  
 74430         of the quoted text. The token shall not be delimited by the end of the quoted field.

5. If the current character is an unquoted '\$' or '\', the shell shall identify the start of any candidates for parameter expansion (Section 2.6.2), command substitution (Section 2.6.3), or arithmetic expansion (Section 2.6.4) from their introductory unquoted character sequences: '\$' or '\${', '\$(' or '\', and '\$((', respectively. The shell shall read sufficient input to determine the end of the unit to be expanded (as explained in the cited sections). While processing the characters, if instances of expansions or quoting are found nested within the substitution, the shell shall recursively process them in the manner specified for the construct that is found. The characters found from the beginning of the substitution to its end, allowing for any recursion necessary to recognize embedded constructs, shall be included unmodified in the result token, including any embedded or enclosing substitution operators or quotes. The token shall not be delimited by the end of the substitution.
6. If the current character is not quoted and can be used as the first character of a new operator, the current token (if any) shall be delimited. The current character shall be used as the beginning of the next (operator) token.
7. If the current character is an unquoted <blank>, any token containing the previous character is delimited and the current character shall be discarded.
8. If the previous character was part of a word, the current character shall be appended to that word.
9. If the current character is a '#', it and all subsequent characters up to, but excluding, the next <newline> shall be discarded as a comment. The <newline> that ends the line is not considered part of the comment.
10. The current character is used as the start of a new word.

Once a token is delimited, it is categorized as required by the grammar in Section 2.10.

### 2.3.1 Alias Substitution

After a token has been delimited, but before applying the grammatical rules in Section 2.10, a resulting word that is identified to be the command name word of a simple command shall be examined to determine whether it is an unquoted, valid alias name. However, reserved words in correct grammatical context shall not be candidates for alias substitution. A valid alias name (see XBD Section 3.10) shall be one that has been defined by the *alias* utility and not subsequently undefined using *unalias*. Implementations also may provide predefined valid aliases that are in effect when the shell is invoked. To prevent infinite loops in recursive aliasing, if the shell is not currently processing an alias of the same name, the word shall be replaced by the value of the alias; otherwise, it shall not be replaced.

If the value of the alias replacing the word ends in a <blank>, the shell shall check the next command word for alias substitution; this process shall continue until a word is found that is not a valid alias or an alias value does not end in a <blank>.

When used as specified by this volume of POSIX.1-2008, alias definitions shall not be inherited by separate invocations of the shell or by the utility execution environments invoked by the shell; see Section 2.12.

## 2.4 Reserved Words

Reserved words are words that have special meaning to the shell; see [Section 2.9](#). The following words shall be recognized as reserved words:

|      |      |      |       |
|------|------|------|-------|
| !    | do   | esac | in    |
| {    | done | fi   | then  |
| }    | elif | for  | until |
| case | else | if   | while |

This recognition shall only occur when none of the characters is quoted and when the word is used as:

- The first word of a command
- The first word following one of the reserved words other than **case**, **for**, or **in**
- The third word in a **case** command (only **in** is valid in this case)
- The third word in a **for** command (only **in** and **do** are valid in this case)

See the grammar in [Section 2.10](#).

The following words may be recognized as reserved words on some implementations (when none of the characters are quoted), causing unspecified results:

|   |   |          |        |
|---|---|----------|--------|
| [ | ] | function | select |
|---|---|----------|--------|

Words that are the concatenation of a name and a <colon> ( ' : ' ) are reserved; their use produces unspecified results.

## 2.5 Parameters and Variables

A parameter can be denoted by a name, a number, or one of the special characters listed in [Section 2.5.2](#). A variable is a parameter denoted by a name.

A parameter is set if it has an assigned value (null is a valid value). Once a variable is set, it can only be unset by using the *unset* special built-in command.

### 2.5.1 Positional Parameters

A positional parameter is a parameter denoted by the decimal value represented by one or more digits, other than the single digit 0. The digits denoting the positional parameters shall always be interpreted as a decimal value, even if there is a leading zero. When a positional parameter with more than one digit is specified, the application shall enclose the digits in braces (see [Section 2.6.2](#)). Positional parameters are initially assigned when the shell is invoked (see *sh*), temporarily replaced when a shell function is invoked (see [Section 2.9.5](#)), and can be reassigned with the *set* special built-in command.



## 2.5.2 Special Parameters

Listed below are the special parameters and the values to which they shall expand. Only the values of the special parameters are listed; see [Section 2.6](#) for a detailed summary of all the stages involved in expanding words.

@ Expands to the positional parameters, starting from one, initially producing one field for each positional parameter that is set. When the expansion occurs in a context where field splitting will be performed, any empty fields may be discarded and each of the non-empty fields shall be further split as described in [Section 2.6.5](#). When the expansion occurs within double-quotes, the behavior is unspecified unless one of the following is true:

- Field splitting as described in [Section 2.6.5](#) would be performed if the expansion were not within double-quotes (regardless of whether field splitting would have any effect; for example, if *IFS* is null).
- The double-quotes are within the *word* of a  $\${parameter}:-word$  or a  $\${parameter}:+word$  expansion (with or without the `<colon>`; see [Section 2.6.2](#)) which would have been subject to field splitting if *parameter* had been expanded instead of *word*.

If one of these conditions is true, the initial fields shall be retained as separate fields, except that if the parameter being expanded was embedded within a word, the first field shall be joined with the beginning part of the original word and the last field shall be joined with the end part of the original word. In all other contexts the results of the expansion are unspecified. If there are no positional parameters, the expansion of '@' shall generate zero fields, even when '@' is within double-quotes; however, if the expansion is embedded within a word which contains one or more other parts that expand to a quoted null string, these null string(s) shall still produce an empty field, except that if the other parts are all within the same double-quotes as the '@', it is unspecified whether the result is zero fields or one empty field.

\* Expands to the positional parameters, starting from one, initially producing one field for each positional parameter that is set. When the expansion occurs in a context where field splitting will be performed, any empty fields may be discarded and each of the non-empty fields shall be further split as described in [Section 2.6.5](#). When the expansion occurs in a context where field splitting will not be performed, the initial fields shall be joined to form a single field with the value of each parameter separated by the first character of the *IFS* variable if *IFS* contains at least one character, or separated by a `<space>` if *IFS* is unset, or with no separation if *IFS* is set to a null string.

# Expands to the decimal number of positional parameters. The command name (parameter 0) shall not be counted in the number given by '#' because it is a special parameter, not a positional parameter.

? Expands to the decimal exit status of the most recent pipeline (see [Section 2.9.2](#)).

– (Hyphen.) Expands to the current option flags (the single-letter option names concatenated into a string) as specified on invocation, by the *set* special built-in command, or implicitly by the shell.

\$ Expands to the decimal process ID of the invoked shell. In a subshell (see [Section 2.12](#)), '\$' shall expand to the same value as that of the current shell.

! Expands to the decimal process ID of the most recent background command (see [Section 2.9.3](#)) executed from the current shell. (For example, background commands executed from subshells do not affect the value of "\$!" in the current shell environment.) For a pipeline, the process ID is that of the last command in the pipeline.



74549 0 (Zero.) Expands to the name of the shell or shell script. See *sh* for a detailed description of  
 74550 how this name is derived.  
 74551 See the description of the *IFS* variable in [Section 2.5.3](#).

### 74552 2.5.3 Shell Variables

74553 Variables shall be initialized from the environment (as defined by XBD [Chapter 8](#) and the *exec*  
 74554 function in the System Interfaces volume of POSIX.1-2008) and can be given new values with  
 74555 variable assignment commands. If a variable is initialized from the environment, it shall be  
 74556 marked for export immediately; see the *export* special built-in. New variables can be defined and  
 74557 initialized with variable assignments, with the *read* or *getopts* utilities, with the *name* parameter  
 74558 in a **for** loop, with the  $\${name=word}$  expansion, or with other mechanisms provided as  
 74559 implementation extensions.

74560 The following variables shall affect the execution of the shell:

|       |    |            |                                                                                                                                  |
|-------|----|------------|----------------------------------------------------------------------------------------------------------------------------------|
| 74561 | UP | <b>ENV</b> | The processing of the <i>ENV</i> shell variable shall be supported if the system supports the User Portability Utilities option. |
|-------|----|------------|----------------------------------------------------------------------------------------------------------------------------------|

|       |  |  |                                                                                            |
|-------|--|--|--------------------------------------------------------------------------------------------|
| 74562 |  |  |                                                                                            |
| 74563 |  |  | This variable, when and only when an interactive shell is invoked, shall be                |
| 74564 |  |  | subjected to parameter expansion (see <a href="#">Section 2.6.2</a> ) by the shell and the |
| 74565 |  |  | resulting value shall be used as a pathname of a file containing shell                     |
| 74566 |  |  | commands to execute in the current environment. The file need not be                       |
| 74567 |  |  | executable. If the expanded value of <i>ENV</i> is not an absolute pathname, the           |
| 74568 |  |  | results are unspecified. <i>ENV</i> shall be ignored if the user's real and effective      |
| 74569 |  |  | user IDs or real and effective group IDs are different.                                    |

|       |             |  |                                                                                    |
|-------|-------------|--|------------------------------------------------------------------------------------|
| 74570 | <i>HOME</i> |  | The pathname of the user's home directory. The contents of <i>HOME</i> are used in |
| 74571 |             |  | tilde expansion (see <a href="#">Section 2.6.1</a> ).                              |

|       |            |  |                                                                                                   |
|-------|------------|--|---------------------------------------------------------------------------------------------------|
| 74572 | <i>IFS</i> |  | A string treated as a list of characters that is used for field splitting, expansion              |
| 74573 |            |  | of the <i>'*'</i> special parameter, and to split lines into fields with the <i>read</i> utility. |
| 74574 |            |  | If the value of <i>IFS</i> includes any bytes that do not form part of a valid character,         |
| 74575 |            |  | the results of field splitting, expansion of <i>'*'</i> , and use of the <i>read</i> utility are  |
| 74576 |            |  | unspecified.                                                                                      |

|       |  |  |                                                                                                       |
|-------|--|--|-------------------------------------------------------------------------------------------------------|
| 74577 |  |  | If <i>IFS</i> is not set, it shall behave as normal for an unset variable, except that                |
| 74578 |  |  | field splitting by the shell and line splitting by the <i>read</i> utility shall be                   |
| 74579 |  |  | performed as if the value of <i>IFS</i> is <space><tab><newline>; see <a href="#">Section 2.6.5</a> . |

|       |  |  |                                                                             |
|-------|--|--|-----------------------------------------------------------------------------|
| 74580 |  |  | The shell shall set <i>IFS</i> to <space><tab><newline> when it is invoked. |
|-------|--|--|-----------------------------------------------------------------------------|

|       |             |  |                                                                                                 |
|-------|-------------|--|-------------------------------------------------------------------------------------------------|
| 74581 | <i>LANG</i> |  | Provide a default value for the internationalization variables that are unset or                |
| 74582 |             |  | null. (See XBD <a href="#">Section 8.2</a> for the precedence of internationalization variables |
| 74583 |             |  | used to determine the values of locale categories.)                                             |

|       |               |  |                                                                                     |
|-------|---------------|--|-------------------------------------------------------------------------------------|
| 74584 | <i>LC_ALL</i> |  | The value of this variable overrides the <i>LC_*</i> variables and <i>LANG</i> , as |
| 74585 |               |  | described in XBD <a href="#">Chapter 8</a> .                                        |

|       |                   |  |                                                                              |
|-------|-------------------|--|------------------------------------------------------------------------------|
| 74586 | <i>LC_COLLATE</i> |  | Determine the behavior of range expressions, equivalence classes, and multi- |
| 74587 |                   |  | character collating elements within pattern matching.                        |

|       |                 |  |                                                                                          |
|-------|-----------------|--|------------------------------------------------------------------------------------------|
| 74588 | <i>LC_CTYPE</i> |  | Determine the interpretation of sequences of bytes of text data as characters            |
| 74589 |                 |  | (for example, single-byte as opposed to multi-byte characters), which                    |
| 74590 |                 |  | characters are defined as letters (character class <b>alpha</b> ) and <blank> characters |
| 74591 |                 |  | (character class <b>blank</b> ), and the behavior of character classes within pattern    |
| 74592 |                 |  | matching. Changing the value of <i>LC_CTYPE</i> after the shell has started shall        |

|       |                    |                                                                                            |
|-------|--------------------|--------------------------------------------------------------------------------------------|
| 74593 |                    | not affect the lexical processing of shell commands in the current shell                   |
| 74594 |                    | execution environment or its subshells. Invoking a shell script or performing              |
| 74595 |                    | <i>exec sh</i> subjects the new shell to the changes in <i>LC_CTYPE</i> .                  |
| 74596 | <i>LC_MESSAGES</i> | Determine the language in which messages should be written.                                |
| 74597 | <i>LINENO</i>      | Set by the shell to a decimal number representing the current sequential line              |
| 74598 |                    | number (numbered starting with 1) within a script or function before it                    |
| 74599 |                    | executes each command. If the user unsets or resets <i>LINENO</i> , the variable may       |
| 74600 |                    | lose its special meaning for the life of the shell. If the shell is not currently          |
| 74601 |                    | executing a script or function, the value of <i>LINENO</i> is unspecified. This            |
| 74602 |                    | volume of POSIX.1-2008 specifies the effects of the variable only for systems              |
| 74603 |                    | supporting the User Portability Utilities option.                                          |
| 74604 | XSI                | <i>NLSPATH</i> Determine the location of message catalogs for the processing of            |
| 74605 |                    | <i>LC_MESSAGES</i> .                                                                       |
| 74606 | <i>PATH</i>        | A string formatted as described in XBD Chapter 8, used to effect command                   |
| 74607 |                    | interpretation; see Section 2.9.1.1.                                                       |
| 74608 | <i>PPID</i>        | Set by the shell to the decimal value of its parent process ID during                      |
| 74609 |                    | initialization of the shell. In a subshell (see Section 2.12), <i>PPID</i> shall be set to |
| 74610 |                    | the same value as that of the parent of the current shell. For example, <i>echo</i>        |
| 74611 |                    | <i>\$PPID</i> and ( <i>echo \$PPID</i> ) would produce the same value.                     |
| 74612 | <i>PS1</i>         | Each time an interactive shell is ready to read a command, the value of this               |
| 74613 |                    | variable shall be subjected to parameter expansion and written to standard                 |
| 74614 |                    | error. The default value shall be "\$ ". For users who have specific additional            |
| 74615 |                    | implementation-defined privileges, the default may be another,                             |
| 74616 |                    | implementation-defined value. The shell shall replace each instance of the                 |
| 74617 |                    | character '!' in <i>PS1</i> with the history file number of the next command to be         |
| 74618 |                    | typed. Escaping the '!' with another '!' (that is, "!!") shall place the literal           |
| 74619 |                    | character '!' in the prompt. This volume of POSIX.1-2008 specifies the effects             |
| 74620 |                    | of the variable only for systems supporting the User Portability Utilities                 |
| 74621 |                    | option.                                                                                    |
| 74622 | <i>PS2</i>         | Each time the user enters a <newline> prior to completing a command line in                |
| 74623 |                    | an interactive shell, the value of this variable shall be subjected to parameter           |
| 74624 |                    | expansion and written to standard error. The default value is "> ". This                   |
| 74625 |                    | volume of POSIX.1-2008 specifies the effects of the variable only for systems              |
| 74626 |                    | supporting the User Portability Utilities option.                                          |
| 74627 | <i>PS4</i>         | When an execution trace ( <i>set -x</i> ) is being performed in an interactive shell,      |
| 74628 |                    | before each line in the execution trace, the value of this variable shall be               |
| 74629 |                    | subjected to parameter expansion and written to standard error. The default                |
| 74630 |                    | value is "+ ". This volume of POSIX.1-2008 specifies the effects of the                    |
| 74631 |                    | variable only for systems supporting the User Portability Utilities option.                |
| 74632 | <i>PWD</i>         | Set by the shell and by the <i>cd</i> utility. In the shell the value shall be initialized |
| 74633 |                    | from the environment as follows. If a value for <i>PWD</i> is passed to the shell in       |
| 74634 |                    | the environment when it is executed, the value is an absolute pathname of the              |
| 74635 |                    | current working directory that is no longer than {PATH_MAX} bytes including                |
| 74636 |                    | the terminating null byte, and the value does not contain any components that              |
| 74637 |                    | are dot or dot-dot, then the shell shall set <i>PWD</i> to the value from the              |
| 74638 |                    | environment. Otherwise, if a value for <i>PWD</i> is passed to the shell in the            |
| 74639 |                    | environment when it is executed, the value is an absolute pathname of the                  |
| 74640 |                    | current working directory, and the value does not contain any components                   |

that are dot or dot-dot, then it is unspecified whether the shell sets *PWD* to the value from the environment or sets *PWD* to the pathname that would be output by *pwd -P*. Otherwise, the *sh* utility sets *PWD* to the pathname that would be output by *pwd -P*. In cases where *PWD* is set to the value from the environment, the value can contain components that refer to files of type symbolic link. In cases where *PWD* is set to the pathname that would be output by *pwd -P*, if there is insufficient permission on the current working directory, or on any parent of that directory, to determine what that pathname would be, the value of *PWD* is unspecified. Assignments to this variable may be ignored. If an application sets or unsets the value of *PWD*, the behaviors of the *cd* and *pwd* utilities are unspecified.

## 2.6 Word Expansions

This section describes the various expansions that are performed on words. Not all expansions are performed on every word, as explained in the following sections.

Tilde expansions, parameter expansions, command substitutions, arithmetic expansions, and quote removals that occur within a single word expand to a single field. It is only field splitting or pathname expansion that can create multiple fields from a single word. The single exception to this rule is the expansion of the special parameter '@' within double-quotes, as described in [Section 2.5.2](#).

The order of word expansion shall be as follows:

1. Tilde expansion (see [Section 2.6.1](#)), parameter expansion (see [Section 2.6.2](#)), command substitution (see [Section 2.6.3](#)), and arithmetic expansion (see [Section 2.6.4](#)) shall be performed, beginning to end. See item 5 in [Section 2.3](#).
2. Field splitting (see [Section 2.6.5](#)) shall be performed on the portions of the fields generated by step 1, unless *IFS* is null.
3. Pathname expansion (see [Section 2.6.6](#)) shall be performed, unless *set -f* is in effect.
4. Quote removal (see [Section 2.6.7](#)) shall always be performed last.

The expansions described in this section shall occur in the same shell environment as that in which the command is executed.

If the complete expansion appropriate for a word results in an empty field, that empty field shall be deleted from the list of fields that form the completely expanded command, unless the original word contained single-quote or double-quote characters.

The '\$' character is used to introduce parameter expansion, command substitution, or arithmetic evaluation. If an unquoted '\$' is followed by a character that is not one of the following:

- A numeric character
- The name of one of the special parameters (see [Section 2.5.2](#))
- A valid first character of a variable name
- A <left-curly-bracket> ('{')
- A <left-parenthesis> '('

the result is unspecified.

## 2.6.1 Tilde Expansion

A “tilde-prefix” consists of an unquoted <tilde> character at the beginning of a word, followed by all of the characters preceding the first unquoted <slash> in the word, or all the characters in the word if there is no <slash>. In an assignment (see XBD [Section 4.23](#)), multiple tilde-prefixes can be used: at the beginning of the word (that is, following the <equals-sign> of the assignment), following any unquoted <colon>, or both. A tilde-prefix in an assignment is terminated by the first unquoted <colon> or <slash>. If none of the characters in the tilde-prefix are quoted, the characters in the tilde-prefix following the <tilde> are treated as a possible login name from the user database. A portable login name cannot contain characters outside the set given in the description of the *LOGNAME* environment variable in XBD [Section 8.3](#). If the login name is null (that is, the tilde-prefix contains only the tilde), the tilde-prefix is replaced by the value of the variable *HOME*. If *HOME* is unset, the results are unspecified. Otherwise, the tilde-prefix shall be replaced by a pathname of the initial working directory associated with the login name obtained using the *getpwnam()* function as defined in the System Interfaces volume of POSIX.1-2008. If the system does not recognize the login name, the results are undefined.

The pathname resulting from tilde expansion shall be treated as if quoted to prevent it being altered by field splitting and pathname expansion.

## 2.6.2 Parameter Expansion

The format for parameter expansion is as follows:

```
${expression}
```

where *expression* consists of all characters until the matching '}'. Any '}' escaped by a <backslash> or within a quoted string, and characters in embedded arithmetic expansions, command substitutions, and variable expansions, shall not be examined in determining the matching '}'.

The simplest form for parameter expansion is:

```
${parameter}
```

The value, if any, of *parameter* shall be substituted.

The parameter name or symbol can be enclosed in braces, which are optional except for positional parameters with more than one digit or when *parameter* is a name and is followed by a character that could be interpreted as part of the name. The matching closing brace shall be determined by counting brace levels, skipping over enclosed quoted strings, and command substitutions.

If the parameter is not enclosed in braces, and is a name, the expansion shall use the longest valid name (see XBD [Section 3.235](#)), whether or not the variable represented by that name exists. Otherwise, the parameter is a single-character symbol, and behavior is unspecified if that character is neither a digit nor one of the special parameters (see [Section 2.5.2](#)).

If a parameter expansion occurs inside double-quotes:

- Pathname expansion shall not be performed on the results of the expansion.
- Field splitting shall not be performed on the results of the expansion.

In addition, a parameter expansion can be modified by using one of the following formats. In each case that a value of *word* is needed (based on the state of *parameter*, as described below), *word* shall be subjected to tilde expansion, parameter expansion, command substitution, and arithmetic expansion. If *word* is not needed, it shall not be expanded. The '}' character that

delimits the following parameter expansion modifications shall be determined as described previously in this section and in [Section 2.2.3](#).

**Use Default Values.** If *parameter* is unset or null, the expansion of *word* (or an empty string if *word* is omitted) shall be substituted; otherwise, the value of *parameter* shall be substituted.

**Assign Default Values.** If *parameter* is unset or null, the expansion of *word* (or an empty string if *word* is omitted) shall be assigned to *parameter*. In all cases, the final value of *parameter* shall be substituted. Only variables, not positional parameters or special parameters, can be assigned in this way.

**Indicate Error if Null or Unset.** If *parameter* is unset or null, the expansion of *word* (or a message indicating it is unset if *word* is omitted) shall be written to standard error and the shell exits with a non-zero exit status. Otherwise, the value of *parameter* shall be substituted. An interactive shell need not exit.

**Use Alternative Value.** If *parameter* is unset or null, null shall be substituted; otherwise, the expansion of *word* (or an empty string if *word* is omitted) shall be substituted.

In the parameter expansions shown previously, use of the <colon> in the format shall result in a test for a parameter that is unset or null; omission of the <colon> shall result in a test for a parameter that is only unset. If *parameter* is '#' and the colon is omitted, the application shall ensure that *word* is specified (this is necessary to avoid ambiguity with the string length expansion). The following table summarizes the effect of the <colon>:

|                                  | <i>parameter</i><br>Set and Not Null | <i>parameter</i><br>Set But Null | <i>parameter</i><br>Unset |
|----------------------------------|--------------------------------------|----------------------------------|---------------------------|
| <code>\${parameter:-word}</code> | substitute <i>parameter</i>          | substitute <i>word</i>           | substitute <i>word</i>    |
| <code>\${parameter-word}</code>  | substitute <i>parameter</i>          | substitute null                  | substitute <i>word</i>    |
| <code>\${parameter:=word}</code> | substitute <i>parameter</i>          | assign <i>word</i>               | assign <i>word</i>        |
| <code>\${parameter=word}</code>  | substitute <i>parameter</i>          | substitute null                  | assign <i>word</i>        |
| <code>\${parameter:?word}</code> | substitute <i>parameter</i>          | error, exit                      | error, exit               |
| <code>\${parameter?word}</code>  | substitute <i>parameter</i>          | substitute null                  | error, exit               |
| <code>\${parameter:+word}</code> | substitute <i>word</i>               | substitute null                  | substitute null           |
| <code>\${parameter+word}</code>  | substitute <i>word</i>               | substitute <i>word</i>           | substitute null           |

In all cases shown with “substitute”, the expression is replaced with the value shown. In all cases shown with “assign”, *parameter* is assigned that value, which also replaces the expression.

**String Length.** The length in characters of the value of *parameter* shall be substituted. If *parameter* is '\*' or '@', the result of the expansion is unspecified. If *parameter* is unset and *set -u* is in effect, the expansion shall fail.

The following four varieties of parameter expansion provide for substring processing. In each case, pattern matching notation (see [Section 2.13](#)), rather than regular expression notation, shall be used to evaluate the patterns. If *parameter* is '#', '\*', or '@', the result of the expansion is unspecified. If *parameter* is unset and *set -u* is in effect, the expansion shall fail. Enclosing the full parameter expansion string in double-quotes shall not cause the following four varieties of pattern characters to be quoted, whereas quoting characters within the braces shall have this effect. In each variety, if *word* is omitted, the empty pattern shall be used.



74771 `${parameter%[word]}` **Remove Smallest Suffix Pattern.** The *word* shall be expanded to produce  
 74772 a pattern. The parameter expansion shall then result in *parameter*, with the  
 74773 smallest portion of the suffix matched by the *pattern* deleted. If present,  
 74774 *word* shall not begin with an unquoted ' % '.

74775 `${parameter%%[word]}` **Remove Largest Suffix Pattern.** The *word* shall be expanded to produce a  
 74776 pattern. The parameter expansion shall then result in *parameter*, with the  
 74777 largest portion of the suffix matched by the *pattern* deleted.

74778 `${parameter#[word]}` **Remove Smallest Prefix Pattern.** The *word* shall be expanded to produce  
 74779 a pattern. The parameter expansion shall then result in *parameter*, with the  
 74780 smallest portion of the prefix matched by the *pattern* deleted. If present,  
 74781 *word* shall not begin with an unquoted ' # '.

74782 `${parameter##[word]}` **Remove Largest Prefix Pattern.** The *word* shall be expanded to produce a  
 74783 pattern. The parameter expansion shall then result in *parameter*, with the  
 74784 largest portion of the prefix matched by the *pattern* deleted.

### Examples

74785 `${parameter}`  
 74786 In this example, the effects of omitting braces are demonstrated.  
 74787  
 74788 `a=1`  
 74789 `set 2`  
 74790 `echo ${a}b-$ab-${1}0-${10}-${10}`  
 74791 **1b--20--20**

74792 `${parameter-word}`  
 74793 This example demonstrates the difference between unset and set to the empty string, as well +  
 74794 as the rules for finding the delimiting close brace. +

74795 `foo=asdf`  
 74796 `echo ${foo-bar}xyz}`  
 74797 **asdfxyz}**  
 74798 `foo=`  
 74799 `echo ${foo-bar}xyz}`  
 74800 **xyz}**  
 74801 `unset foo`  
 74802 `echo ${foo-bar}xyz}`  
 74803 **barxyz}**

74804 `${parameter:-word}`  
 74805 In this example, *ls* is executed only if *x* is null or unset. (The `$(ls)` command substitution  
 74806 notation is explained in [Section 2.6.3](#).)

74807 `${x:-$(ls)}`

74808 `${parameter:=word}`  
 74809 `unset X`  
 74810 `echo ${X:=abc}`  
 74811 **abc**

74812 `${parameter:?word}`  
 74813 `unset posix`  
 74814 `echo ${posix:?}`  
 74815 **sh: posix: parameter null or not set**

```

74816  ${parameter:+word}
74817      set a b c
74818      echo ${3:+posix}
74819      posix

74820  ${#parameter}
74821      HOME=/usr/posix
74822      echo ${#HOME}
74823      10

74824  ${parameter%word}
74825      x=file.c
74826      echo ${x%.c}.o
74827      file.o

74828  ${parameter%%word}
74829      x=posix/src/std
74830      echo ${x%/*}
74831      posix

74832  ${parameter#word}
74833      x=$HOME/src/cmd
74834      echo ${x#$HOME}
74835      /src/cmd

74836  ${parameter##word}
74837      x=/one/two/three
74838      echo ${x##*/}
74839      three

```

The double-quoting of patterns is different depending on where the double-quotes are placed:

```

74841  "$ {x#*} "    The <asterisk> is a pattern character.
74842  "${x#" *"} "  The literal <asterisk> is quoted and not special.

```

### 74843 2.6.3 Command Substitution

74844 Command substitution allows the output of a command to be substituted in place of the  
 74845 command name itself. Command substitution shall occur when the command is enclosed as  
 74846 follows:

```
74847 $( command )
```

74848 or (backquoted version):

```
74849 `command`
```

74850 The shell shall expand the command substitution by executing *command* in a subshell  
 74851 environment (see [Section 2.12](#)) and replacing the command substitution (the text of *command*  
 74852 plus the enclosing "\$ ( ) " or backquotes) with the standard output of the command, removing  
 74853 sequences of one or more <newline> characters at the end of the substitution. Embedded  
 74854 <newline> characters before the end of the output shall not be removed; however, they may be  
 74855 treated as field delimiters and eliminated during field splitting, depending on the value of *IFS*  
 74856 and quoting that is in effect. If the output contains any null bytes, the behavior is unspecified.

74857 Within the backquoted style of command substitution, <backslash> shall retain its literal  
 74858 meaning, except when followed by: '\$', '`', or <backslash>. The search for the matching

backquote shall be satisfied by the first unquoted non-escaped backquote; during this search, if a non-escaped backquote is encountered within a shell comment, a here-document, an embedded command substitution of the  $\$(command)$  form, or a quoted string, undefined results occur. A single-quoted or double-quoted string that begins, but does not end, within the `"`...`"` sequence produces undefined results.

With the  $\$(command)$  form, all characters following the open parenthesis to the matching closing parenthesis constitute the *command*. Any valid shell script can be used for *command*, except a script consisting solely of redirections which produces unspecified results.

The results of command substitution shall not be processed for further tilde expansion, parameter expansion, command substitution, or arithmetic expansion. If a command substitution occurs inside double-quotes, field splitting and pathname expansion shall not be performed on the results of the substitution.

Command substitution can be nested. To specify nesting within the backquoted version, the application shall precede the inner backquotes with `<backslash>` characters; for example:

```
\`command\`
```

The syntax of the shell command language has an ambiguity for expansions beginning with `"$( "`, which can introduce an arithmetic expansion or a command substitution that starts with a subshell. Arithmetic expansion has precedence; that is, the shell shall first determine whether it can parse the expansion as an arithmetic expansion and shall only parse the expansion as a command substitution if it determines that it cannot parse the expansion as an arithmetic expansion. The shell need not evaluate nested expansions when performing this determination. If it encounters the end of input without already having determined that it cannot parse the expansion as an arithmetic expansion, the shell shall treat the expansion as an incomplete arithmetic expansion and report a syntax error. A conforming application shall ensure that it separates the `"$( "` and `'( '` into two tokens (that is, separate them with white space) in a command substitution that starts with a subshell. For example, a command substitution containing a single subshell could be written as:

```
$( (command) )
```

## 2.6.4 Arithmetic Expansion

Arithmetic expansion provides a mechanism for evaluating an arithmetic expression and substituting its value. The format for arithmetic expansion shall be as follows:

```
$( (expression) )
```

The expression shall be treated as if it were in double-quotes, except that a double-quote inside the expression is not treated specially. The shell shall expand all tokens in the expression for parameter expansion, command substitution, and quote removal.

Next, the shell shall treat this as an arithmetic expression and substitute the value of the expression. The arithmetic expression shall be processed according to the rules given in [Section 1.1.2.1](#), with the following exceptions:

- Only signed long integer arithmetic is required.
- Only the decimal-constant, octal-constant, and hexadecimal-constant constants specified in the ISO C standard, Section 6.4.4.1 are required to be recognized as constants.
- The `sizeof()` operator and the prefix and postfix `"++"` and `"--"` operators are not required.



- Selection, iteration, and jump statements are not supported.

All changes to variables in an arithmetic expression shall be in effect after the arithmetic expansion, as in the parameter expansion "`${x=value}`".

If the shell variable *x* contains a value that forms a valid integer constant, optionally including a leading `<plus-sign>` or `<hyphen-minus>`, then the arithmetic expansions "`$( (x) )`" and "`$( ( $x ) )`" shall return the same value.

As an extension, the shell may recognize arithmetic expressions beyond those listed. The shell may use a signed integer type with a rank larger than the rank of **signed long**. The shell may use a real-floating type instead of **signed long** as long as it does not affect the results in cases where there is no overflow. If the expression is invalid, or the contents of a shell variable used in the expression are not recognized by the shell, the expansion fails and the shell shall write a diagnostic message to standard error indicating the failure.

### Examples

A simple example using arithmetic expansion:

```
# repeat a command 100 times
x=100
while [ $x -gt 0 ]
do
    command
    x=$(( $x-1 ))
done
```

## 2.6.5 Field Splitting

After parameter expansion (Section 2.6.2), command substitution (Section 2.6.3), and arithmetic expansion (Section 2.6.4), the shell shall scan the results of expansions and substitutions that did not occur in double-quotes for field splitting and multiple fields can result.

The shell shall treat each character of the *IFS* as a delimiter and use the delimiters as field terminators to split the results of parameter expansion, command substitution, and arithmetic expansion into fields.

1. If the value of *IFS* is a `<space>`, `<tab>`, and `<newline>`, or if it is unset, any sequence of `<space>`, `<tab>`, or `<newline>` characters at the beginning or end of the input shall be ignored and any sequence of those characters within the input shall delimit a field. For example, the input:

```
<newline><space><tab>foo<tab><tab>bar<space>
```

yields two fields, **foo** and **bar**.

2. If the value of *IFS* is null, no field splitting shall be performed.
3. Otherwise, the following rules shall be applied in sequence. The term "*IFS* white space" is used to mean any sequence (zero or more instances) of white-space characters that are in the *IFS* value (for example, if *IFS* contains `<space>/<comma>/<tab>`, any sequence of `<space>` and `<tab>` characters is considered *IFS* white space).

- 74940 a. *IFS* white space shall be ignored at the beginning and end of the input.
- 74941 b. Each occurrence in the input of an *IFS* character that is not *IFS* white space, along
- 74942 with any adjacent *IFS* white space, shall delimit a field, as described previously.
- 74943 c. Non-zero-length *IFS* white space shall delimit a field.

## 74944 2.6.6 Pathname Expansion

74945 After field splitting, if *set -f* is not in effect, each field in the resulting command line shall be  
 74946 expanded using the algorithm described in [Section 2.13](#), qualified by the rules in [Section 2.13.3](#).

## 74947 2.6.7 Quote Removal

74948 The quote characters (<backslash>, single-quote, and double-quote) that were present in the  
 74949 original word shall be removed unless they have themselves been quoted.

## 74950 2.7 Redirection

74951 Redirection is used to open and close files for the current shell execution environment (see  
 74952 [Section 2.12](#)) or for any command. Redirection operators can be used with numbers representing  
 74953 file descriptors (see XBD [Section 3.166](#)) as described below.

74954 The overall format used for redirection is:

74955 `[n]redir-op word`

74956 The number *n* is an optional decimal number designating the file descriptor number; the  
 74957 application shall ensure it is delimited from any preceding text and immediately precede the  
 74958 redirection operator *redir-op*. If *n* is quoted, the number shall not be recognized as part of the  
 74959 redirection expression. For example:

74960 `echo \2>a`

74961 writes the character 2 into file **a**. If any part of *redir-op* is quoted, no redirection expression is  
 74962 recognized. For example:

74963 `echo 2\>a`

74964 writes the characters `2>a` to standard output. The optional number, redirection operator, and  
 74965 *word* shall not appear in the arguments provided to the command to be executed (if any).

74966 Open files are represented by decimal numbers starting with zero. The largest possible value is  
 74967 implementation-defined; however, all implementations shall support at least 0 to 9, inclusive, for  
 74968 use by the application. These numbers are called “file descriptors”. The values 0, 1, and 2 have  
 74969 special meaning and conventional uses and are implied by certain redirection operations; they  
 74970 are referred to as *standard input*, *standard output*, and *standard error*, respectively. Programs  
 74971 usually take their input from standard input, and write output on standard output. Error  
 74972 messages are usually written on standard error. The redirection operators can be preceded by  
 74973 one or more digits (with no intervening <blank> characters allowed) to designate the file  
 74974 descriptor number.

74975 If the redirection operator is “<<” or “<<-”, the word that follows the redirection operator shall  
 74976 be subjected to quote removal; it is unspecified whether any of the other expansions occur. For  
 74977 the other redirection operators, the word that follows the redirection operator shall be subjected

74978 to tilde expansion, parameter expansion, command substitution, arithmetic expansion, and  
 74979 quote removal. Pathname expansion shall not be performed on the word by a non-interactive  
 74980 shell; an interactive shell may perform it, but shall do so only when the expansion would result  
 74981 in one word.

74982 If more than one redirection operator is specified with a command, the order of evaluation is  
 74983 from beginning to end.

74984 A failure to open or create a file shall cause a redirection to fail.

### 74985 **2.7.1 Redirecting Input**

74986 Input redirection shall cause the file whose name results from the expansion of *word* to be  
 74987 opened for reading on the designated file descriptor, or standard input if the file descriptor is  
 74988 not specified.

74989 The general format for redirecting input is:

74990 `[n]<word`

74991 where the optional *n* represents the file descriptor number. If the number is omitted, the  
 74992 redirection shall refer to standard input (file descriptor 0).

### 74993 **2.7.2 Redirecting Output**

74994 The two general formats for redirecting output are:

74995 `[n]>word`

74996 `[n]>|word`

74997 where the optional *n* represents the file descriptor number. If the number is omitted, the  
 74998 redirection shall refer to standard output (file descriptor 1).

74999 Output redirection using the '*>*' format shall fail if the *noclobber* option is set (see the  
 75000 description of *set -C*) and the file named by the expansion of *word* exists and is a regular file.  
 75001 Otherwise, redirection using the '*>*' or '*>|*' formats shall cause the file whose name results  
 75002 from the expansion of *word* to be created and opened for output on the designated file  
 75003 descriptor, or standard output if none is specified. If the file does not exist, it shall be created;  
 75004 otherwise, it shall be truncated to be an empty file after being opened.

### 75005 **2.7.3 Appending Redirected Output**

75006 Appended output redirection shall cause the file whose name results from the expansion of  
 75007 *word* to be opened for output on the designated file descriptor. The file is opened as if the *open()*  
 75008 function as defined in the System Interfaces volume of POSIX.1-2008 was called with the  
 75009 *O\_APPEND* flag. If the file does not exist, it shall be created.

75010 The general format for appending redirected output is as follows:

75011 `[n]>>word`

75012 where the optional *n* represents the file descriptor number. If the number is omitted, the  
 75013 redirection refers to standard output (file descriptor 1).

## 2.7.4 Here-Document

The redirection operators "<<" and "<<-" both allow redirection of subsequent lines read by the shell to the input of a command. The redirected lines are known as a "here-document".

The here-document shall be treated as a single word that begins after the next <newline> and continues until there is a line containing only the delimiter and a <newline>, with no <blank> characters in between. Then the next here-document starts, if there is one. The format is as follows:

```
[n]<<word
      here-document
      delimiter
```

where the optional *n* represents the file descriptor number. If the number is omitted, the here-document refers to standard input (file descriptor 0). It is unspecified whether the file descriptor is opened as a regular file, a special file, or a pipe. Portable applications cannot rely on the file descriptor being seekable (see [lseek\(\)](#)).

If any part of *word* is quoted, the delimiter shall be formed by performing quote removal on *word*, and the here-document lines shall not be expanded. Otherwise, the delimiter shall be the *word* itself.

If no part of *word* is quoted, all lines of the here-document shall be expanded for parameter expansion, command substitution, and arithmetic expansion. In this case, the <backslash> in the input behaves as the <backslash> inside double-quotes (see [Section 2.2.3](#)). However, the double-quote character ('"') shall not be treated specially within a here-document, except when the double-quote appears within "\$()", "\$`", or "\${ }".

If the redirection operator is "<<-", all leading <tab> characters shall be stripped from input lines and the line containing the trailing delimiter. If more than one "<<" or "<<-" operator is specified on a line, the here-document associated with the first operator shall be supplied first by the application and shall be read first by the shell.

When a here-document is read from a terminal device and the shell is interactive, it shall write the contents of the variable *PS2*, processed as described in [Section 2.5.3](#), to standard error before reading each line of input until the delimiter has been recognized.

### Examples

An example of a here-document follows:

```
cat <<eof1; cat <<eof2
Hi,
eof1
Helene.
eof2
```

**2.7.5 Duplicating an Input File Descriptor**

The redirection operator:

`[n]<&word`

shall duplicate one input file descriptor from another, or shall close one. If *word* evaluates to one or more digits, the file descriptor denoted by *n*, or standard input if *n* is not specified, shall be made to be a copy of the file descriptor denoted by *word*; if the digits in *word* do not represent a file descriptor already open for input, a redirection error shall result; see [Section 2.8.1](#). If *word* evaluates to `'-'`, file descriptor *n*, or standard input if *n* is not specified, shall be closed. Attempts to close a file descriptor that is not open shall not constitute an error. If *word* evaluates to something else, the behavior is unspecified.

**2.7.6 Duplicating an Output File Descriptor**

The redirection operator:

`[n]>&word`

shall duplicate one output file descriptor from another, or shall close one. If *word* evaluates to one or more digits, the file descriptor denoted by *n*, or standard output if *n* is not specified, shall be made to be a copy of the file descriptor denoted by *word*; if the digits in *word* do not represent a file descriptor already open for output, a redirection error shall result; see [Section 2.8.1](#). If *word* evaluates to `'-'`, file descriptor *n*, or standard output if *n* is not specified, is closed. Attempts to close a file descriptor that is not open shall not constitute an error. If *word* evaluates to something else, the behavior is unspecified.

**2.7.7 Open File Descriptors for Reading and Writing**

The redirection operator:

`[n]<>word`

shall cause the file whose name is the expansion of *word* to be opened for both reading and writing on the file descriptor denoted by *n*, or standard input if *n* is not specified. If the file does not exist, it shall be created.

**2.8 Exit Status and Errors****2.8.1 Consequences of Shell Errors**

Certain errors shall cause the shell to write a diagnostic message to standard error and exit as shown in the following table:

| Error                                                          | Non-Interactive Shell | Interactive Shell | Shell Diagnostic Message Required |
|----------------------------------------------------------------|-----------------------|-------------------|-----------------------------------|
| Shell language syntax error                                    | shall exit            | shall not exit    | yes                               |
| Special built-in utility error                                 | shall exit            | shall not exit    | no <sup>1</sup>                   |
| Other utility (not a special built-in) error                   | shall not exit        | shall not exit    | no <sup>2</sup>                   |
| Redirection error with special built-in utilities              | shall exit            | shall not exit    | yes                               |
| Redirection error with compound commands                       | may exit <sup>3</sup> | shall not exit    | yes                               |
| Redirection error with function execution                      | may exit <sup>3</sup> | shall not exit    | yes                               |
| Redirection error with other utilities (not special built-ins) | shall not exit        | shall not exit    | yes                               |
| Variable assignment error                                      | shall exit            | shall not exit    | yes                               |
| Expansion error                                                | shall exit            | shall not exit    | yes                               |
| Command not found                                              | may exit              | shall not exit    | yes                               |

Notes:

1. Although special built-ins are part of the shell, a diagnostic message written by a special built-in is not considered to be a shell diagnostic message, and can be redirected like any other utility.
2. The shell is not required to write a diagnostic message, but the utility itself shall write a diagnostic message if required to do so.
3. A future version of this standard may require the shell to not exit in this condition.

An expansion error is one that occurs when the shell expansions define in [Section 2.6](#) are carried out (for example, "`{x!y}`", because '`!`' is not a valid operator); an implementation may treat these as syntax errors if it is able to detect them during tokenization, rather than during expansion.

If any of the errors shown as "shall exit" or "may exit" occur in a subshell environment, the shell shall (respectively, may) exit from the subshell environment with a non-zero status and continue in the environment from which that subshell environment was invoked.

In all of the cases shown in the table where an interactive shell is required not to exit, the shell shall not perform any further processing of the command in which the error occurred.

## 2.8.2 Exit Status for Commands

Each command has an exit status that can influence the behavior of other shell commands. The exit status of commands that are not utilities is documented in this section. The exit status of the standard utilities is documented in their respective sections.

If a command is not found, the exit status shall be 127. If the command name is found, but it is not an executable utility, the exit status shall be 126. Applications that invoke utilities without using the shell should use these exit status values to report similar errors.

If a command fails during word expansion or redirection, its exit status shall be between 1 and 125 inclusive.

Internally, for purposes of deciding whether a command exits with a non-zero exit status, the

shell shall recognize the entire status value retrieved for the command by the equivalent of the `wait()` function WEXITSTATUS macro (as defined in the System Interfaces volume of POSIX.1-2008). When reporting the exit status with the special parameter `'?'`, the shell shall report the full eight bits of exit status available. The exit status of a command that terminated because it received a signal shall be reported as greater than 128.

## 2.9 Shell Commands

This section describes the basic structure of shell commands. The following command descriptions each describe a format of the command that is only used to aid the reader in recognizing the command type, and does not formally represent the syntax. In particular, the representations include spacing between tokens in some places where `<blank>s` would not be necessary (when one of the tokens is an operator). Each description discusses the semantics of the command; for a formal definition of the command language, consult [Section 2.10](#).

A *command* is one of the following:

- Simple command (see [Section 2.9.1](#))
- Pipeline (see [Section 2.9.2](#))
- List compound-list (see [Section 2.9.3](#))
- Compound command (see [Section 2.9.4](#))
- Function definition (see [Section 2.9.5](#))

Unless otherwise stated, the exit status of a command shall be that of the last simple command executed by the command. There shall be no limit on the size of any shell command other than that imposed by the underlying system (memory constraints, `{ARG_MAX}`, and so on).

### 2.9.1 Simple Commands

A “simple command” is a sequence of optional variable assignments and redirections, in any sequence, optionally followed by words and redirections, terminated by a control operator.

When a given simple command is required to be executed (that is, when any conditional construct such as an AND-OR list or a **case** statement has not bypassed the simple command), the following expansions, assignments, and redirections shall all be performed from the beginning of the command text to the end:

1. The words that are recognized as variable assignments or redirections according to [Section 2.10.2](#) are saved for processing in steps 3 and 4.
2. The words that are not variable assignments or redirections shall be expanded. If any fields remain following their expansion, the first field shall be considered the command name and remaining fields are the arguments for the command.
3. Redirections shall be performed as described in [Section 2.7](#).
4. Each variable assignment shall be expanded for tilde expansion, parameter expansion, command substitution, arithmetic expansion, and quote removal prior to assigning the value.

In the preceding list, the order of steps 3 and 4 may be reversed if no command name results from step 2 or if the command name matches the name of a special built-in utility; see [Section 2.14](#).



|       |                                                                                                                            |   |
|-------|----------------------------------------------------------------------------------------------------------------------------|---|
| 75163 | Variable assignments shall be performed as follows:                                                                        |   |
| 75164 | • If no command name results, variable assignments shall affect the current execution                                      | + |
| 75165 | environment.                                                                                                               | + |
| 75166 | • If the command name is not a special built-in utility or function, the variable assignments                              | + |
| 75167 | shall be exported for the execution environment of the command and shall not affect the                                    | + |
| 75168 | current execution environment except as a side-effect of the expansions performed in step                                  | + |
| 75169 | 4. In this case it is unspecified:                                                                                         | + |
| 75170 | — Whether or not the assignments are visible for subsequent expansions in step 4                                           |   |
| 75171 | — Whether variable assignments made as side-effects of these expansions are visible for                                    |   |
| 75172 | subsequent expansions in step 4, or in the current shell execution environment, or                                         |   |
| 75173 | both                                                                                                                       |   |
| 75174 | • If the command name is a standard utility implemented as a function (see XBD <a href="#">Section</a>                     | + |
| 75175 | <a href="#">4.22</a> ), the effect of variable assignments shall be as if the utility was not implemented as a             | + |
| 75176 | function.                                                                                                                  | + |
| 75177 | • If the command name is a special built-in utility, variable assignments shall affect the                                 | + |
| 75178 | current execution environment. Unless the <i>set</i> <b>-a</b> option is on (see <a href="#">set</a> ), it is unspecified: | + |
| 75179 | — Whether or not the variables gain the <i>export</i> attribute during the execution of the                                | + |
| 75180 | special built-in utility                                                                                                   | + |
| 75181 | — Whether or not <i>export</i> attributes gained as a result of the variable assignments persist                           | + |
| 75182 | after the completion of the special built-in utility                                                                       | + |
| 75183 | • If the command name is a function that is not a standard utility implemented as a function,                              | + |
| 75184 | variable assignments shall affect the current execution environment during the execution                                   | + |
| 75185 | of the function. It is unspecified:                                                                                        | + |
| 75186 | — Whether or not the variable assignments persist after the completion of the function                                     | + |
| 75187 | — Whether or not the variables gain the <i>export</i> attribute during the execution of the                                | + |
| 75188 | function                                                                                                                   | + |
| 75189 | — Whether or not <i>export</i> attributes gained as a result of the variable assignments persist                           | + |
| 75190 | after the completion of the function (if variable assignments persist after the                                            | + |
| 75191 | completion of the function)                                                                                                | + |
| 75192 | If any of the variable assignments attempt to assign a value to a variable for which the <i>readonly</i>                   |   |
| 75193 | attribute is set in the current shell environment (regardless of whether the assignment is made in                         |   |
| 75194 | that environment), a variable assignment error shall occur. See <a href="#">Section 2.8.1</a> for the consequences         |   |
| 75195 | of these errors.                                                                                                           |   |
| 75196 | If there is no command name, any redirections shall be performed in a subshell environment; it                             |   |
| 75197 | is unspecified whether this subshell environment is the same one as that used for a command                                |   |
| 75198 | substitution within the command. (To affect the current execution environment, see the <i>exec</i>                         |   |
| 75199 | special built-in.) If any of the redirections performed in the current shell execution environment                         |   |
| 75200 | fail, the command shall immediately fail with an exit status greater than zero, and the shell shall                        |   |
| 75201 | write an error message indicating the failure. See <a href="#">Section 2.8.1</a> for the consequences of these             |   |
| 75202 | failures on interactive and non-interactive shells.                                                                        |   |
| 75203 | If there is a command name, execution shall continue as described in <a href="#">Section 2.9.1.1</a> . If there is         |   |
| 75204 | no command name, but the command contained a command substitution, the command shall                                       |   |
| 75205 | complete with the exit status of the last command substitution performed. Otherwise, the                                   |   |
| 75206 | command shall complete with a zero exit status.                                                                            |   |



## 75207 2.9.1.1 Command Search and Execution

75208 If a simple command results in a command name and an optional list of arguments, the  
75209 following actions shall be performed:

75210 1. If the command name does not contain any <slash> characters, the first successful step in  
75211 the following sequence shall occur:

75212 a. If the command name matches the name of a special built-in utility, that special  
75213 built-in utility shall be invoked.

75214 b. If the command name matches the name of a utility listed in the following table, +  
75215 the results are unspecified. +

|       |                 |                      |                   |                |                    |
|-------|-----------------|----------------------|-------------------|----------------|--------------------|
| 75216 | <i>alloc</i>    | <i>comparguments</i> | <i>comptry</i>    | <i>history</i> | <i>pushd</i>       |
| 75217 | <i>autoload</i> | <i>compcall</i>      | <i>compvalues</i> | <i>hist</i>    | <i>readarray</i>   |
| 75218 | <i>bind</i>     | <i>compctl</i>       | <i>declare</i>    | <i>let</i>     | <i>repeat</i>      |
| 75219 | <i>bindkey</i>  | <i>compdescribe</i>  | <i>dirs</i>       | <i>local</i>   | <i>savehistory</i> |
| 75220 | <i>builtin</i>  | <i>compfiles</i>     | <i>disable</i>    | <i>login</i>   | <i>source</i>      |
| 75221 | <i>bye</i>      | <i>compgen</i>       | <i>disown</i>     | <i>logout</i>  | <i>shopt</i>       |
| 75222 | <i>caller</i>   | <i>compgroups</i>    | <i>dosh</i>       | <i>map</i>     | <i>stop</i>        |
| 75223 | <i>cap</i>      | <i>complete</i>      | <i>echoct</i>     | <i>mapfile</i> | <i>suspend</i>     |
| 75224 | <i>chdir</i>    | <i>compquote</i>     | <i>echoit</i>     | <i>popd</i>    | <i>typeset</i>     |
| 75225 | <i>clone</i>    | <i>comptags</i>      | <i>help</i>       | <i>print</i>   | <i>whence</i>      |

75226 c. If the command name matches the name of a function known to this shell, the  
75227 function shall be invoked as described in [Section 2.9.5](#). If the implementation has  
75228 provided a standard utility in the form of a function, it shall not be recognized at  
75229 this point. It shall be invoked in conjunction with the path search in step 1d.

75230 XSI d. If the command name matches the name of the *type* or *ulimit* utility, or of a utility |  
75231 listed in the following table, that utility shall be invoked.

|       |                |                |               |              |                |
|-------|----------------|----------------|---------------|--------------|----------------|
| 75232 | <i>alias</i>   | <i>false</i>   | <i>hash</i>   | <i>pwd</i>   | <i>unalias</i> |
| 75233 | <i>bg</i>      | <i>fc</i>      | <i>jobs</i>   | <i>read</i>  | <i>wait</i>    |
| 75234 | <i>cd</i>      | <i>fg</i>      | <i>kill</i>   | <i>true</i>  |                |
| 75235 | <i>command</i> | <i>getopts</i> | <i>newgrp</i> | <i>umask</i> |                |

75236 e. Otherwise, the command shall be searched for using the *PATH* environment  
75237 variable as described in XBD [Chapter 8](#):

75238 i. If the search is successful:

75239 a. If the system has implemented the utility as a regular built-in or as a  
75240 shell function, it shall be invoked at this point in the path search.

75241 b. Otherwise, the shell executes the utility in a separate utility  
75242 environment (see [Section 2.12](#)) with actions equivalent to calling the  
75243 *execl()* function as defined in the System Interfaces volume of  
75244 POSIX.1-2008 with the *path* argument set to the pathname resulting  
75245 from the search, *arg0* set to the command name, and the remaining  
75246 *execl()* arguments set to the command arguments (if any) and the  
75247 null terminator.

75248 If the *execl()* function fails due to an error equivalent to the  
75249 [ENOEXEC] error defined in the System Interfaces volume of  
75250 POSIX.1-2008, the shell shall execute a command equivalent to  
75251 having a shell invoked with the pathname resulting from the search  
75252 as its first operand, with any remaining arguments passed to the new

shell, except that the value of "\$0" in the new shell may be set to the command name. If the executable file is not a text file, the shell may bypass this command execution. In this case, it shall write an error message, and shall return an exit status of 126.

It is unspecified whether environment variables that were passed to the shell when it was invoked, but were not used to initialize shell variables (see [Section 2.5.3](#)) because they had invalid names, are included in the environment passed to *execl()* and (if *execl()* fails as described above) to the new shell.

Once a utility has been searched for and found (either as a result of this specific search or as part of an unspecified shell start-up activity), an implementation may remember its location and need not search for the utility again unless the *PATH* variable has been the subject of an assignment. If the remembered location fails for a subsequent invocation, the shell shall repeat the search to find the new location for the utility, if any.

- ii. If the search is unsuccessful, the command shall fail with an exit status of 127 and the shell shall write an error message.

2. If the command name contains at least one <slash>, the shell shall execute the utility in a separate utility environment with actions equivalent to calling the *execl()* function defined in the System Interfaces volume of POSIX.1-2008 with the *path* and *arg0* arguments set to the command name, and the remaining *execl()* arguments set to the command arguments (if any) and the null terminator.

If the *execl()* function fails due to an error equivalent to the [ENOEXEC] error, the shell shall execute a command equivalent to having a shell invoked with the command name as its first operand, with any remaining arguments passed to the new shell. If the executable file is not a text file, the shell may bypass this command execution. In this case, it shall write an error message and shall return an exit status of 126.

It is unspecified whether environment variables that were passed to the shell when it was invoked, but were not used to initialize shell variables (see [Section 2.5.3](#)) because they had invalid names, are included in the environment passed to *execl()* and (if *execl()* fails as described above) to the new shell.

If the utility would be executed with file descriptor 0, 1, or 2 closed, implementations may execute the utility with the file descriptor open to an unspecified file. If a standard utility or a conforming application is executed with file descriptor 0 not open for reading or with file descriptor 1 or 2 not open for writing, the environment in which the utility or application is executed shall be deemed non-conforming, and consequently the utility or application might not behave as described in this standard.

## 2.9.2 Pipelines

A *pipeline* is a sequence of one or more commands separated by the control operator '|'. For each command but the last, the shell shall connect the standard output of the command to the standard input of the next command as if by creating a pipe and passing the write end of the pipe as the standard output of the command and the read end of the pipe as the standard input of the next command.

The format for a pipeline is:

```
[!] command1 [ | command2 ... ]
```

If the pipeline begins with the reserved word `!` and *command1* is a subshell command, the application shall ensure that the `(` operator at the beginning of *command1* is separated from the `!` by one or more <blank> characters. The behavior of the reserved word `!` immediately followed by the `(` operator is unspecified.

The standard output of *command1* shall be connected to the standard input of *command2*. The standard input, standard output, or both of a command shall be considered to be assigned by the pipeline before any redirection specified by redirection operators that are part of the command (see [Section 2.7](#)).

If the pipeline is not in the background (see [Section 2.9.3.1](#)), the shell shall wait for the last command specified in the pipeline to complete, and may also wait for all commands to complete.

### Exit Status

If the pipeline does not begin with the `!` reserved word, the exit status shall be the exit status of the last command specified in the pipeline. Otherwise, the exit status shall be the logical NOT of the exit status of the last command. That is, if the last command returns zero, the exit status shall be 1; if the last command returns greater than zero, the exit status shall be zero.

## 2.9.3 Lists

An *AND-OR list* is a sequence of one or more pipelines separated by the operators `&&` and `||`.

A *list* is a sequence of one or more AND-OR lists separated by the operators `;` and `&'`.

The operators `&&` and `||` shall have equal precedence and shall be evaluated with left associativity. For example, both of the following commands write solely **bar** to standard output:

```
false && echo foo || echo bar
true || echo foo && echo bar
```

A `;` separator or a `&'` or <newline> terminator shall cause the preceding AND-OR list to be executed sequentially; an `&'` separator or terminator shall cause asynchronous execution of the preceding AND-OR list.

The term “compound-list” is derived from the grammar in [Section 2.10](#); it is equivalent to a sequence of *lists*, separated by <newline> characters, that can be preceded or followed by an arbitrary number of <newline> characters.

### Examples

The following is an example that illustrates <newline> characters in compound-lists:

```
while
    # a couple of <newline>s
    # a list
    date && who || ls; cat file
    # a couple of <newline>s
    # another list
    wc file > output & true
do
    # 2 lists
```

```

75339         ls
75340         cat file
75341     done

```

#### 75342 2.9.3.1 Asynchronous Lists

75343 If a command is terminated by the control operator <ampersand> ('&'), the shell shall execute  
 75344 the command asynchronously in a subshell. This means that the shell shall not wait for the  
 75345 command to finish before executing the next command.

75346 The format for running a command in the background is:

```
75347 command1 & [command2 & ... ]
```

75348 If job control is disabled (see [set](#), **-m**), the standard input for an asynchronous list, before any  
 75349 explicit redirections are performed, shall be considered to be assigned to a file that has the same  
 75350 properties as **/dev/null**. This shall not happen if job control is enabled. In all cases, explicit  
 75351 redirection of standard input shall override this activity.

75352 When an element of an asynchronous list (the portion of the list ended by an <ampersand>,  
 75353 such as *command1*, above) is started by the shell, the process ID of the last command in the  
 75354 asynchronous list element shall become known in the current shell execution environment; see  
 75355 [Section 2.12](#). This process ID shall remain known until:

- 75356 1. The command terminates and the application waits for the process ID.
- 75357 2. Another asynchronous list is invoked before "\$!" (corresponding to the previous  
 75358 asynchronous list) is expanded in the current execution environment.

75359 The implementation need not retain more than the {CHILD\_MAX} most recent entries in its list  
 75360 of known process IDs in the current shell execution environment.

#### 75361 Exit Status

75362 The exit status of an asynchronous list shall be zero.

#### 75363 2.9.3.2 Sequential Lists

75364 Commands that are separated by a <semicolon> (';') shall be executed sequentially.

75365 The format for executing commands sequentially shall be:

```
75366 command1 [ ; command2 ] ...
```

75367 Each command shall be expanded and executed in the order specified.

#### 75368 Exit Status

75369 The exit status of a sequential list shall be the exit status of the last command in the list.

#### 75370 2.9.3.3 AND Lists

75371 The control operator "&&" denotes an AND list. The format shall be:

```
75372 command1 [ && command2 ] ...
```

75373 First *command1* shall be executed. If its exit status is zero, *command2* shall be executed, and so on,  
 75374 until a command has a non-zero exit status or there are no more commands left to execute. The

75375 commands are expanded only if they are executed.

#### 75376 **Exit Status**

75377 The exit status of an AND list shall be the exit status of the last command that is executed in the  
75378 list.

#### 75379 2.9.3.4 *OR Lists*

75380 The control operator "`|`" denotes an OR List. The format shall be:

75381 `command1 [ | | command2 ] . . .`

75382 First, *command1* shall be executed. If its exit status is non-zero, *command2* shall be executed, and  
75383 so on, until a command has a zero exit status or there are no more commands left to execute.

#### 75384 **Exit Status**

75385 The exit status of an OR list shall be the exit status of the last command that is executed in the  
75386 list.

### 75387 **2.9.4 Compound Commands**

75388 The shell has several programming constructs that are “compound commands”, which provide  
75389 control flow for commands. Each of these compound commands has a reserved word or control  
75390 operator at the beginning, and a corresponding terminator reserved word or operator at the end.  
75391 In addition, each can be followed by redirections on the same line as the terminator. Each  
75392 redirection shall apply to all the commands within the compound command that do not  
75393 explicitly override that redirection.

#### 75394 2.9.4.1 *Grouping Commands*

75395 The format for grouping commands is as follows:

75396 (`compound-list`) Execute *compound-list* in a subshell environment; see [Section 2.12](#).  
75397 Variable assignments and built-in commands that affect the environment  
75398 shall not remain in effect after the list finishes.

75399 If a character sequence beginning with "`(`" would be parsed by the shell  
75400 as an arithmetic expansion if preceded by a '`$`', shells which implement  
75401 an extension whereby "`((expression))`" is evaluated as an arithmetic  
75402 expression may treat the "`(`" as introducing as an arithmetic evaluation  
75403 instead of a grouping command. A conforming application shall ensure  
75404 that it separates the two leading '`(`' characters with white space to  
75405 prevent the shell from performing an arithmetic evaluation.

75406 { `compound-list` ; } Execute *compound-list* in the current process environment. The semicolon  
75407 shown here is an example of a control operator delimiting the `}` reserved  
75408 word. Other delimiters are possible, as shown in [Section 2.10](#); a  
75409 `<newline>` is frequently used.

**Exit Status**

The exit status of a grouping command shall be the exit status of *compound-list*.

2.9.4.2 *The for Loop*

The **for** loop shall execute a sequence of commands for each member in a list of *items*. The **for** loop requires that the reserved words **do** and **done** be used to delimit the sequence of commands.

The format for the **for** loop is as follows:

```
for name [ in [word ... ] ]
do
    compound-list
done
```

First, the list of words following **in** shall be expanded to generate a list of items. Then, the variable *name* shall be set to each item, in turn, and the *compound-list* executed each time. If no items result from the expansion, the *compound-list* shall not be executed. Omitting:

```
in word ...
```

shall be equivalent to:

```
in "$@"
```

**Exit Status**

The exit status of a **for** command shall be the exit status of the last command that executes. If there are no items, the exit status shall be zero.

2.9.4.3 *Case Conditional Construct*

The conditional construct **case** shall execute the *compound-list* corresponding to the first one of several *patterns* (see [Section 2.13](#)) that is matched by the string resulting from the tilde expansion, parameter expansion, command substitution, arithmetic expansion, and quote removal of the given word. The reserved word **in** shall denote the beginning of the patterns to be matched. Multiple patterns with the same *compound-list* shall be delimited by the '|' symbol. The control operator ')' terminates a list of patterns corresponding to a given action. The *compound-list* for each list of patterns, with the possible exception of the last, shall be terminated with ";;". The **case** construct terminates with the reserved word **esac** (**case** reversed).

The format for the **case** construct is as follows:

```
case word in
    [(] pattern1 ) compound-list ;;
    [(] pattern[ | pattern] ... ) compound-list ;;] ...
    [(] pattern[ | pattern] ... ) compound-list]
esac
```

The " ; ; " is optional for the last *compound-list*.

In order from the beginning to the end of the **case** statement, each *pattern* that labels a *compound-list* shall be subjected to tilde expansion, parameter expansion, command substitution, and arithmetic expansion, and the result of these expansions shall be compared against the expansion of *word*, according to the rules described in [Section 2.13](#) (which also describes the

effect of quoting parts of the pattern). After the first match, no more patterns shall be expanded, and the *compound-list* shall be executed. The order of expansion and comparison of multiple *patterns* that label a *compound-list* statement is unspecified.

#### Exit Status

The exit status of **case** shall be zero if no patterns are matched. Otherwise, the exit status shall be the exit status of the last command executed in the *compound-list*.

#### 2.9.4.4 The if Conditional Construct

The **if** command shall execute a *compound-list* and use its exit status to determine whether to execute another *compound-list*.

The format for the **if** construct is as follows:

```
if compound-list
then
    compound-list
[elif compound-list
then
    compound-list] ...
[else
    compound-list]
fi
```

The **if** *compound-list* shall be executed; if its exit status is zero, the **then** *compound-list* shall be executed and the command shall complete. Otherwise, each **elif** *compound-list* shall be executed, in turn, and if its exit status is zero, the **then** *compound-list* shall be executed and the command shall complete. Otherwise, the **else** *compound-list* shall be executed.

#### Exit Status

The exit status of the **if** command shall be the exit status of the **then** or **else** *compound-list* that was executed, or zero, if none was executed.

#### 2.9.4.5 The while Loop

The **while** loop shall continuously execute one *compound-list* as long as another *compound-list* has a zero exit status.

The format of the **while** loop is as follows:

```
while compound-list-1
do
    compound-list-2
done
```

The *compound-list-1* shall be executed, and if it has a non-zero exit status, the **while** command shall complete. Otherwise, the *compound-list-2* shall be executed, and the process shall repeat.



75487 **Exit Status**

75488 The exit status of the **while** loop shall be the exit status of the last *compound-list-2* executed, or  
 75489 zero if none was executed.

75490 2.9.4.6 *The until Loop*

75491 The **until** loop shall continuously execute one *compound-list* as long as another *compound-list* has  
 75492 a non-zero exit status.

75493 The format of the **until** loop is as follows:

```
75494 until compound-list-1
75495 do
75496     compound-list-2
75497 done
```

75498 The *compound-list-1* shall be executed, and if it has a zero exit status, the **until** command  
 75499 completes. Otherwise, the *compound-list-2* shall be executed, and the process repeats.

75500 **Exit Status**

75501 The exit status of the **until** loop shall be the exit status of the last *compound-list-2* executed, or  
 75502 zero if none was executed.

75503 **2.9.5 Function Definition Command**

75504 A function is a user-defined name that is used as a simple command to call a compound  
 75505 command with new positional parameters. A function is defined with a “function definition  
 75506 command”.

75507 The format of a function definition command is as follows:

```
75508 fname ( ) compound-command [io-redirect ...]
```

75509 The function is named *fname*; the application shall ensure that it is a name (see XBD [Section](#)  
 75510 [3.235](#)) and that it is not the name of a special built-in utility. An implementation may allow other  
 75511 characters in a function name as an extension. The implementation shall maintain separate name  
 75512 spaces for functions and variables.

75513 The argument *compound-command* represents a compound command, as described in [Section](#)  
 75514 [2.9.4](#).

75515 When the function is declared, none of the expansions in [Section 2.6](#) shall be performed on the  
 75516 text in *compound-command* or *io-redirect*; all expansions shall be performed as normal each time  
 75517 the function is called. Similarly, the optional *io-redirect* redirections and any variable assignments  
 75518 within *compound-command* shall be performed during the execution of the function itself, not the  
 75519 function definition. See [Section 2.8.1](#) for the consequences of failures of these operations on  
 75520 interactive and non-interactive shells.

75521 When a function is executed, it shall have the syntax-error properties described for special built-  
 75522 in utilities in the first item in the enumerated list at the beginning of [Section 2.14](#).

75523 The *compound-command* shall be executed whenever the function name is specified as the name  
 75524 of a simple command (see [Section 2.9.1.1](#)). The operands to the command temporarily shall  
 75525 become the positional parameters during the execution of the *compound-command*; the special  
 75526 parameter ‘#’ also shall be changed to reflect the number of operands. The special parameter 0  
 75527 shall be unchanged. When the function completes, the values of the positional parameters and



75528 the special parameter '#' shall be restored to the values they had before the function was  
 75529 executed. If the special built-in *return* (see *return*) is executed in the *compound-command*, the  
 75530 function completes and execution shall resume with the next command after the function call.

## 75531 Exit Status

75532 The exit status of a function definition shall be zero if the function was declared successfully;  
 75533 otherwise, it shall be greater than zero. The exit status of a function invocation shall be the exit  
 75534 status of the last command executed by the function.

## 75535 2.10 Shell Grammar

75536 The following grammar defines the Shell Command Language. This formal syntax shall take  
 75537 precedence over the preceding text syntax description.

### 75538 2.10.1 Shell Grammar Lexical Conventions

75539 The input language to the shell must be first recognized at the character level. The resulting  
 75540 tokens shall be classified by their immediate context according to the following rules (applied in  
 75541 order). These rules shall be used to determine what a "token" is that is subject to parsing at the  
 75542 token level. The rules for token recognition in [Section 2.3](#) shall apply.

- 75543 1. If the token is an operator, the token identifier for that operator shall result.
- 75544 2. If the string consists solely of digits and the delimiter character is one of '<' or '>', the  
 75545 token identifier **IO\_NUMBER** shall be returned.
- 75546 3. Otherwise, the token identifier **TOKEN** results.

75547 Further distinction on **TOKEN** is context-dependent. It may be that the same **TOKEN** yields  
 75548 **WORD**, a **NAME**, an **ASSIGNMENT\_WORD**, or one of the reserved words below, dependent  
 75549 upon the context. Some of the productions in the grammar below are annotated with a rule  
 75550 number from the following list. When a **TOKEN** is seen where one of those annotated  
 75551 productions could be used to reduce the symbol, the applicable rule shall be applied to convert  
 75552 the token identifier type of the **TOKEN** to a token identifier acceptable at that point in the  
 75553 grammar. The reduction shall then proceed based upon the token identifier type yielded by the  
 75554 rule applied. When more than one rule applies, the highest numbered rule shall apply (which in  
 75555 turn may refer to another rule). (Note that except in rule 7, the presence of an '=' in the token  
 75556 has no effect.)

75557 The **WORD** tokens shall have the word expansion rules applied to them immediately before the  
 75558 associated command is executed, not at the time the command is parsed.

### 75559 2.10.2 Shell Grammar Rules

- 75560 1. [Command Name]

75561 When the **TOKEN** is exactly a reserved word, the token identifier for that reserved word  
 75562 shall result. Otherwise, the token **WORD** shall be returned. Also, if the parser is in any  
 75563 state where only a reserved word could be the next correct token, proceed as above.

75564 **Note:** Because at this point <quotation-mark> characters are retained in the token, quoted  
 75565 strings cannot be recognized as reserved words. This rule also implies that reserved  
 75566 words are not recognized except in certain positions in the input, such as after a

<newline> or <semicolon>; the grammar presumes that if the reserved word is intended, it is properly delimited by the user, and does not attempt to reflect that requirement directly. Also note that line joining is done before tokenization, as described in [Section 2.2.1](#), so escaped <newline> characters are already removed at this point.

Rule 1 is not directly referenced in the grammar, but is referred to by other rules, or applies globally.

## 2. [Redirection to or from filename]

The expansions specified in [Section 2.7](#) shall occur. As specified there, exactly one field can result (or the result is unspecified), and there are additional requirements on pathname expansion.

## 3. [Redirection from here-document]

Quote removal shall be applied to the word to determine the delimiter that is used to find the end of the here-document that begins after the next <newline>.

## 4. [Case statement termination]

When the **TOKEN** is exactly the reserved word **esac**, the token identifier for **esac** shall result. Otherwise, the token **WORD** shall be returned.

## 5. [NAME in for]

When the **TOKEN** meets the requirements for a name (see XBD [Section 3.235](#)), the token identifier **NAME** shall result. Otherwise, the token **WORD** shall be returned.

## 6. [Third word of for and case]

### a. [case only]

When the **TOKEN** is exactly the reserved word **in**, the token identifier for **in** shall result. Otherwise, the token **WORD** shall be returned.

### b. [for only]

When the **TOKEN** is exactly the reserved word **in** or **do**, the token identifier for **in** or **do** shall result, respectively. Otherwise, the token **WORD** shall be returned.

(For a. and b.: As indicated in the grammar, a *linebreak* precedes the tokens **in** and **do**. If <newline> characters are present at the indicated location, it is the token after them that is treated in this fashion.)

## 7. [Assignment preceding command name]

### a. [When the first word]

If the **TOKEN** does not contain the character ' = ', rule 1 is applied. Otherwise, 7b shall be applied.

### b. [Not the first word]

If the **TOKEN** contains an unquoted (as determined while applying rule 4 from [Section 2.3](#)) <equals-sign> character that is not part of an embedded parameter expansion, command substitution, or arithmetic expansion construct (as determined while applying rule 5 from [Section 2.3](#)):

— If the **TOKEN** begins with ' = ', then rule 1 shall be applied.

— If all the characters in the **TOKEN** preceding the first such <equals-sign> form a valid name (see XBD [Section 3.235](#), on page 71), the token **ASSIGNMENT\_WORD** shall be returned.

```

75609         — Otherwise, it is unspecified whether rule 1 is applied or |
75610         ASSIGNMENT_WORD is returned. |
75611         Otherwise, rule 1 shall be applied. +
75612         Assignment to the name within a returned ASSIGNMENT_WORD token shall occur as |
75613         specified in Section 2.9.1.
75614     8. [NAME in function]
75615         When the TOKEN is exactly a reserved word, the token identifier for that reserved word
75616         shall result. Otherwise, when the TOKEN meets the requirements for a name, the token
75617         identifier NAME shall result. Otherwise, rule 7 applies.
75618     9. [Body of function]
75619         Word expansion and assignment shall never occur, even when required by the rules
75620         above, when this rule is being parsed. Each TOKEN that might either be expanded or
75621         have assignment applied to it shall instead be returned as a single WORD consisting only
75622         of characters that are exactly the token described in Section 2.3.
75623     /* -----
75624         The grammar symbols
75625         ----- */
75626     %token  WORD
75627     %token  ASSIGNMENT_WORD
75628     %token  NAME
75629     %token  NEWLINE
75630     %token  IO_NUMBER
75631     /* The following are the operators (see XBD Section 3.260) |
75632         containing more than one character. */ |
75633     %token  AND_IF      OR_IF      DSEMI
75634     /*      '&&'      '||'      ';;'      */
75635     %token  DLESS      DGREAT      LESSAND      GREATAND      LESSGREAT      DLESSDASH
75636     /*      '<<'      '>>'      '<&'      '>&'      '<>'      '<<-'      */
75637     %token  CLOBBER
75638     /*      '>|'      */
75639     /* The following are the reserved words. */
75640     %token  If      Then      Else      Elif      Fi      Do      Done
75641     /*      'if'      'then'      'else'      'elif'      'fi'      'do'      'done'      */
75642     %token  Case      Esac      While      Until      For
75643     /*      'case'      'esac'      'while'      'until'      'for'      */
75644     /* These are reserved words, not operator tokens, and are
75645         recognized when reserved words are recognized. */
75646     %token  Lbrace      Rbrace      Bang
75647     /*      '{'      '}'      '!'      */
75648     %token  In
75649     /*      'in'      */
75650     /* -----
75651         The Grammar
75652         ----- */

```

```

75653 %start program
75654 %%
75655 program      : linebreak complete_commands linebreak
75656               | linebreak
75657               ;
75658 complete_commands: complete_commands newline_list complete_command
75659               |
75660               | complete_command
75661               ;
75662 complete_command : list separator_op
75663               | list
75664               ;
75665 list            : list separator_op and_or
75666               |
75667               | and_or
75668               ;
75669 and_or          :
75670               | and_or AND_IF linebreak pipeline
75671               | and_or OR_IF linebreak pipeline
75672               ;
75673 pipeline        : pipe_sequence
75674               | Bang pipe_sequence
75675               ;
75676 pipe_sequence   :
75677               | pipe_sequence '|' linebreak command
75678               ;
75679 command         : simple_command
75680               | compound_command
75681               | compound_command redirect_list
75682               | function_definition
75683               ;
75684 compound_command : brace_group
75685               | subshell
75686               | for_clause
75687               | case_clause
75688               | if_clause
75689               | while_clause
75690               | until_clause
75691               ;
75692 subshell        : '(' compound_list ')'
75693               ;
75694 compound_list   : linebreak term
75695               | linebreak term separator
75696               ;
75697 term           : term separator and_or
75698               |
75699               | and_or
75700               ;
75701 for_clause      : For name
75702               | For name sequential_sep do_group
75703               | For name linebreak in sequential_sep do_group
75704               | For name linebreak in wordlist sequential_sep do_group
75705               ;
75706 name           : NAME /* Apply rule 5 */
75707               ;
75708 in             : In /* Apply rule 6 */

```

```

75706      ;
75707      wordlist      : wordlist WORD
75708                    | WORD
75709      ;
75710      case_clause    : Case WORD linebreak in linebreak case_list      Esac
75711                    | Case WORD linebreak in linebreak case_list_ns Esac
75712                    | Case WORD linebreak in linebreak                  Esac
75713      ;
75714      case_list_ns    : case_list case_item_ns
75715                    | case_item_ns
75716      ;
75717      case_list       : case_list case_item
75718                    | case_item
75719      ;
75720      case_item_ns    : pattern ')' linebreak
75721                    | pattern ')' compound_list
75722                    | '(' pattern ')' linebreak
75723                    | '(' pattern ')' compound_list
75724      ;
75725      case_item       : pattern ')' linebreak      DSEMI linebreak
75726                    | pattern ')' compound_list DSEMI linebreak
75727                    | '(' pattern ')' linebreak   DSEMI linebreak
75728                    | '(' pattern ')' compound_list DSEMI linebreak
75729      ;
75730      pattern         : WORD /* Apply rule 4 */
75731                    | pattern '|' WORD /* Do not apply rule 4 */
75732      ;
75733      if_clause       : If compound_list Then compound_list else_part Fi
75734                    | If compound_list Then compound_list      Fi
75735      ;
75736      else_part       : Elif compound_list Then compound_list
75737                    | Elif compound_list Then compound_list else_part
75738                    | Else compound_list
75739      ;
75740      while_clause    : While compound_list do_group
75741      ;
75742      until_clause    : Until compound_list do_group
75743      ;
75744      function_definition : fname '(' ')' linebreak function_body
75745      ;
75746      function_body    : compound_command /* Apply rule 9 */
75747                    | compound_command redirect_list /* Apply rule 9 */
75748      ;
75749      fname           : NAME /* Apply rule 8 */
75750      ;
75751      brace_group     : Lbrace compound_list Rbrace
75752      ;
75753      do_group        : Do compound_list Done /* Apply rule 6 */
75754      ;
75755      simple_command  : cmd_prefix cmd_word cmd_suffix
75756                    | cmd_prefix cmd_word
75757                    | cmd_prefix
75758                    | cmd_name cmd_suffix

```

```

75759         | cmd_name
75760         ;
75761     cmd_name      : WORD                      /* Apply rule 7a */
75762         ;
75763     cmd_word      : WORD                      /* Apply rule 7b */
75764         ;
75765     cmd_prefix    : io_redirect
75766         | cmd_prefix io_redirect
75767         | ASSIGNMENT_WORD
75768         | cmd_prefix ASSIGNMENT_WORD
75769         ;
75770     cmd_suffix    : io_redirect
75771         | cmd_suffix io_redirect
75772         | WORD
75773         | cmd_suffix WORD
75774         ;
75775     redirect_list : io_redirect
75776         | redirect_list io_redirect
75777         ;
75778     io_redirect   : io_file
75779         | IO_NUMBER io_file
75780         | io_here
75781         | IO_NUMBER io_here
75782         ;
75783     io_file       : '<'      filename
75784         | LESSAND  filename
75785         | '>'      filename
75786         | GREATAND filename
75787         | DGREAT  filename
75788         | LESSGREAT filename
75789         | CLOBBER  filename
75790         ;
75791     filename      : WORD                      /* Apply rule 2 */
75792         ;
75793     io_here       : DLESS      here_end
75794         | DLESSDASH here_end
75795         ;
75796     here_end      : WORD                      /* Apply rule 3 */
75797         ;
75798     newline_list  : NEWLINE
75799         | newline_list NEWLINE
75800         ;
75801     linebreak     : newline_list
75802         | /* empty */
75803         ;
75804     separator_op  : '&'
75805         | ';'
75806         ;
75807     separator     : separator_op linebreak
75808         | newline_list
75809         ;
75810     sequential_sep : ';' linebreak
75811         | newline_list

```

75812 ;

## 75813 2.11 Signals and Error Handling

75814 If job control is disabled (see the description of *set -m*) when the shell executes an asynchronous |  
 75815 list, the commands in the list shall inherit from the shell a signal action of ignored (SIG\_IGN) for |  
 75816 the SIGINT and SIGQUIT signals. In all other cases, commands executed by the shell shall |  
 75817 inherit the same signal actions as those inherited by the shell from its parent unless a signal |  
 75818 action is modified by the *trap* special built-in (see *trap*)

75819 When a signal for which a trap has been set is received while the shell is waiting for the  
 75820 completion of a utility executing a foreground command, the trap associated with that signal  
 75821 shall not be executed until after the foreground command has completed. When the shell is  
 75822 waiting, by means of the *wait* utility, for asynchronous commands to complete, the reception of a  
 75823 signal for which a trap has been set shall cause the *wait* utility to return immediately with an exit  
 75824 status >128, immediately after which the trap associated with that signal shall be taken.

75825 If multiple signals are pending for the shell for which there are associated trap actions, the order  
 75826 of execution of trap actions is unspecified.

## 75827 2.12 Shell Execution Environment

75828 A shell execution environment consists of the following:

- 75829 • Open files inherited upon invocation of the shell, plus open files controlled by *exec*
- 75830 • Working directory as set by *cd*
- 75831 • File creation mask set by *umask*
- 75832 XSI • File size limit as set by *ulimit* +
- 75833 • Current traps set by *trap*
- 75834 • Shell parameters that are set by variable assignment (see the *set* special built-in) or from
- 75835 the System Interfaces volume of POSIX.1-2008 environment inherited by the shell when it
- 75836 begins (see the *export* special built-in)
- 75837 • Shell functions; see [Section 2.9.5](#)
- 75838 • Options turned on at invocation or by *set*
- 75839 • Process IDs of the last commands in asynchronous lists known to this shell environment;
- 75840 see [Section 2.9.3.1](#)
- 75841 • Shell aliases; see [Section 2.3.1](#)

75842 Utilities other than the special built-ins (see [Section 2.14](#)) shall be invoked in a separate  
 75843 environment that consists of the following. The initial value of these objects shall be the same as  
 75844 that for the parent shell, except as noted below.

- 75845 • Open files inherited on invocation of the shell, open files controlled by the *exec* special
- 75846 built-in plus any modifications, and additions specified by any redirections to the utility

- Current working directory
- File creation mask
- If the utility is a shell script, traps caught by the shell shall be set to the default values and traps ignored by the shell shall be set to be ignored by the utility; if the utility is not a shell script, the trap actions (default or ignore) shall be mapped into the appropriate signal handling actions for the utility
- Variables with the *export* attribute, along with those explicitly exported for the duration of the command, shall be passed to the utility environment variables

The environment of the shell process shall not be changed by the utility unless explicitly specified by the utility description (for example, *cd* and *umask*).

A subshell environment shall be created as a duplicate of the shell environment, except that signal traps that are not being ignored shall be set to the default action. Changes made to the subshell environment shall not affect the shell environment. Command substitution, commands that are grouped with parentheses, and asynchronous lists shall be executed in a subshell environment. Additionally, each command of a multi-command pipeline is in a subshell environment; as an extension, however, any or all commands in a pipeline may be executed in the current environment. All other commands shall be executed in the current shell environment.

## 2.13 Pattern Matching Notation

The pattern matching notation described in this section is used to specify patterns for matching strings in the shell. Historically, pattern matching notation is related to, but slightly different from, the regular expression notation described in XBD [Chapter 9](#). For this reason, the description of the rules for this pattern matching notation are based on the description of regular expression notation, modified to account for the differences.

### 2.13.1 Patterns Matching a Single Character

The following patterns matching a single character shall match a single character: ordinary characters, special pattern characters, and pattern bracket expressions. The pattern bracket expression also shall match a single collating element. A `<backslash>` character shall escape the following character. The escaping `<backslash>` shall be discarded. If a pattern ends with an unescaped `<backslash>`, it is unspecified whether the pattern does not match anything or the pattern is treated as invalid.

An ordinary character is a pattern that shall match itself. It can be any character in the supported character set except for NUL, those special shell characters in [Section 2.2](#) that require quoting, and the following three special pattern characters. Matching shall be based on the bit pattern used for encoding the character, not on the graphic representation of the character. If any character (ordinary, shell special, or pattern special) is quoted, that pattern shall match the character itself. The shell special characters always require quoting.

When unquoted and outside a bracket expression, the following three characters shall have special meaning in the specification of patterns:

- ? A `<question-mark>` is a pattern that shall match any character.



\* An <asterisk> is a pattern that shall match multiple characters, as described in [Section 2.13.2](#).

[ If an open bracket introduces a bracket expression as in XBD [Section 9.3.5](#), except that the <exclamation-mark> character ( ' ! ' ) shall replace the <circumflex> character ( ' ^ ' ) in its role in a non-matching list in the regular expression notation, it shall introduce a pattern bracket expression. A bracket expression starting with an unquoted <circumflex> character produces unspecified results. Otherwise, ' [ ' shall match the character itself.

When pattern matching is used where shell quote removal is not performed (such as in the argument to the *find -name* primary when *find* is being called using one of the *exec* functions as defined in the System Interfaces volume of POSIX.1-2008, or in the *pattern* argument to the *fnmatch()* function), special characters can be escaped to remove their special meaning by preceding them with a <backslash> character. This escaping <backslash> is discarded. The sequence "\\\" represents one literal <backslash>. All of the requirements and effects of quoting on ordinary, shell special, and special pattern characters shall apply to escaping in this context.

## 2.13.2 Patterns Matching Multiple Characters

The following rules are used to construct patterns matching multiple characters from patterns matching a single character:

1. The <asterisk> ( ' \* ' ) is a pattern that shall match any string, including the null string.
2. The concatenation of patterns matching a single character is a valid pattern that shall match the concatenation of the single characters or collating elements matched by each of the concatenated patterns.
3. The concatenation of one or more patterns matching a single character with one or more <asterisk> characters is a valid pattern. In such patterns, each <asterisk> shall match a string of zero or more characters, matching the greatest possible number of characters that still allows the remainder of the pattern to match the string.

## 2.13.3 Patterns Used for Filename Expansion

The rules described so far in [Section 2.13.1](#) and [Section 2.13.2](#) are qualified by the following rules that apply when pattern matching notation is used for filename expansion:

1. The <slash> character in a pathname shall be explicitly matched by using one or more <slash> characters in the pattern; it shall neither be matched by the <asterisk> or <question-mark> special characters nor by a bracket expression. <slash> characters in the pattern shall be identified before bracket expressions; thus, a <slash> cannot be included in a pattern bracket expression used for filename expansion. If a <slash> character is found following an unescaped <left-square-bracket> character before a corresponding <right-square-bracket> is found, the open bracket shall be treated as an ordinary character. For example, the pattern "a[b/c]d" does not match such pathnames as **abd** or **a/d**. It only matches a pathname of literally **a[b/c]d**.
2. If a filename begins with a <period> ( ' . ' ), the <period> shall be explicitly matched by using a <period> as the first character of the pattern or immediately following a <slash> character. The leading <period> shall not be matched by:
  - The <asterisk> or <question-mark> special characters

- A bracket expression containing a non-matching list, such as "[!a]", a range expression, such as "[%–0]", or a character class expression, such as "[[:punct:]]"

It is unspecified whether an explicit <period> in a bracket expression matching list, such as "[.abc]", can match a leading <period> in a filename.

3. Specified patterns shall be matched against existing filenames and pathnames, as appropriate. Each component that contains a pattern character shall require read permission in the directory containing that component. Any component, except the last, that does not contain a pattern character shall require search permission. For example, given the pattern:

/foo/bar/x\*/bam

search permission is needed for directories / and **foo**, search and read permissions are needed for directory **bar**, and search permission is needed for each **x\*** directory. If the pattern matches any existing filenames or pathnames, the pattern shall be replaced with those filenames and pathnames, sorted according to the collating sequence in effect in the current locale. If this collating sequence does not have a total ordering of all characters (see XBD [Section 7.3.2](#), on page 147), any filenames or pathnames that collate equally should be further compared byte-by-byte using the collating sequence for the POSIX locale.

**Note:** A future version of this standard may require the byte-by-byte further comparison described above.

If the pattern contains an open bracket ('[') that does not introduce a bracket expression as in XBD [Section 9.3.5](#), it is unspecified whether other unquoted pattern matching characters within the same slash-delimited component of the pattern retain their special meanings or are treated as ordinary characters. For example, the pattern "a\*[b\*]" may match all filenames beginning with 'b' in the directory "a\*" or it may match all filenames beginning with 'b' in all directories with names beginning with 'a' and ending with '['.

If the pattern does not match any existing filenames or pathnames, the pattern string shall be left unchanged.

## 2.14 Special Built-In Utilities

The following “special built-in” utilities shall be supported in the shell command language. The output of each command, if any, shall be written to standard output, subject to the normal redirection and piping possible with all commands.

The term “built-in” implies that the shell can execute the utility directly and does not need to search for it. An implementation may choose to make any utility a built-in; however, the special built-in utilities described here differ from regular built-in utilities in two respects:

1. An error in a special built-in utility may cause a shell executing that utility to abort, while an error in a regular built-in utility shall not cause a shell executing that utility to abort. (See [Section 2.8.1](#) for the consequences of errors on interactive and non-interactive shells.) If a special built-in utility encountering an error does not abort the shell, its exit value shall be non-zero.

- 75970           2. As described in [Section 2.9.1](#), variable assignments preceding the invocation of a special |  
75971           built-in utility remain in effect after the built-in completes; this shall not be the case with a |  
75972           regular built-in or other utility.

75973           The special built-in utilities in this section need not be provided in a manner accessible via the  
75974           *exec* family of functions defined in the System Interfaces volume of POSIX.1-2008.

75975           Some of the special built-ins are described as conforming to XBD [Section 12.2](#). For those that are  
75976           not, the requirement in [Section 1.4](#) that "--" be recognized as a first argument to be discarded  
75977           does not apply and a conforming application shall not use that argument.

75978 **NAME**

75979 break — exit from for, while, or until loop

75980 **SYNOPSIS**75981 break [*n*]75982 **DESCRIPTION**

75983 If *n* is specified, the *break* utility shall exit from the *n*th enclosing **for**, **while**, or **until** loop. If *n* is  
 75984 not specified, *break* shall behave as if *n* was specified as 1. Execution shall continue with the  
 75985 command immediately following the exited loop. The value of *n* is a positive decimal integer. If  
 75986 *n* is greater than the number of enclosing loops, the outermost enclosing loop shall be exited. If  
 75987 there is no enclosing loop, the behavior is unspecified.

75988 A loop shall enclose a *break* or *continue* command if the loop lexically encloses the command. A  
 75989 loop lexically encloses a *break* or *continue* command if the command is:

- 75990 • Executing in the same execution environment (see [Section 2.12](#)) as the compound-list of the  
 75991 loop's do-group (see [Section 2.10.2](#)), and
- 75992 • Contained in a compound-list associated with the loop (either in the compound-list of the  
 75993 loop's do-group or, if the loop is a **while** or **until** loop, in the compound-list following the  
 75994 **while** or **until** reserved word), and
- 75995 • Not in the body of a function whose function definition command (see [Section 2.9.5](#)) is  
 75996 contained in a compound-list associated with the loop.

75997 If *n* is greater than the number of lexically enclosing loops and there is a non-lexically enclosing  
 75998 loop in progress in the same execution environment as the *break* or *continue* command, it is  
 75999 unspecified whether that loop encloses the command.

76000 **OPTIONS**

76001 None.

76002 **OPERANDS**

76003 See the DESCRIPTION.

76004 **STDIN**

76005 Not used.

76006 **INPUT FILES**

76007 None.

76008 **ENVIRONMENT VARIABLES**

76009 None.

76010 **ASYNCHRONOUS EVENTS**

76011 Default.

76012 **STDOUT**

76013 Not used.

76014 **STDERR**

76015 The standard error shall be used only for diagnostic messages.

76016 **OUTPUT FILES**

76017 None.

76018 **EXTENDED DESCRIPTION**

76019       None.

76020 **EXIT STATUS**

76021       0   Successful completion.

76022       >0   The *n* value was not an unsigned decimal integer greater than or equal to 1.76023 **CONSEQUENCES OF ERRORS**

76024       Default.

76025 **APPLICATION USAGE**

76026       None.

76027 **EXAMPLES**

```

76028     for i in *
76029     do
76030         if test -d "$i"
76031         then break
76032         fi
76033     done

```

76034       The results of running the following example are unspecified: there are two loops in progress +  
 76035       when the *break* command is executed, and they are in the same execution environment, but +  
 76036       neither loop is lexically enclosing the *break* command. (There are no loops lexically enclosing the +  
 76037       *continue* commands, either.) +

```

76038     foo() { +
76039         for j in 1 2; do +
76040             echo 'break 2' >/tmp/do_break +
76041             echo "   sourcing /tmp/do_break ($j)..." +
76042             # the behavior of the break from running the following command +
76043             # results in unspecified behavior: +
76044             . /tmp/do_break +
76045 +
76046             do_continue() { continue 2; } +
76047             echo "   running do_continue ($j)..." +
76048             # the behavior of the continue in the following function call +
76049             # results in unspecified behavior (if execution reaches this +
76050             # point): +
76051             do_continue +
76052 +
76053             trap 'continue 2' USR1 +
76054             echo "   sending SIGUSR1 to self ($j)..." +
76055             # the behavior of the continue in the trap invoked from the +
76056             # following signal results in unspecified behavior (if +
76057             # execution reaches this point): +
76058             kill -s USR1 $$ +
76059             sleep 1 +
76060         done +
76061     } +
76062     for i in 1 2; do +
76063         echo "running foo ($i)..." +
76064         foo +
76065     done +

```

**RATIONALE**

In early proposals, consideration was given to expanding the syntax of *break* and *continue* to refer to a label associated with the appropriate loop as a preferable alternative to the *n* method. However, this volume of POSIX.1-2008 does reserve the name space of command names ending with a <colon>. It is anticipated that a future implementation could take advantage of this and provide something like:

```
outofloop: for i in a b c d e
do
    for j in 0 1 2 3 4 5 6 7 8 9
    do
        if test -r "${i}${j}"
        then break outofloop
        fi
    done
done
```

and that this might be standardized after implementation experience is achieved.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 2.14](#)

**CHANGE HISTORY****Issue 6**

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in behavior is intended.

76089 **NAME**

76090 colon — null utility

76091 **SYNOPSIS**76092 : [*argument...*]76093 **DESCRIPTION**

76094 This utility shall only expand command *arguments*. It is used when a command is needed, as in  
 76095 the **then** condition of an **if** command, but nothing is to be done by the command.

76096 **OPTIONS**

76097 None.

76098 **OPERANDS**

76099 See the DESCRIPTION.

76100 **STDIN**

76101 Not used.

76102 **INPUT FILES**

76103 None.

76104 **ENVIRONMENT VARIABLES**

76105 None.

76106 **ASYNCHRONOUS EVENTS**

76107 Default.

76108 **STDOUT**

76109 Not used.

76110 **STDERR**

76111 The standard error shall be used only for diagnostic messages.

76112 **OUTPUT FILES**

76113 None.

76114 **EXTENDED DESCRIPTION**

76115 None.

76116 **EXIT STATUS**

76117 Zero.

76118 **CONSEQUENCES OF ERRORS**

76119 Default.

76120 **APPLICATION USAGE**

76121 None.

76122 **EXAMPLES**

```
76123 : ${X=abc}
76124 if     false
76125 then  :
76126 else  echo $X
76127 fi
76128 abc
```

76129 As with any of the special built-ins, the null utility can also have variable assignments and  
 76130 redirections associated with it, such as:

76131 **x=y : > z**

76132 which sets variable *x* to the value *y* (so that it persists after the null utility completes) and creates  
76133 or truncates file *z*.

76134 **RATIONALE**

76135 None.

76136 **FUTURE DIRECTIONS**

76137 None.

76138 **SEE ALSO**

76139 [Section 2.14](#)

76140 **CHANGE HISTORY**

76141 **Issue 6**

76142 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page  
76143 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in  
76144 behavior is intended.

76145 **Issue 7**

76146 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



76147 **NAME**

76148           continue — continue for, while, or until loop

76149 **SYNOPSIS**76150           continue [*n*]76151 **DESCRIPTION**

76152           If *n* is specified, the *continue* utility shall return to the top of the *n*th enclosing **for**, **while**, or **until** |  
 76153           loop. If *n* is not specified, *continue* shall behave as if *n* was specified as 1. Returning to the top of |  
 76154           the loop involves repeating the condition list of a **while** or **until** loop or performing the next |  
 76155           assignment of a **for** loop, and re-executing the loop if appropriate.

76156           The value of *n* is a positive decimal integer. If *n* is greater than the number of enclosing loops, |  
 76157           the outermost enclosing loop shall be used. If there is no enclosing loop, the behavior is |  
 76158           unspecified.

76159           The meaning of “enclosing” shall be as specified in the description of the *break* utility.

76160 **OPTIONS**

76161           None.

76162 **OPERANDS**

76163           See the DESCRIPTION.

76164 **STDIN**

76165           Not used.

76166 **INPUT FILES**

76167           None.

76168 **ENVIRONMENT VARIABLES**

76169           None.

76170 **ASYNCHRONOUS EVENTS**

76171           Default.

76172 **STDOUT**

76173           Not used.

76174 **STDERR**

76175           The standard error shall be used only for diagnostic messages.

76176 **OUTPUT FILES**

76177           None.

76178 **EXTENDED DESCRIPTION**

76179           None.

76180 **EXIT STATUS**

76181           0   Successful completion.

76182           >0   The *n* value was not an unsigned decimal integer greater than or equal to 1.76183 **CONSEQUENCES OF ERRORS**

76184           Default.

76185 **APPLICATION USAGE**

76186 None.

76187 **EXAMPLES**

```
76188     for i in *
76189     do
76190         if test -d "$i"
76191         then continue
76192         fi
76193         printf "%s" is not a directory.\n' "$i"
76194     done
```

76195 **RATIONALE**

76196 None.

76197 **FUTURE DIRECTIONS**

76198 None.

76199 **SEE ALSO**76200 [Section 2.14](#)76201 **CHANGE HISTORY**76202 **Issue 6**

76203 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page  
76204 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in  
76205 behavior is intended.

76206 **Issue 7**

76207 The example is changed to use the *printf* utility rather than *echo*.

76208 **NAME**

76209 dot — execute commands in the current environment

76210 **SYNOPSIS**76211 . *file*76212 **DESCRIPTION**76213 The shell shall execute commands from the *file* in the current environment.

76214 If *file* does not contain a <slash>, the shell shall use the search path specified by *PATH* to find the  
 76215 directory containing *file*. Unlike normal command search, however, the file searched for by the  
 76216 *dot* utility need not be executable. If no readable file is found, a non-interactive shell shall abort;  
 76217 an interactive shell shall write a diagnostic message to standard error, but this condition shall  
 76218 not be considered a syntax error.

76219 **OPTIONS**

76220 None.

76221 **OPERANDS**

76222 See the DESCRIPTION.

76223 **STDIN**

76224 Not used.

76225 **INPUT FILES**

76226 See the DESCRIPTION.

76227 **ENVIRONMENT VARIABLES**

76228 See the DESCRIPTION.

76229 **ASYNCHRONOUS EVENTS**

76230 Default.

76231 **STDOUT**

76232 Not used.

76233 **STDERR**

76234 The standard error shall be used only for diagnostic messages.

76235 **OUTPUT FILES**

76236 None.

76237 **EXTENDED DESCRIPTION**

76238 None.

76239 **EXIT STATUS**

76240 If no readable file was found or if the commands in the file could not be parsed, and the shell is  
 76241 interactive (and therefore does not abort; see [Section 2.8.1](#)), the exit status shall be non-zero.  
 76242 Otherwise, return the value of the last command executed, or a zero exit status if no command is  
 76243 executed.

76244 **CONSEQUENCES OF ERRORS**

76245 Default.

**76246 APPLICATION USAGE**

76247 None.

**76248 EXAMPLES**

```
76249 cat foobar
76250 foo=hello bar=world
76251 . ./foobar
76252 echo $foo $bar
76253 hello world
```

**76254 RATIONALE**

76255 Some older implementations searched the current directory for the *file*, even if the value of *PATH*  
76256 disallowed it. This behavior was omitted from this volume of POSIX.1-2008 due to concerns  
76257 about introducing the susceptibility to trojan horses that the user might be trying to avoid by  
76258 leaving **dot** out of *PATH*.

76259 The KornShell version of *dot* takes optional arguments that are set to the positional parameters.  
76260 This is a valid extension that allows a *dot* script to behave identically to a function.

**76261 FUTURE DIRECTIONS**

76262 None.

**76263 SEE ALSO**

76264 [Section 2.14, \*return\*](#)

**76265 CHANGE HISTORY****76266 Issue 6**

76267 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page  
76268 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in  
76269 behavior is intended.

**76270 Issue 7**

76271 SD5-XCU-ERN-164 is applied.

76272 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0038 [114] and XCU/TC1-2008/0039  
76273 [214] are applied.

76274 **NAME**

76275 eval — construct command by concatenating arguments

76276 **SYNOPSIS**76277 eval [*argument...*]76278 **DESCRIPTION**76279 The *eval* utility shall construct a command by concatenating *arguments* together, separating each  
76280 with a <space> character. The constructed command shall be read and executed by the shell.76281 **OPTIONS**

76282 None.

76283 **OPERANDS**

76284 See the DESCRIPTION.

76285 **STDIN**

76286 Not used.

76287 **INPUT FILES**

76288 None.

76289 **ENVIRONMENT VARIABLES**

76290 None.

76291 **ASYNCHRONOUS EVENTS**

76292 Default.

76293 **STDOUT**

76294 Not used.

76295 **STDERR**

76296 The standard error shall be used only for diagnostic messages.

76297 **OUTPUT FILES**

76298 None.

76299 **EXTENDED DESCRIPTION**

76300 None.

76301 **EXIT STATUS**76302 If there are no *arguments*, or only null *arguments*, *eval* shall return a zero exit status; otherwise, it  
76303 shall return the exit status of the command defined by the string of concatenated *arguments*  
76304 separated by <space> characters, or a non-zero exit status if the concatenation could not be  
76305 parsed as a command and the shell is interactive (and therefore did not abort).76306 **CONSEQUENCES OF ERRORS**

76307 Default.

76308 **APPLICATION USAGE**76309 Since *eval* is not required to recognize the "--" end of options delimiter, in cases where the  
76310 argument(s) to *eval* might begin with '-' it is recommended that the first argument is prefixed  
76311 by a string that will not alter the commands to be executed, such as a <space> character:

76312 eval " \$commands"

76313 or:

76314 eval " \$(some\_command)"

**EXAMPLES**

```
foo=10 x=foo
y=' '$x
echo $y
$foo
eval y=' '$x
echo $y
10
```

**RATIONALE**

This standard allows, but does not require, *eval* to recognize "--". Although this means applications cannot use "--" to protect against options supported as an extension (or errors reported for unsupported options), the nature of the *eval* utility is such that other means can be used to provide this protection (see APPLICATION USAGE above).

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 2.14](#)

**CHANGE HISTORY****Issue 6**

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in behavior is intended.

**Issue 7**

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0040 [114], XCU/TC1-2008/0041 [163], and XCU/TC1-2008/0042 [163] are applied.

76341 **NAME**76342        *exec* — execute commands and open, close, or copy file descriptors76343 **SYNOPSIS**76344        *exec* [*command* [*argument...*]]76345 **DESCRIPTION**76346        The *exec* utility shall open, close, and/or copy file descriptors as specified by any redirections as  
76347        part of the command.76348        If *exec* is specified without *command* or *arguments*, and any file descriptors with numbers greater  
76349        than 2 are opened with associated redirection statements, it is unspecified whether those file  
76350        descriptors remain open when the shell invokes another utility. Scripts concerned that child  
76351        shells could misuse open file descriptors can always close them explicitly, as shown in one of the  
76352        following examples.76353        If *exec* is specified with *command*, it shall replace the shell with *command* without creating a new  
76354        process. If *arguments* are specified, they shall be arguments to *command*. Redirection affects the  
76355        current shell execution environment.76356 **OPTIONS**

76357        None.

76358 **OPERANDS**

76359        See the DESCRIPTION.

76360 **STDIN**

76361        Not used.

76362 **INPUT FILES**

76363        None.

76364 **ENVIRONMENT VARIABLES**

76365        None.

76366 **ASYNCHRONOUS EVENTS**

76367        Default.

76368 **STDOUT**

76369        Not used.

76370 **STDERR**

76371        The standard error shall be used only for diagnostic messages.

76372 **OUTPUT FILES**

76373        None.

76374 **EXTENDED DESCRIPTION**

76375        None.

76376 **EXIT STATUS**76377        If *command* is specified, *exec* shall not return to the shell; rather, the exit status of the process shall  
76378        be the exit status of the program implementing *command*, which overlaid the shell. If *command* is  
76379        not found, the exit status shall be 127. If *command* is found, but it is not an executable utility, the  
76380        exit status shall be 126. If a redirection error occurs (see [Section 2.8.1](#)), the shell shall exit with a  
76381        value in the range 1–125. Otherwise, *exec* shall return a zero exit status.

76382 **CONSEQUENCES OF ERRORS**

76383 Default.

76384 **APPLICATION USAGE**

76385 None.

76386 **EXAMPLES**76387 Open *readfile* as file descriptor 3 for reading:76388 `exec 3< readfile`76389 Open *writefile* as file descriptor 4 for writing:76390 `exec 4> writefile`

76391 Make file descriptor 5 a copy of file descriptor 0:

76392 `exec 5<&0`

76393 Close file descriptor 3:

76394 `exec 3<&-`76395 Cat the file **maggie** by replacing the current shell with the *cat* utility:76396 `exec cat maggie`76397 **RATIONALE**

76398 Most historical implementations were not conformant in that:

76399 `foo=bar exec cmd`76400 did not pass **foo** to **cmd**.76401 **FUTURE DIRECTIONS**

76402 None.

76403 **SEE ALSO**76404 [Section 2.14](#)76405 **CHANGE HISTORY**76406 **Issue 6**

76407 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page  
76408 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in  
76409 behavior is intended.

76410 **Issue 7**

76411 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



76412 **NAME**

76413       exit — cause the shell to exit

76414 **SYNOPSIS**76415       exit [*n*]76416 **DESCRIPTION**

76417       The *exit* utility shall cause the shell to exit from its current execution environment with the exit  
 76418       status specified by the unsigned decimal integer *n*. If the current execution environment is a  
 76419       subshell environment, the shell shall exit from the subshell environment with the specified exit  
 76420       status and continue in the environment from which that subshell environment was invoked;  
 76421       otherwise, the shell utility shall terminate with the specified exit status. If *n* is specified, but its  
 76422       value is not between 0 and 255 inclusively, the exit status is undefined.

76423       A *trap* on **EXIT** shall be executed before the shell terminates, except when the *exit* utility is  
 76424       invoked in that *trap* itself, in which case the shell shall exit immediately.

76425 **OPTIONS**

76426       None.

76427 **OPERANDS**

76428       See the DESCRIPTION.

76429 **STDIN**

76430       Not used.

76431 **INPUT FILES**

76432       None.

76433 **ENVIRONMENT VARIABLES**

76434       None.

76435 **ASYNCHRONOUS EVENTS**

76436       Default.

76437 **STDOUT**

76438       Not used.

76439 **STDERR**

76440       The standard error shall be used only for diagnostic messages.

76441 **OUTPUT FILES**

76442       None.

76443 **EXTENDED DESCRIPTION**

76444       None.

76445 **EXIT STATUS**

76446       The exit status shall be *n*, if specified, except that the behavior is unspecified if *n* is not an  
 76447       unsigned decimal integer or is greater than 255. Otherwise, the value shall be the exit value of  
 76448       the last command executed, or zero if no command was executed. When *exit* is executed in a *trap*  
 76449       action, the last command is considered to be the command that executed immediately preceding  
 76450       the *trap* action.

76451 **CONSEQUENCES OF ERRORS**

76452       Default.

**APPLICATION USAGE**

None.

**EXAMPLES**

Exit with a *true* value:

```
exit 0
```

Exit with a *false* value:

```
exit 1
```

Propagate error handling from within a subshell:

```
(
    command1 || exit 1
    command2 || exit 1
    exec command3
) > outputfile || exit 1
echo "outputfile created successfully"
```

**RATIONALE**

As explained in other sections, certain exit status values have been reserved for special uses and should be used by applications only for those purposes:

126 A file to be executed was found, but it was not an executable utility.

127 A utility to be executed was not found.

>128 A command was interrupted by a signal.

The behavior of *exit* when given an invalid argument or unknown option is unspecified, because of differing practices in the various historical implementations. A value larger than 255 might be truncated by the shell, and be unavailable even to a parent process that uses *waitid()* to get the full exit value. It is recommended that implementations that detect any usage error should cause a non-zero exit status (or, if the shell is interactive and the error does not cause the shell to abort, store a non-zero value in "\$?"), but even this was not done historically in all shells.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 2.14](#)

**CHANGE HISTORY****Issue 6**

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in behavior is intended.

76488 **NAME**

76489       export — set the export attribute for variables

76490 **SYNOPSIS**

76491       export name[=word]...

76492       export -p

76493 **DESCRIPTION**

76494       The shell shall give the *export* attribute to the variables corresponding to the specified *names*,  
 76495       which shall cause them to be in the environment of subsequently executed commands. If the  
 76496       name of a variable is followed by =*word*, then the value of that variable shall be set to *word*.

76497       The *export* special built-in shall support XBD [Section 12.2](#).

76498       When **-p** is specified, *export* shall write to the standard output the names and values of all  
 76499       exported variables, in the following format:

76500       "export %s=%s\n", &lt;name&gt;, &lt;value&gt;

76501       if *name* is set, and:

76502       "export %s\n", &lt;name&gt;

76503       if *name* is unset.

76504       The shell shall format the output, including the proper use of quoting, so that it is suitable for  
 76505       reinput to the shell as commands that achieve the same exporting results, except:

- 76506       1. Read-only variables with values cannot be reset.
- 76507       2. Variables that were unset at the time they were output need not be reset to the unset state  
 76508       if a value is assigned to the variable between the time the state was saved and the time at  
 76509       which the saved output is reinput to the shell.

76510       When no arguments are given, the results are unspecified. |

76511 **OPTIONS**

76512       See the DESCRIPTION.

76513 **OPERANDS**

76514       See the DESCRIPTION.

76515 **STDIN**

76516       Not used.

76517 **INPUT FILES**

76518       None.

76519 **ENVIRONMENT VARIABLES**

76520       None.

76521 **ASYNCHRONOUS EVENTS**

76522       Default.

76523 **STDOUT**

76524       See the DESCRIPTION.

76525 **STDERR**

76526       The standard error shall be used only for diagnostic messages.

76527 **OUTPUT FILES**

76528 None.

76529 **EXTENDED DESCRIPTION**

76530 None.

76531 **EXIT STATUS**76532 0 All *name* operands were successfully exported. |76533 >0 At least one *name* could not be exported, or the **-p** option was specified and an error |  
76534 occurred. |76535 **CONSEQUENCES OF ERRORS**

76536 Default.

76537 **APPLICATION USAGE**76538 Note that, unless *X* was previously marked readonly, the value of "\$?" after: |76539 `export X=$(false)` |76540 will be 0 (because *export* successfully set *X* to the empty string) and that execution continues, |  
76541 even if *set -e* is in effect. In order to detect command substitution failures, a user must separate |  
76542 the assignment from the export, as in: |76543 `X=$(false)` |76544 `export X` |76545 **EXAMPLES**76546 Export *PWD* and *HOME* variables:76547 `export PWD HOME`76548 Set and export the *PATH* variable:76549 `export PATH=/local/bin:$PATH`

76550 Save and restore all exported variables:

76551 `export -p > temp-file`76552 `unset a lot of variables`76553 `... processing`76554 `. temp-file`76555 **RATIONALE**76556 Some historical shells use the no-argument case as the functional equivalent of what is required  
76557 here with **-p**. This feature was left unspecified because it is not historical practice in all shells,  
76558 and some scripts may rely on the now-unspecified results on their implementations. Attempts to  
76559 specify the **-p** output as the default case were unsuccessful in achieving consensus. The **-p**  
76560 option was added to allow portable access to the values that can be saved and then later restored  
76561 using; for example, a *dot* script.76562 **FUTURE DIRECTIONS**

76563 None.

76564 **SEE ALSO**76565 [Section 2.14](#)76566 [XBD Section 12.2](#)

76567 **CHANGE HISTORY**

76568 **Issue 6**

76569 IEEE PASC Interpretation 1003.2 #203 is applied, clarifying the format when a variable is unset.

76570 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page  
76571 sections use terms as described in the Utility Description Defaults (Section 1.4). No change in  
76572 behavior is intended.

76573 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/6 is applied, adding the following text to  
76574 the end of the first paragraph of the DESCRIPTION: "If the name of a variable is followed by  
76575 *=word*, then the value of that variable shall be set to *word*.". The reason for this change is that the  
76576 SYNOPSIS for *export* includes:

76577 `export name[=word]...`

76578 but the meaning of the optional "*=word*" is never explained in the text.

76579 **Issue 7**

76580 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0043 [352] is applied.

76581 **NAME**

76582       readonly — set the readonly attribute for variables

76583 **SYNOPSIS**76584       readonly name[=*word*]...

76585       readonly -p

76586 **DESCRIPTION**

76587       The variables whose *names* are specified shall be given the *readonly* attribute. The values of  
76588       variables with the *readonly* attribute cannot be changed by subsequent assignment, nor can those  
76589       variables be unset by the *unset* utility. If the name of a variable is followed by =*word*, then the  
76590       value of that variable shall be set to *word*.

76591       The *readonly* special built-in shall support XBD [Section 12.2](#).

76592       When -p is specified, *readonly* writes to the standard output the names and values of all read-  
76593       only variables, in the following format:

76594       "readonly %s=%s\n", <*name*>, <*value*>76595       if *name* is set, and76596       "readonly %s\n", <*name*>76597       if *name* is unset.

76598       The shell shall format the output, including the proper use of quoting, so that it is suitable for  
76599       reinput to the shell as commands that achieve the same value and *readonly* attribute-setting  
76600       results in a shell execution environment in which:

- 76601           1. Variables with values at the time they were output do not have the *readonly* attribute set.
- 76602           2. Variables that were unset at the time they were output do not have a value at the time at  
76603           which the saved output is reinput to the shell.

76604       When no arguments are given, the results are unspecified.

76605 **OPTIONS**

76606       See the DESCRIPTION.

76607 **OPERANDS**

76608       See the DESCRIPTION.

76609 **STDIN**

76610       Not used.

76611 **INPUT FILES**

76612       None.

76613 **ENVIRONMENT VARIABLES**

76614       None.

76615 **ASYNCHRONOUS EVENTS**

76616       Default.

76617 **STDOUT**

76618       See the DESCRIPTION.

76619 **STDERR**

76620 The standard error shall be used only for diagnostic messages.

76621 **OUTPUT FILES**

76622 None.

76623 **EXTENDED DESCRIPTION**

76624 None.

76625 **EXIT STATUS**

76626 0 All *name* operands were successfully marked readonly. |

76627 >0 At least one *name* could not be marked readonly, or the **-p** option was specified and an error |  
76628 occurred. |

76629 **CONSEQUENCES OF ERRORS**

76630 Default.

76631 **APPLICATION USAGE**

76632 None.

76633 **EXAMPLES**

76634 readonly HOME PWD

76635 **RATIONALE**

76636 Some historical shells preserve the *readonly* attribute across separate invocations. This volume of  
76637 POSIX.1-2008 allows this behavior, but does not require it.

76638 The **-p** option allows portable access to the values that can be saved and then later restored  
76639 using, for example, a *dot* script. Also see the RATIONALE for *export* for a description of the no-  
76640 argument and **-p** output cases and a related example.

76641 Read-only functions were considered, but they were omitted as not being historical practice or  
76642 particularly useful. Furthermore, functions must not be read-only across invocations to preclude  
76643 “spoofing” (spoofing is the term for the practice of creating a program that acts like a well-  
76644 known utility with the intent of subverting the real intent of the user) of administrative or  
76645 security-relevant (or security-conscious) shell scripts.

76646 **FUTURE DIRECTIONS**

76647 None.

76648 **SEE ALSO**

76649 [Section 2.14](#)

76650 [XBD Section 12.2](#)

76651 **CHANGE HISTORY**

76652 **Issue 6**

76653 IEEE PASC Interpretation 1003.2 #203 is applied, clarifying the format when a variable is unset.

76654 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page  
76655 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in  
76656 behavior is intended.

76657 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/7 is applied, adding the following text to  
76658 the end of the first paragraph of the DESCRIPTION: “If the name of a variable is followed by  
76659 *=word*, then the value of that variable shall be set to *word*.”. The reason for this change is that the  
76660 SYNOPSIS for *readonly* includes:

76661        `readonly name[=word]...`

76662        but the meaning of the optional “=word” is never explained in the text.



76663 **NAME**

76664       return — return from a function or dot script

76665 **SYNOPSIS**76666       return [*n*]76667 **DESCRIPTION**

76668       The *return* utility shall cause the shell to stop executing the current function or *dot* script. If the  
 76669       shell is not currently executing a function or *dot* script, the results are unspecified.

76670 **OPTIONS**

76671       None.

76672 **OPERANDS**

76673       See the DESCRIPTION.

76674 **STDIN**

76675       Not used.

76676 **INPUT FILES**

76677       None.

76678 **ENVIRONMENT VARIABLES**

76679       None.

76680 **ASYNCHRONOUS EVENTS**

76681       Default.

76682 **STDOUT**

76683       Not used.

76684 **STDERR**

76685       The standard error shall be used only for diagnostic messages.

76686 **OUTPUT FILES**

76687       None.

76688 **EXTENDED DESCRIPTION**

76689       None.

76690 **EXIT STATUS**

76691       The value of the special parameter '?' shall be set to *n*, an unsigned decimal integer, or to the  
 76692       exit status of the last command executed if *n* is not specified. If *n* is not an unsigned decimal  
 76693       integer, or is greater than 255, the results are unspecified. When *return* is executed in a *trap*  
 76694       action, the last command is considered to be the command that executed immediately preceding  
 76695       the *trap* action.

76696 **CONSEQUENCES OF ERRORS**

76697       Default.

76698 **APPLICATION USAGE**

76699       None.

76700 **EXAMPLES**

76701       None.

76702 **RATIONALE**

76703       The behavior of *return* when not in a function or *dot* script differs between the System V shell  
 76704       and the KornShell. In the System V shell this is an error, whereas in the KornShell, the effect is  
 76705       the same as *exit*.

76706 The results of returning a number greater than 255 are undefined because of differing practices  
76707 in the various historical implementations. Some shells AND out all but the low-order 8 bits;  
76708 others allow larger values, but not of unlimited size.

76709 See the discussion of appropriate exit status values under *exit*.

#### 76710 FUTURE DIRECTIONS

76711 None.

#### 76712 SEE ALSO

76713 [Section 2.9.5](#), [Section 2.14](#), *dot*

#### 76714 CHANGE HISTORY

##### 76715 Issue 6

76716 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page  
76717 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in  
76718 behavior is intended.

##### 76719 Issue 7

76720 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0044 [214] and XCU/TC1-2008/0045  
76721 [214] are applied.

**NAME**

set — set or unset options and positional parameters

**SYNOPSIS**

set [-abCefhmnvux] [-o *option*] [*argument...*]

set [+abCefhmnvux] [+o *option*] [*argument...*]

set -- [*argument...*]

set -o

set +o

**DESCRIPTION**

If no *options* or *arguments* are specified, *set* shall write the names and values of all shell variables in the collation sequence of the current locale. Each *name* shall start on a separate line, using the format:

"%s=%s\n", <name>, <value>

The *value* string shall be written with appropriate quoting; see the description of shell quoting in [Section 2.2](#). The output shall be suitable for reinput to the shell, setting or resetting, as far as possible, the variables that are currently set; read-only variables cannot be reset.

When options are specified, they shall set or unset attributes of the shell, as described below. When *arguments* are specified, they cause positional parameters to be set or unset, as described below. Setting or unsetting attributes and positional parameters are not necessarily related actions, but they can be combined in a single invocation of *set*.

The *set* special built-in shall support XBD [Section 12.2](#) except that options can be specified with either a leading <hyphen-minus> (meaning enable the option) or <plus-sign> (meaning disable it) unless otherwise specified.

Implementations shall support the options in the following list in both their <hyphen-minus> and <plus-sign> forms. These options can also be specified as options to *sh*.

–a When this option is on, the *export* attribute shall be set for each variable to which an assignment is performed; see XBD [Section 4.23](#). If the assignment precedes a utility name in a command, the *export* attribute shall not persist in the current execution environment after the utility completes, with the exception that preceding one of the special built-in utilities causes the *export* attribute to persist after the built-in has completed. If the assignment does not precede a utility name in the command, or if the assignment is a result of the operation of the *getopts* or *read* utilities, the *export* attribute shall persist until the variable is unset.

–b This option shall be supported if the implementation supports the User Portability Utilities option. It shall cause the shell to notify the user asynchronously of background job completions. The following message is written to standard error:

"[%d]%c %s\n", <job-number>, <current>, <status>, <job-name>

where the fields shall be as follows:

|           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <current> | The character '+' identifies the job that would be used as a default for the <i>fg</i> or <i>bg</i> utilities; this job can also be specified using the <i>job_id</i> "%+" or "%%". The character '-' identifies the job that would become the default if the current default job were to exit; this job can also be specified using the <i>job_id</i> "%-". For other jobs, this field is a <space>. At most one job can be identified with '+' and at most one job can be identified with '-'. If there is any suspended job, then the current job shall be a suspended |
|-----------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

76766 job. If there are at least two suspended jobs, then the previous job also  
76767 shall be a suspended job.

76768 `<job-number>` A number that can be used to identify the process group to the *wait*, *fg*, *bg*,  
76769 and *kill* utilities. Using these utilities, the job can be identified by prefixing  
76770 the job number with ' % '.

76771 `<status>` Unspecified.

76772 `<job-name>` Unspecified.

76773 When the shell notifies the user a job has been completed, it may remove the job's process  
76774 ID from the list of those known in the current shell execution environment; see [Section](#)  
76775 [2.9.3.1](#). Asynchronous notification shall not be enabled by default.

76776 **-C** (Uppercase C.) Prevent existing files from being overwritten by the shell's '`>`' redirection  
76777 operator (see [Section 2.7.2](#)); the '`>|`' redirection operator shall override this *noclobber*  
76778 option for an individual file.

76779 **-e** When this option is on, when any command fails (for any of the reasons listed in [Section](#)  
76780 [2.8.1](#) or by returning an exit status greater than zero), the shell immediately shall exit, as if  
76781 by executing the *exit* special built-in utility with no arguments, with the following  
76782 exceptions:

- 76783 1. The failure of any individual command in a multi-command pipeline shall not cause  
76784 the shell to exit. Only the failure of the pipeline itself shall be considered.
- 76785 2. The **-e** setting shall be ignored when executing the compound list following the  
76786 **while**, **until**, **if**, or **elif** reserved word, a pipeline beginning with the **!** reserved word,  
76787 or any command of an AND-OR list other than the last.
- 76788 3. If the exit status of a compound command other than a subshell command was the  
76789 result of a failure while **-e** was being ignored, then **-e** shall not apply to this  
76790 command.

76791 This requirement applies to the shell environment and each subshell environment  
76792 separately. For example, in:

76793 `set -e; (false; echo one) | cat; echo two`

76794 the *false* command causes the subshell to exit without executing `echo one`; however, `echo`  
76795 `two` is executed because the exit status of the pipeline `(false; echo one) | cat` is  
76796 zero.

76797 **-f** The shell shall disable pathname expansion.

76798 **-h** Locate and remember utilities invoked by functions as those functions are defined (the  
76799 utilities are normally located when the function is executed).

76800 **-m** This option shall be supported if the implementation supports the User Portability Utilities  
76801 option. All jobs shall be run in their own process groups. Immediately before the shell issues  
76802 a prompt after completion of the background job, a message reporting the exit status of the  
76803 background job shall be written to standard error. If a foreground job stops, the shell shall  
76804 write a message to standard error to that effect, formatted as described by the *jobs* utility. In  
76805 addition, if a job changes status other than exiting (for example, if it stops for input or  
76806 output or is stopped by a SIGSTOP signal), the shell shall write a similar message  
76807 immediately prior to writing the next prompt. This option is enabled by default for  
76808 interactive shells.

- 76809        **-n** The shell shall read commands but does not execute them; this can be used to check for  
 76810        shell script syntax errors. An interactive shell may ignore this option.
- 76811        **-o** Write the current settings of the options to standard output in an unspecified format.
- 76812        **+o** Write the current option settings to standard output in a format that is suitable for reinput  
 76813        to the shell as commands that achieve the same options settings.
- 76814        **-o option**
- 76815        This option is supported if the system supports the User Portability Utilities option. It shall  
 76816        set various options, many of which shall be equivalent to the single option letters. The  
 76817        following values of *option* shall be supported:
- 76818        *allexport*        Equivalent to **-a**.
- 76819        *errexit*        Equivalent to **-e**.
- 76820        *ignoreeof*        Prevent an interactive shell from exiting on end-of-file. This setting prevents  
 76821        accidental logouts when <control>-D is entered. A user shall explicitly *exit* to  
 76822        leave the interactive shell.
- 76823        *monitor*        Equivalent to **-m**. This option is supported if the system supports the User  
 76824        Portability Utilities option.
- 76825        *noclobber*        Equivalent to **-C** (uppercase C).
- 76826        *noglob*        Equivalent to **-f**.
- 76827        *noexec*        Equivalent to **-n**.
- 76828        *nolog*        Prevent the entry of function definitions into the command history; see  
 76829        [Command History List](#).
- 76830        *notify*        Equivalent to **-b**.
- 76831        *nounset*        Equivalent to **-u**.
- 76832        *verbose*        Equivalent to **-v**.
- 76833        *vi*        Allow shell command line editing using the built-in *vi* editor. Enabling *vi*  
 76834        mode shall disable any other command line editing mode provided as an  
 76835        implementation extension.
- 76836        It need not be possible to set *vi* mode on for certain block-mode terminals.
- 76837        *xtrace*        Equivalent to **-x**.
- 76838        **-u** When the shell tries to expand an unset parameter other than the '@' and '\*' special  
 76839        parameters, it shall write a message to standard error and the expansion shall fail with the  
 76840        consequences specified in [Section 2.8.1](#).
- 76841        **-v** The shell shall write its input to standard error as it is read.
- 76842        **-x** The shell shall write to standard error a trace for each command after it expands the  
 76843        command and before it executes it. It is unspecified whether the command that turns  
 76844        tracing off is traced.
- 76845        The default for all these options shall be off (unset) unless stated otherwise in the description of  
 76846        the option or unless the shell was invoked with them on; see *sh*.
- 76847        The remaining arguments shall be assigned in order to the positional parameters. The special  
 76848        parameter '#' shall be set to reflect the number of positional parameters. All positional  
 76849        parameters shall be unset before any new values are assigned.

76850 If the first argument is `'-'`, the results are unspecified.

76851 The special argument `--` immediately following the `set` command name can be used to

76852 delimit the arguments if the first argument begins with `'+'` or `'-'`, or to prevent inadvertent

76853 listing of all shell variables when there are no arguments. The command `set --` without *argument*

76854 shall unset all positional parameters and set the special parameter `'#'` to zero.

#### 76855 OPTIONS

76856 See the DESCRIPTION.

#### 76857 OPERANDS

76858 See the DESCRIPTION.

#### 76859 STDIN

76860 Not used.

#### 76861 INPUT FILES

76862 None.

#### 76863 ENVIRONMENT VARIABLES

76864 None.

#### 76865 ASYNCHRONOUS EVENTS

76866 Default.

#### 76867 STDOUT

76868 See the DESCRIPTION.

#### 76869 STDERR

76870 The standard error shall be used only for diagnostic messages.

#### 76871 OUTPUT FILES

76872 None.

#### 76873 EXTENDED DESCRIPTION

76874 None.

#### 76875 EXIT STATUS

76876 0 Successful completion. |

76877 >0 An invalid option was specified, or an error occurred. |

#### 76878 CONSEQUENCES OF ERRORS

76879 Default.

#### 76880 APPLICATION USAGE

76881 Application writers should avoid relying on `set -e` within functions. For example, in the

76882 following script:

```
76883 set -e
76884 start() {
76885     some_server
76886     echo some_server started successfully
76887 }
76888 start || echo >&2 some_server failed
```

76889 the `-e` setting is ignored within the function body (because the function is a command in an

76890 AND-OR list other than the last). Therefore, if `some_server` fails, the function carries on to

76891 echo "some\_server started successfully", and the exit status of the function is zero

76892 (which means "some\_server failed" is not output).

**EXAMPLES**

Write out all variables and their values:

```
set
```

Set \$1, \$2, and \$3 and set "\$#" to 3:

```
set c a b
```

Turn on the `-x` and `-v` options:

```
set -xv
```

Unset all positional parameters:

```
set --
```

Set \$1 to the value of *x*, even if it begins with `'-'` or `'+'`:

```
set -- "$x"
```

Set the positional parameters to the expansion of *x*, even if *x* expands with a leading `'-'` or `'+'`:

```
set -- $x
```

**RATIONALE**

The `set --` form is listed specifically in the SYNOPSIS even though this usage is implied by the Utility Syntax Guidelines. The explanation of this feature removes any ambiguity about whether the `set --` form might be misinterpreted as being equivalent to `set` without any options or arguments. The functionality of this form has been adopted from the KornShell. In System V, `set --` only unsets parameters if there is at least one argument; the only way to unset all parameters is to use *shift*. Using the KornShell version should not affect System V scripts because there should be no reason to issue it without arguments deliberately; if it were issued as, for example:

```
set -- "$@"
```

and there were in fact no arguments resulting from `"$@"`, unsetting the parameters would have no result.

The `set +` form in early proposals was omitted as being an unnecessary duplication of `set` alone and not widespread historical practice.

The *noclobber* option was changed to allow `set -C` as well as the `set -o noclobber` option. The single-letter version was added so that the historical `"$-"` paradigm would not be broken; see [Section 2.5.2](#).

The description of the `-e` option is intended to match the behavior of the 1988 version of the KornShell.

The `-h` flag is related to command name hashing. See [hash](#).

The following `set` flags were omitted intentionally with the following rationale:

**-k** The `-k` flag was originally added by the author of the Bourne shell to make it easier for users of pre-release versions of the shell. In early versions of the Bourne shell the construct `set name=value` had to be used to assign values to shell variables. The problem with `-k` is that the behavior affects parsing, virtually precluding writing any compilers. To explain the behavior of `-k`, it is necessary to describe the parsing algorithm, which is implementation-defined. For example:

```
set -k; echo name=value
```

and:



```
76934     set -k
76935     echo name=value
```

76936 behave differently. The interaction with functions is even more complex. What is more, the  
76937 `-k` flag is never needed, since the command line could have been reordered.

76938 `-t` The `-t` flag is hard to specify and almost never used. The only known use could be done  
76939 with here-documents. Moreover, the behavior with *ksh* and *sh* differs. The reference page  
76940 says that it exits after reading and executing one command. What is one command? If the  
76941 input is *date;date*, *sh* executes both *date* commands while *ksh* does only the first.

76942 Consideration was given to rewriting *set* to simplify its confusing syntax. A specific suggestion  
76943 was that the *unset* utility should be used to unset options instead of using the non-*getopt*()-able  
76944 `+option` syntax. However, the conclusion was reached that the historical practice of using `+option`  
76945 was satisfactory and that there was no compelling reason to modify such widespread historical  
76946 practice.

76947 The `-o` option was adopted from the KornShell to address user needs. In addition to its  
76948 generally friendly interface, `-o` is needed to provide the *vi* command line editing mode, for  
76949 which historical practice yields no single-letter option name. (Although it might have been  
76950 possible to invent such a letter, it was recognized that other editing modes would be developed  
76951 and `-o` provides ample name space for describing such extensions.)

76952 Historical implementations are inconsistent in the format used for `-o` option status reporting.  
76953 The `+o` format without an option-argument was added to allow portable access to the options  
76954 that can be saved and then later restored using, for instance, a dot script.

76955 Historically, *sh* did trace the command *set +x*, but *ksh* did not.

76956 The *ignoreeof* setting prevents accidental logouts when the end-of-file character (typically  
76957 `<control>-D`) is entered. A user shall explicitly *exit* to leave the interactive shell.

76958 The *set -m* option was added to apply only to the UPE because it applies primarily to interactive  
76959 use, not shell script applications.

76960 The ability to do asynchronous notification became available in the 1988 version of the  
76961 KornShell. To have it occur, the user had to issue the command:

```
76962     trap "jobs -n" CLD
```

76963 The C shell provides two different levels of an asynchronous notification capability. The  
76964 environment variable *notify* is analogous to what is done in *set -b* or *set -o notify*. When set, it  
76965 notifies the user immediately of background job completions. When unset, this capability is  
76966 turned off.

76967 The other notification ability comes through the built-in utility *notify*. The syntax is:

```
76968     notify [%job ... ]
```

76969 By issuing *notify* with no operands, it causes the C shell to notify the user asynchronously when  
76970 the state of the current job changes. If given operands, *notify* asynchronously informs the user of  
76971 changes in the states of the specified jobs.

76972 To add asynchronous notification to the POSIX shell, neither the KornShell extensions to *trap*,  
76973 nor the C shell *notify* environment variable seemed appropriate (*notify* is not a proper POSIX  
76974 environment variable name).

76975 The *set -b* option was selected as a compromise.

76976 The *notify* built-in was considered to have more functionality than was required for simple



76977 asynchronous notification.

76978 Historically, some shells applied the `-u` option to all parameters including `$@` and `$*`. The  
 76979 standard developers felt that this was a misfeature since it is normal and common for `$@` and `$*`  
 76980 to be used in shell scripts regardless of whether they were passed any arguments. Treating these  
 76981 uses as an error when no arguments are passed reduces the value of `-u` for its intended purpose  
 76982 of finding spelling mistakes in variable names and uses of unset positional parameters.

## 76983 FUTURE DIRECTIONS

76984 None.

## 76985 SEE ALSO

76986 [Section 2.14, \*hash\*](#)

76987 [XBD Section 4.23, Section 12.2](#)

## 76988 CHANGE HISTORY

### 76989 Issue 6

76990 The obsolescent *set* command name followed by `'-'` has been removed.

76991 The following new requirements on POSIX implementations derive from alignment with the  
 76992 Single UNIX Specification:

- 76993 • The *nolog* option is added to *set* `-o`.

76994 IEEE PASC Interpretation 1003.2 #167 is applied, clarifying that the options default also takes  
 76995 into account the description of the option.

76996 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page  
 76997 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in  
 76998 behavior is intended.

76999 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/8 is applied, changing the square  
 77000 brackets in the example in RATIONALE to be in bold, which is the typeface used for optional  
 77001 items.

### 77002 Issue 7

77003 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying the behavior if the first  
 77004 argument is `'-'`.

77005 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

77006 XSI shading is removed from the `-h` functionality.

77007 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0046 [52], XCU/TC1-2008/0047  
 77008 [155,280], XCU/TC1-2008/0048 [52], XCU/TC1-2008/0049 [52], and XCU/TC1-2008/0050  
 77009 [155,430] are applied.

77010 **NAME**

77011 shift — shift positional parameters

77012 **SYNOPSIS**77013 shift [*n*]77014 **DESCRIPTION**

77015 The positional parameters shall be shifted. Positional parameter 1 shall be assigned the value of  
77016 parameter (1+*n*), parameter 2 shall be assigned the value of parameter (2+*n*), and so on. The  
77017 parameters represented by the numbers "\$#" down to "\$#-*n*+1" shall be unset, and the  
77018 parameter '#' is updated to reflect the new number of positional parameters.

77019 The value *n* shall be an unsigned decimal integer less than or equal to the value of the special  
77020 parameter '#'. If *n* is not given, it shall be assumed to be 1. If *n* is 0, the positional and special  
77021 parameters are not changed.

77022 **OPTIONS**

77023 None.

77024 **OPERANDS**

77025 See the DESCRIPTION.

77026 **STDIN**

77027 Not used.

77028 **INPUT FILES**

77029 None.

77030 **ENVIRONMENT VARIABLES**

77031 None.

77032 **ASYNCHRONOUS EVENTS**

77033 Default.

77034 **STDOUT**

77035 Not used.

77036 **STDERR**

77037 The standard error shall be used only for diagnostic messages.

77038 **OUTPUT FILES**

77039 None.

77040 **EXTENDED DESCRIPTION**

77041 None.

77042 **EXIT STATUS**

77043 If the *n* operand is invalid or is greater than "\$#", this may be considered a syntax error and a  
77044 non-interactive shell may exit; if the shell does not exit in this case, a non-zero exit status shall be  
77045 returned. Otherwise, zero shall be returned.

77046 **CONSEQUENCES OF ERRORS**

77047 Default.

77048 **APPLICATION USAGE**

77049 None.

77050 **EXAMPLES**77051 `$ set a b c d e`77052 `$ shift 2`77053 `$ echo $*`77054 `c d e`77055 **RATIONALE**

77056 None.

77057 **FUTURE DIRECTIONS**

77058 None.

77059 **SEE ALSO**77060 [Section 2.14](#)77061 **CHANGE HISTORY**77062 **Issue 6**

77063 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page  
77064 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in  
77065 behavior is intended.

77066 **Issue 7**

77067 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0051 [459] is applied.

77068 **NAME**

77069 times — write process times

77070 **SYNOPSIS**

77071 times

77072 **DESCRIPTION**

77073 The *times* utility shall write the accumulated user and system times for the shell and for all of its  
 77074 child processes, in the following POSIX locale format:

```
77075 "%dm%fs %dm%fs\n%dm%fs %dm%fs\n", <shell user minutes>,  

77076 <shell user seconds>, <shell system minutes>,  

77077 <shell system seconds>, <children user minutes>,  

77078 <children user seconds>, <children system minutes>,  

77079 <children system seconds>
```

77080 The four pairs of times shall correspond to the members of the **<sys/times.h> tms** structure  
 77081 (defined in XBD [Chapter 13](#)) as returned by *times()*: *tms\_utime*, *tms\_stime*, *tms\_cutime*, and  
 77082 *tms\_cstime*, respectively.

77083 **OPTIONS**

77084 None.

77085 **OPERANDS**

77086 None.

77087 **STDIN**

77088 Not used.

77089 **INPUT FILES**

77090 None.

77091 **ENVIRONMENT VARIABLES**

77092 None.

77093 **ASYNCHRONOUS EVENTS**

77094 Default.

77095 **STDOUT**

77096 See the DESCRIPTION.

77097 **STDERR**

77098 The standard error shall be used only for diagnostic messages.

77099 **OUTPUT FILES**

77100 None.

77101 **EXTENDED DESCRIPTION**

77102 None.

77103 **EXIT STATUS**

77104 0 Successful completion. |

77105 &gt;0 An error occurred. |

77106 **CONSEQUENCES OF ERRORS**

77107 Default.

77108 **APPLICATION USAGE**

77109 None.

77110 **EXAMPLES**

77111 **\$ times**

77112 **0m0.43s 0m1.11s**

77113 **8m44.18s 1m43.23s**

77114 **RATIONALE**

77115 The *times* special built-in from the Single UNIX Specification is now required for all conforming  
77116 shells.

77117 **FUTURE DIRECTIONS**

77118 None.

77119 **SEE ALSO**

77120 [Section 2.14](#)

77121 XBD [<sys/times.h>](#)

77122 **CHANGE HISTORY**

77123 **Issue 6**

77124 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/9 is applied, changing text in the  
77125 DESCRIPTION from: "Write the accumulated user and system times for the shell and for all of  
77126 its child processes ..." to: "The *times* utility shall write the accumulated user and system times for  
77127 the shell and for all of its child processes ...".

77128 **NAME**

77129 trap — trap signals

77130 **SYNOPSIS**77131 trap *n* [*condition...*]77132 trap [*action condition...*]77133 **DESCRIPTION**

77134 If the first operand is an unsigned decimal integer, the shell shall treat all operands as  
 77135 conditions, and shall reset each condition to the default value. Otherwise, if there are operands,  
 77136 the first is treated as an action and the remaining as conditions.

77137 If *action* is `'-'`, the shell shall reset each *condition* to the default value. If *action* is null (`" "`), the  
 77138 shell shall ignore each specified *condition* if it arises. Otherwise, the argument *action* shall be read  
 77139 and executed by the shell when one of the corresponding conditions arises. The action of *trap*  
 77140 shall override a previous action (either default action or one explicitly set). The value of `"$?"`  
 77141 after the *trap* action completes shall be the value it had before *trap* was invoked.

77142 The condition can be EXIT, 0 (equivalent to EXIT), or a signal specified using a symbolic name,  
 77143 without the SIG prefix, as listed in the tables of signal names in the `<signal.h>` header defined in  
 77144 XBD [Chapter 13](#); for example, HUP, INT, QUIT, TERM. Implementations may permit names  
 77145 with the SIG prefix or ignore case in signal names as an extension. Setting a trap for SIGKILL or  
 77146 SIGSTOP produces undefined results.

77147 The environment in which the shell executes a *trap* on EXIT shall be identical to the environment  
 77148 immediately after the last command executed before the *trap* on EXIT was taken.

77149 Each time *trap* is invoked, the *action* argument shall be processed in a manner equivalent to:

77150 eval *action*

77151 Signals that were ignored on entry to a non-interactive shell cannot be trapped or reset, although  
 77152 no error need be reported when attempting to do so. An interactive shell may reset or catch  
 77153 signals ignored on entry. Traps shall remain in place for a given shell until explicitly changed  
 77154 with another *trap* command.

77155 When a subshell is entered, traps that are not being ignored shall be set to the default actions,  
 77156 except in the case of a command substitution containing only a single *trap* command, when the  
 77157 traps need not be altered. Implementations may check for this case using only lexical analysis;  
 77158 for example, if `'trap'` and `$( trap -- )` do not alter the traps in the subshell, cases such as  
 77159 assigning `var=trap` and then using `$( $var )` may still alter them. This does not imply that the  
 77160 *trap* command cannot be used within the subshell to set new traps.

77161 The *trap* command with no operands shall write to standard output a list of commands  
 77162 associated with each condition. If the command is executed in a subshell, the implementation  
 77163 does not perform the optional check described above for a command substitution containing  
 77164 only a single *trap* command, and no *trap* commands with operands have been executed since  
 77165 entry to the subshell, the list shall contain the commands that were associated with each  
 77166 condition immediately before the subshell environment was entered. Otherwise, the list shall  
 77167 contain the commands currently associated with each condition. The format shall be:

77168 "trap -- %s %s ...\n", *<action>*, *<condition>* ...

77169 The shell shall format the output, including the proper use of quoting, so that it is suitable for  
 77170 reinput to the shell as commands that achieve the same trapping results. For example:

77171 save\_traps=\$(trap)

77172 ...

77173 eval "\$save\_traps"

77174 XSI XSI-conformant systems also allow numeric signal numbers for the conditions corresponding to  
 77175 the following signal names:

|       |    |         |
|-------|----|---------|
| 77176 | 1  | SIGHUP  |
| 77177 | 2  | SIGINT  |
| 77178 | 3  | SIGQUIT |
| 77179 | 6  | SIGABRT |
| 77180 | 9  | SIGKILL |
| 77181 | 14 | SIGALRM |
| 77182 | 15 | SIGTERM |

77183 The *trap* special built-in shall conform to XBD [Section 12.2](#).

#### 77184 OPTIONS

77185 None.

#### 77186 OPERANDS

77187 See the DESCRIPTION.

#### 77188 STDIN

77189 Not used.

#### 77190 INPUT FILES

77191 None.

#### 77192 ENVIRONMENT VARIABLES

77193 None.

#### 77194 ASYNCHRONOUS EVENTS

77195 Default.

#### 77196 STDOUT

77197 See the DESCRIPTION.

#### 77198 STDERR

77199 The standard error shall be used only for diagnostic messages.

#### 77200 OUTPUT FILES

77201 None.

#### 77202 EXTENDED DESCRIPTION

77203 None.

#### 77204 EXIT STATUS

77205 XSI If the trap name **or number** is invalid, a non-zero exit status shall be returned; otherwise, zero  
 77206 XSI shall be returned. For both interactive and non-interactive shells, invalid signal names **or**  
 77207 **numbers** shall not be considered a syntax error and do not cause the shell to abort.

#### 77208 CONSEQUENCES OF ERRORS

77209 Default.

**APPLICATION USAGE**

None.

**EXAMPLES**

Write out a list of all traps and actions:

```
trap
```

Set a trap so the *logout* utility in the directory referred to by the *HOME* environment variable executes when the shell terminates:

```
trap "$HOME"/logout EXIT
```

or:

```
trap "$HOME"/logout 0
```

Unset traps on INT, QUIT, TERM, and EXIT:

```
trap - INT QUIT TERM EXIT
```

**RATIONALE**

Implementations may permit lowercase signal names as an extension. Implementations may also accept the names with the SIG prefix; no known historical shell does so. The *trap* and *kill* utilities in this volume of POSIX.1-2008 are now consistent in their omission of the SIG prefix for signal names. Some *kill* implementations do not allow the prefix, and *kill -l* lists the signals without prefixes.

Trapping SIGKILL or SIGSTOP is syntactically accepted by some historical implementations, but it has no effect. Portable POSIX applications cannot attempt to trap these signals.

The output format is not historical practice. Since the output of historical *trap* commands is not portable (because numeric signal values are not portable) and had to change to become so, an opportunity was taken to format the output in a way that a shell script could use to save and then later reuse a trap if it wanted.

The KornShell uses an **ERR** trap that is triggered whenever *set -e* would cause an exit. This is allowable as an extension, but was not mandated, as other shells have not used it.

The text about the environment for the EXIT trap invalidates the behavior of some historical versions of interactive shells which, for example, close the standard input before executing a trap on 0. For example, in some historical interactive shell sessions the following trap on 0 would always print "--":

```
trap 'read foo; echo "--$foo--"' 0
```

The command:

```
trap 'eval "$cmd"' 0
```

causes the contents of the shell variable *cmd* to be executed as a command when the shell exits. Using:

```
trap '$cmd' 0
```

does not work correctly if *cmd* contains any special characters such as quoting or redirections. Using:

```
trap "$cmd" 0
```

also works (the leading <space> character protects against unlikely cases where *cmd* is a decimal integer or begins with '-'), but it expands the *cmd* variable when the *trap* command is executed,



77251 not when the exit action is executed.

## 77252 FUTURE DIRECTIONS

77253 None.

## 77254 SEE ALSO

77255 [Section 2.14](#)

77256 XBD [Section 12.2, <signal.h>](#)

## 77257 CHANGE HISTORY

### 77258 Issue 6

77259 XSI-conforming implementations provide the mapping of signal names to numbers given above  
77260 (previously this had been marked obsolescent). Other implementations need not provide this  
77261 optional mapping.

77262 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page  
77263 sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in  
77264 behavior is intended.

### 77265 Issue 7

77266 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

77267 Austin Group Interpretation 1003.1-2001 #116 is applied.

77268 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0052 [53,268,440],  
77269 XCU/TC1-2008/0053 [53,268,440], XCU/TC1-2008/0054 [163], XCU/TC1-2008/0055 [163], and  
77270 XCU/TC1-2008/0056 [163] are applied.

77271 **NAME**

77272       unset — unset values and attributes of variables and functions

77273 **SYNOPSIS**77274       unset [-fv] *name*...77275 **DESCRIPTION**77276       Each variable or function specified by *name* shall be unset.77277       If **-v** is specified, *name* refers to a variable name and the shell shall unset it and remove it from  
77278       the environment. Read-only variables cannot be unset.77279       If **-f** is specified, *name* refers to a function and the shell shall unset the function definition.77280       If neither **-f** nor **-v** is specified, *name* refers to a variable; if a variable by that name does not  
77281       exist, it is unspecified whether a function by that name, if any, shall be unset.77282       Unsetting a variable or function that was not previously set shall not be considered an error and  
77283       does not cause the shell to abort.77284       The *unset* special built-in shall support XBD [Section 12.2](#).

77285       Note that:

77286       VARIABLE=

77287       is not equivalent to an *unset* of **VARIABLE**; in the example, **VARIABLE** is set to " ". Also, the  
77288       variables that can be *unset* should not be misinterpreted to include the special parameters (see  
77289       [Section 2.5.2](#)).77290 **OPTIONS**

77291       See the DESCRIPTION.

77292 **OPERANDS**

77293       See the DESCRIPTION.

77294 **STDIN**

77295       Not used.

77296 **INPUT FILES**

77297       None.

77298 **ENVIRONMENT VARIABLES**

77299       None.

77300 **ASYNCHRONOUS EVENTS**

77301       Default.

77302 **STDOUT**

77303       Not used.

77304 **STDERR**

77305       The standard error shall be used only for diagnostic messages.

77306 **OUTPUT FILES**

77307       None.

77308 **EXTENDED DESCRIPTION**

77309       None.

77310 **EXIT STATUS**

77311           0   All *name* operands were successfully unset.

77312           >0   At least one *name* could not be unset.

77313 **CONSEQUENCES OF ERRORS**

77314           Default.

77315 **APPLICATION USAGE**

77316           None.

77317 **EXAMPLES**

77318           Unset *VISUAL* variable:

77319           unset -v VISUAL

77320           Unset the functions **foo** and **bar**:

77321           unset -f foo bar

77322 **RATIONALE**

77323           Consideration was given to omitting the **-f** option in favor of an *unfunction* utility, but the  
77324           standard developers decided to retain historical practice.

77325           The **-v** option was introduced because System V historically used one name space for both  
77326           variables and functions. When *unset* is used without options, System V historically unset either a  
77327           function or a variable, and there was no confusion about which one was intended. A portable  
77328           POSIX application can use *unset* without an option to unset a variable, but not a function; the **-f**  
77329           option must be used.

77330 **FUTURE DIRECTIONS**

77331           None.

77332 **SEE ALSO**

77333           [Section 2.14](#)

77334           XBD [Section 12.2](#)

77335 **CHANGE HISTORY**

77336 **Issue 6**

77337           IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/5 is applied so that the reference page  
77338           sections use terms as described in the Utility Description Defaults ([Section 1.4](#)). No change in  
77339           behavior is intended.

77340 **Issue 7**

77341           SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



# Batch Environment Services



This chapter describes the services and utilities that shall be implemented on all systems that claim conformance to the Batch Environment Services and Utilities option. The functionality described in this section shall be provided on implementations that support the Batch Environment Services and Utilities option (and the rest of this section is not further shaded for this option).

Note that the Batch Environment Services and Utilities option is marked obsolescent in Issue 7.

## 3.1 General Concepts

### 3.1.1 Batch Client-Server Interaction

Batch jobs are created and managed by batch servers. A batch client interacts with a batch server to access batch services on behalf of the user. In order to use batch services, a user must have access to a batch client.

A batch server is a computational entity, such as a daemon process, that provides batch services. Batch servers route, queue, modify, and execute batch jobs on behalf of batch clients.

The batch utilities described in this volume of POSIX.1-2008 (and listed in Table 3-1) are clients of batch services; they allow users to perform actions on the job such as creating, modifying, and deleting batch jobs from a shell command line. Although these batch utilities may be said to accomplish certain services, they actually obtain services on behalf of a user by means of requests to batch servers.

Table 3-1 Batch Utilities

|               |               |                |              |
|---------------|---------------|----------------|--------------|
| <i>qalter</i> | <i>qmove</i>  | <i>qrls</i>    | <i>qstat</i> |
| <i>qdel</i>   | <i>qmsg</i>   | <i>qselect</i> | <i>qsub</i>  |
| <i>qhold</i>  | <i>qrerun</i> | <i>qsig</i>    |              |

Client-server interaction takes place by means of the batch requests defined in this chapter. Because direct access to batch jobs and queues is limited to batch servers, clients and servers of different implementations can interoperate, since dependencies on private structures for batch jobs and queues are limited to batch servers. Also, batch servers may be clients of other batch servers.

### 3.1.2 Batch Queues

Two types of batch queue are described: routing queues and execution queues. When a batch job is placed in a routing queue, it is a candidate for routing. A batch job is removed from routing queues under the following conditions:

- The batch job has been routed to another queue.
- The batch job has been deleted from the batch queue.
- The batch job has been aborted.

When a batch job is placed in an execution queue, it is a candidate for execution.

A batch job is removed from an execution queue under the following conditions:

- The batch job has been executed and exited.
- The batch job has been aborted.
- The batch job has been deleted from the batch queue.
- The batch job has been moved to another queue.

Access to a batch queue is limited to the batch server that manages the batch queue. Clients never access a batch queue or a batch job directly, either to read or write information; all client access to batch queues or jobs takes place through batch servers.

### 3.1.3 Batch Job Creation

When a batch server creates a batch job on behalf of a client, it shall assign a batch job identifier to the job. A batch job identifier consists of both a sequence number that is unique among the sequence numbers issued by that server and the name of the server. Since the batch server name is unique within a name space, the job identifier is likewise unique within the name space.

The batch server that creates a batch job shall return the batch server-assigned job identifier to the client that requested the job creation. If the batch server routes or moves the job to another server, it sends the job identifier with the job. Once assigned, the job identifier of a batch job shall never change.

### 3.1.4 Batch Job Tracking

Since a batch job may be moved after creation, the batch server name component of the job identifier need not indicate the location of the job. An implementation may provide a batch job tracking mechanism, in which case the user generally does not need to know the location of the job. However, an implementation need not provide a batch job tracking mechanism, in which case the user must find routed jobs by probing the possible destinations.

**3.1.5 Batch Job Routing**

To route a batch job, a batch server either moves the job to some other queue that is managed by the batch server, or requests that some other batch server accept the job.

Each routing queue has one or more queues to which it can route batch jobs. The batch server administrator creates routing queues.

A batch server may route a batch job from a routing queue to another routing queue. Batch servers shall prevent or otherwise handle cases of circular routing paths. As a deferred service, a batch server routes jobs from the routing queues that it manages. The algorithm by which a batch server selects a batch queue to which to route a batch job is implementation-defined.

A batch job need not be eligible for routing to all the batch queues fed by the routing queue from which it is routed. A batch server that has been asked to accept the job may reject the request if the job requires resources that are unavailable to that batch server, or if the client is not authorized to access the batch server.

Batch servers may route high-priority jobs before low-priority jobs, but, on other than overloaded systems, the effect may be imperceptible to the user. If all the batch servers fed by a routing queue reject requests to accept the job for reasons that are permanent, the batch server that manages the job shall abort the job. If all or some rejections are temporary, the batch server should try to route the job again at some later point.

The reasons for rejecting a batch job are implementation-defined.

The reasons for which the routing should be retried later and the reasons for which the job should be aborted are also implementation-defined.

**3.1.6 Batch Job Execution**

To execute a batch job is to create a session leader (a process) that runs the shell program indicated by the *Shell\_Path* attribute of the job. The script shall be passed to the program as its standard input. An implementation may pass the script to the program by other implementation-defined means. At the time a batch job begins execution, it is defined to enter the RUNNING state. The primary program that is executed by a batch job is typically, though not necessarily, a shell program.

A batch server shall execute eligible jobs as a deferred service—no client request is necessary once the batch job is created and eligible. However, the attributes of a batch job, such as the job hold type, may render the job ineligible. A batch server shall scan the execution queues that it manages for jobs that are eligible for execution. The algorithm by which the batch server selects eligible jobs for execution is implementation-defined.

As part of creating the process for the batch job, the batch server shall open the standard output and standard error streams of the session.

The attributes of a batch job may indicate that the batch server executing the job shall send mail to a list of users at the time it begins execution of the job.

### 77439 3.1.7 Batch Job Exit

77440 When the session leader of an executing job terminates, the job exits. As part of exiting a batch  
77441 job, the batch server that manages the job shall remove the job from the batch queue in which it  
77442 resides. The server shall transfer output files of the job to a location described by the attributes of  
77443 the job.

77444 The attributes of a batch job may indicate that the batch server managing the job shall send mail  
77445 to a list of users at the time the job exits.

### 77446 3.1.8 Batch Job Abort

77447 A batch server shall abort jobs for which a required deferred service cannot be performed. The  
77448 attributes of a batch job may indicate that the batch server that aborts the job shall send mail to a  
77449 list of users at the time it aborts the job.

### 77450 3.1.9 Batch Authorization

77451 Clients, such as the batch environment utilities (marked BE), access batch services by means of  
77452 requests to one or more batch servers. To acquire the services of any given batch server, the user  
77453 identifier under which the client runs must be authorized to use that batch server.

77454 The user with an associated user name that creates a batch job shall own the job and can perform  
77455 actions such as read, modify, delete, and move.

77456 A user identifier of the same value at a different host need not be the same user. For example,  
77457 user name *smith* at host **alpha** may or may not represent the same person as user name *smith* at  
77458 host **beta**. Likewise, the same person may have access to different user names on different hosts.

77459 An implementation may optionally provide an authorization mechanism that permits one user  
77460 name to access jobs under another user name.

77461 A process on a client host may be authorized to run processes under multiple user names at a  
77462 batch server host. Where appropriate, the utilities defined in this volume of POSIX.1-2008  
77463 provide a means for a user to choose from among such user names when creating or modifying  
77464 a batch job.

### 77465 3.1.10 Batch Administration

77466 The processing of a batch job by a batch server is affected by the attributes of the job. The  
77467 processing of a batch job may also be affected by the attributes of the batch queue in which the  
77468 job resides and by the status of the batch server that manages the job. See also XBD [Chapter 3](#)  
77469 (on page 33) for batch definitions.



**3.1.11 Batch Notification**

Whereas batch servers are persistent entities, clients are often transient. For example, the *qsub* utility creates a batch job and exits. For this reason, batch servers notify users of batch job events by sending mail to the user that owns the job, or to other designated users.

**3.2 Batch Services**

The presence of Batch Environment Services and Utilities option services is indicated by the configuration variable `POSIX2_PBS`. A conforming batch server provides services as defined in this section.

A batch server shall provide batch services in two ways:

1. The batch server provides a service at the request of a client.
2. The batch server provides a deferred service as a result of a change in conditions monitored by the batch server.

If a batch server cannot complete a request, it shall reject the request. If a batch server cannot complete a deferred service for a batch job, the batch server shall abort the batch job. [Table 3-2](#) is a summary of environment variables that shall be supported by an implementation of the batch server and utilities.

**Table 3-2** Environment Variable Summary

| Variable                     | Description                                                                       |
|------------------------------|-----------------------------------------------------------------------------------|
| <code>PBS_DPREFIX</code>     | Defines the directive prefix (see <a href="#">qsub</a> )                          |
| <code>PBS_ENVIRONMENT</code> | Batch Job is batch or interactive (see <a href="#">Section 3.2.2.1</a> )          |
| <code>PBS_JOBID</code>       | The <i>job_identifier</i> attribute of job (see <a href="#">Section 3.2.3.8</a> ) |
| <code>PBS_JOBNAME</code>     | The <i>job_name</i> attribute of job (see <a href="#">Section 3.2.3.8</a> )       |
| <code>PBS_O_HOME</code>      | Defines the <i>HOME</i> of the batch client (see <a href="#">qsub</a> )           |
| <code>PBS_O_HOST</code>      | Defines the host name of the batch client (see <a href="#">qsub</a> )             |
| <code>PBS_O_LANG</code>      | Defines the <i>LANG</i> of the batch client (see <a href="#">qsub</a> )           |
| <code>PBS_O_LOGNAME</code>   | Defines the <i>LOGNAME</i> of the batch client (see <a href="#">qsub</a> )        |
| <code>PBS_O_MAIL</code>      | Defines the <i>MAIL</i> of the batch client (see <a href="#">qsub</a> )           |
| <code>PBS_O_PATH</code>      | Defines the <i>PATH</i> of the batch client (see <a href="#">qsub</a> )           |
| <code>PBS_O_QUEUE</code>     | Defines the submit queue of the batch client (see <a href="#">qsub</a> )          |
| <code>PBS_O_SHELL</code>     | Defines the <i>SHELL</i> of the batch client (see <a href="#">qsub</a> )          |
| <code>PBS_O_TZ</code>        | Defines the <i>TZ</i> of the batch client (see <a href="#">qsub</a> )             |
| <code>PBS_O_WORKDIR</code>   | Defines the working directory of the batch client (see <a href="#">qsub</a> )     |
| <code>PBS_QUEUE</code>       | Defines the initial execution queue (see <a href="#">Section 3.2.2.1</a> )        |

### 3.2.1 Batch Job States

A batch job shall always be in one of the following states: QUEUED, RUNNING, HELD, WAITING, EXITING, or TRANSITING. The state of a batch job determines the types of requests that the batch server that manages the batch job can accept for the batch job. A batch server shall change the state of a batch job either in response to service requests from clients or as a result of deferred services, such as job execution or job routing.

A batch job that is in the QUEUED state resides in a queue but is still pending either execution or routing, depending on the queue type.

A batch server that queues a batch job in a routing queue shall put the batch job in the QUEUED state. A batch server that puts a batch job in an execution queue, but has not yet executed the batch job, shall put the batch job in the QUEUED state. A batch job that resides in an execution queue and is executing is defined to be in the RUNNING state. While a batch job is in the RUNNING state, a session leader is associated with the batch job.

A batch job that resides in an execution queue, but is ineligible to run because of a hold attribute, is defined to be in the HELD state.

A batch job that is not held, but must wait until a future date and time before executing, is defined to be in the WAITING state.

When the session leader associated with a running job exits, the batch job shall be placed in the EXITING state.

A batch job for which the session leader has terminated is defined to be in the EXITING state, and the batch server that manages such a batch job cannot accept job modification requests that affect the batch job. While a batch job is in the EXITING state, the batch server that manages the batch job is staging output files and notifying clients of job completion. Once a batch job has exited, it no longer exists as an object managed by a batch server.

A batch job that is being moved from a routing queue to another queue is defined to be in the TRANSITING state.

When a batch job in a routing queue has been selected to be moved to a new destination, then the batch job shall be in either the QUEUED state or the TRANSITING state, depending on the batch server implementation.

Batch jobs with either an *Execution\_Time* attribute value set in the future or a *Hold\_Types* attribute of value not equal to NO\_HOLD, or both, may be routed or held in the routing queue. The treatment of jobs with the *Execution\_Time* or *Hold\_Types* attributes in a routing queue is implementation-defined.

When a batch job in a routing queue has not been selected to be moved to a new destination and the batch job has a *Hold\_Types* attribute value of other than NO\_HOLD, then the job should be in the HELD state.

**Note:** The effect of a hold upon a batch job in a routing queue is implementation-defined. The implementation should use the state that matches whether the batch job can route with a hold or not.

When a batch job in a routing queue has not been selected to be moved to a new destination and the batch job has:

- A *Hold\_Types* attribute value of NO\_HOLD
- An *Execution\_Time* attribute in the past

then the batch job shall be in the QUEUED state.

When a batch job in a routing queue has not been selected to be moved to a new destination and the batch job has:

- A *Hold\_Types* attribute value of NO\_HOLD
- An *Execution\_Time* attribute in the future

then the batch job may be in the WAITING state.

**Note:** The effect of a future execution time upon a batch job in a routing queue is implementation-defined. The implementation should use the state that matches whether the batch job can route with a hold or not.

Table 3-3 describes the next state of a batch job, given the current state of the batch job and the type of request. Table 3-4 (on page 2425) describes the response of a batch server to a request, given the current state of the batch job and the type of request.

### 3.2.2 Deferred Batch Services

This section describes the deferred services performed by batch servers: job execution, job routing, job exit, job abort, and the rerunning of jobs after a restart.

#### 3.2.2.1 Batch Job Execution

To execute a batch job is to create a session leader (a process) that runs the shell program indicated by the *Shell\_Path\_List* attribute of the batch job. The script is passed to the program as its standard input. An implementation may pass the script to the program by other implementation-defined means. At the time a batch job begins execution, it is defined to enter the RUNNING state.

Table 3-3 Next State Table

| Request Type               | Current State |   |     |       |   |   |   |
|----------------------------|---------------|---|-----|-------|---|---|---|
|                            | X             | Q | R   | H     | W | E | T |
| Queue Batch Job Request    | Q             | e | e   | e     | e | e | e |
| Modify Batch Job Request   | e             | Q | R   | H     | W | e | T |
| Delete Batch Job Request   | e             | X | E   | X     | X | E | X |
| Batch Job Message Request  | e             | Q | R   | H     | W | E | T |
| Rerun Batch Job Request    | e             | e | Q   | e     | e | e | e |
| Signal Batch Job Request   | e             | e | R   | H     | W | e | e |
| Batch Job Status Request   | e             | Q | R   | H     | W | E | T |
| Batch Queue Status Request | X             | Q | R   | H     | W | E | T |
| Server Status Request      | X             | Q | R   | H     | W | E | T |
| Select Batch Jobs Request  | X             | Q | R   | H     | W | E | T |
| Move Batch Job Request     | e             | Q | R   | H     | W | e | T |
| Hold Batch Job Request     | e             | H | R/H | H     | H | e | T |
| Release Batch Job Request  | e             | Q | R   | Q/W/H | W | e | T |
| Server Shutdown Request    | X             | Q | Q   | H     | W | E | T |
| Locate Batch Job Request   | e             | Q | R   | H     | W | E | T |

77585      **Legend**

77586      X    Nonexistent

77587      Q    QUEUED

77588      R    RUNNING

77589      H    HELD

77590      W    WAITING

77591      E    EXITING

77592      T    TRANSITING

77593      e    Error

77594      A batch server that has an execution queue containing jobs is said to own the queue and manage  
 77595      the batch jobs in that queue. A batch server that has been started shall execute the batch jobs in  
 77596      the execution queues owned by the batch server. The batch server shall schedule for execution  
 77597      those jobs in the execution queues that are in the QUEUED state. The algorithm for scheduling  
 77598      jobs is implementation-defined.

77599      A batch server that executes a batch job shall create, in the environment of the session leader of  
 77600      the batch job, an environment variable named *PBS\_ENVIRONMENT*, the value of which is the  
 77601      string PBS\_BATCH encoded in the portable character set.

77602      A batch server that executes a batch job shall create, in the environment of the session leader of  
 77603      the batch job, an environment variable named *PBS\_QUEUE*, the value of which is the name of  
 77604      the execution queue of the batch job encoded in the portable character set.

77605      To rerun a batch job is to requeue a batch job that is currently executing and then kill the session  
 77606      leader of the executing job by sending a SIGKILL prior to completion; see [Section 3.2.3.11](#) (on  
 77607      page 2437). A batch server that reruns a batch job shall append the standard output and  
 77608      standard error files of the batch job to the corresponding files of the previous execution, if they  
 77609      exist, with appropriate annotation. If either file does not exist, that file shall be created as in  
 77610      normal execution.

Table 3-4 Results/Output Table

| Request Type               | Current State |   |   |   |   |   |   |
|----------------------------|---------------|---|---|---|---|---|---|
|                            | X             | Q | R | H | W | E | T |
| Queue Batch Job Request    | O             | e | e | e | e | e | e |
| Modify Batch Job Request   | e             | O | e | O | O | e | e |
| Delete Batch Job Request   | e             | O | O | O | O | e | O |
| Batch Job Message Request  | e             | e | O | e | e | e | e |
| Rerun Batch Job Request    | e             | e | O | e | e | e | e |
| Signal Batch Job Request   | e             | e | O | e | e | e | e |
| Batch Job Status Request   | e             | O | O | O | O | O | O |
| Batch Queue Status Request | O             | O | O | O | O | O | O |
| Server Status Request      | O             | O | O | O | O | O | O |
| Select Batch Job Request   | e             | O | O | O | O | O | O |
| Move Batch Job Request     | e             | O | O | O | O | e | e |
| Hold Batch Job Request     | e             | O | O | O | O | e | e |
| Release Batch Job Request  | e             | O | e | O | O | e | e |
| Server Shutdown Request    | O             | O | e | O | O | e | e |
| Locate Batch Job Request   | e             | O | O | O | O | O | O |

**Legend**

O OK

e Error message

The execution of a batch job by a batch server shall be controlled by job, queue, and server attributes, as defined in this section.

**Account\_Name Attribute**

Batch accounting is an optional feature of batch servers. If a batch server implements accounting, the statements in this section apply and the configuration variable POSIX2\_PBS\_ACCOUNTING shall be set to 1.

A batch server that executes a batch job shall charge the account named in the *Account\_Name* attribute of the batch job for resources consumed by the batch job.

If the *Account\_Name* attribute of the batch job is absent from the batch job attribute list or is altered while the batch job is in execution, the batch server action is implementation-defined.

**Checkpoint Attribute**

Batch checkpointing is an optional feature of batch servers. If a batch server implements checkpointing, the statements in this section apply and the configuration variable POSIX2\_PBS\_CHECKPOINT shall be set to 1.

There are two attributes associated with the checkpointing feature: *Checkpoint* and *Minimum\_Cpu\_Interval*. *Checkpoint* is a batch job attribute, while *Minimum\_Cpu\_Interval* is a queue attribute. An implementation that does not support checkpointing shall support the *Checkpoint* job attribute to the extent that the batch server shall maintain and pass this attribute to other servers.

The behavior of a batch server that executes a batch job for which the value of the *Checkpoint* attribute is CHECKPOINT\_UNSPECIFIED is implementation-defined. A batch server that executes a batch job for which the value of the *Checkpoint* attribute is NO\_CHECKPOINT shall

77654 not checkpoint the batch job.

77655 A batch server that executes a batch job for which the value of the *Checkpoint* attribute is  
77656 CHECKPOINT\_AT\_SHUTDOWN shall checkpoint the batch job only when the batch server  
77657 accepts a request to shut down during the time when the batch job is in the RUNNING state.

77658 A batch server that executes a batch job for which the value of the *Checkpoint* attribute is  
77659 CHECKPOINT\_AT\_MIN\_CPU\_INTERVAL shall checkpoint the batch job at the interval  
77660 specified by the *Minimum\_Cpu\_Interval* attribute of the queue for which the batch job has been  
77661 selected. The *Minimum\_Cpu\_Interval* attribute shall be specified in units of CPU minutes.

77662 A batch server that executes a batch job for which the value of the *Checkpoint* attribute is an  
77663 unsigned integer shall checkpoint the batch job at an interval that is the value of either the  
77664 *Checkpoint* attribute, or the *Minimum\_Cpu\_Interval* attribute of the queue for which the batch job  
77665 has been selected, whichever is greater. Both intervals shall be in units of CPU minutes. When  
77666 the *Minimum\_Cpu\_Interval* attribute is greater than the *Checkpoint* attribute, the batch job shall  
77667 write a warning message to the standard error stream of the batch job.

#### 77668 **Error\_Path Attribute**

77669 The *Error\_Path* attribute of a running job cannot be changed by a *Modify Batch Job Request*. When  
77670 the *Join\_Path* attribute of the batch job is set to the value FALSE and the *Keep\_Files* attribute of  
77671 the batch job does not contain the value KEEP\_STD\_ERROR, a batch server that executes a batch  
77672 job shall perform one of the following actions:

- 77673 • Set the standard error stream of the session leader of the batch job to the path described by  
77674 the value of the *Error\_Path* attribute of the batch job.
- 77675 • Buffer the standard error of the session leader of the batch job until completion of the batch  
77676 job, and when the batch job exits return the contents to the destination described by the  
77677 value of the *Error\_Path* attribute of the batch job.

77678 Applications shall not rely on having access to the standard error of a batch job prior to the  
77679 completion of the batch job.

77680 When the *Error\_Path* attribute does not specify a host name, then the batch server shall retain the  
77681 standard error of the batch job on the host of execution.

77682 When the *Error\_Path* attribute does specify a host name and the *Keep\_Files* attribute does not  
77683 contain the value KEEP\_STD\_ERROR, then the final destination of the standard error of the  
77684 batch job shall be on the host whose host name is specified.

77685 If the path indicated by the value of the *Error\_Path* attribute of the batch job is a relative path, the  
77686 batch server shall expand the path relative to the home directory of the user on the host to which  
77687 the file is being returned.

77688 When the batch server buffers the standard error of the batch job and the file cannot be opened  
77689 for write upon completion of the batch job, then the server shall place the standard error in an  
77690 implementation-defined location and notify the user of the location via mail. It shall be possible  
77691 for the user to process this mail using the *mailx* utility.

77692 If a batch server that does not buffer the standard error cannot open the standard error path of  
77693 the batch job for write access, then the batch server shall abort the batch job.



**Execution\_Time Attribute**

A batch server shall not execute a batch job before the time represented by the value of the *Execution\_Time* attribute of the batch job. The *Execution\_Time* attribute is defined in seconds since the Epoch.

**Hold\_Types Attribute**

A batch server shall support the following hold types:

- s Can be set or released by a user with at least a privilege level of batch administrator (SYSTEM).
- o Can be set or released by a user with at least a privilege level of batch operator (OPERATOR).
- u Can be set or released by the user with at least a privilege level of user, where the user is defined in the *Job\_Owner* attribute (USER).
- n Indicates that none of the *Hold\_Types* attributes are set (NO\_HOLD).

An implementation may define other hold types. Any additional hold types, how they are specified, their internal representation, their behavior, and how they affect the behavior of other utilities are implementation-defined.

The value of the *Hold\_Types* attribute shall be the union of the valid hold types ('s', 'o', 'u', and any implementation-defined hold types), or 'n'.

A batch server shall not execute a batch job if the *Hold\_Types* attribute of the batch job has a value other than NO\_HOLD. If the *Hold\_Types* attribute of the batch job has a value other than NO\_HOLD, the batch job shall be in the HELD state.

**Job\_Owner Attribute**

The *Job\_Owner* attribute consists of a pair of user name and host name values of the form:

username@hostname

A batch server that accepts a *Queue Batch Job Request* shall set the *Job\_Owner* attribute to a string that is the *username@hostname* of the user who submitted the job.

**Join\_Path Attribute**

A batch server that executes a batch job for which the value of the *Join\_Path* attribute is TRUE shall ignore the value of the *Error\_Path* attribute and merge the standard error of the batch job with the standard output of the batch job.

**Keep\_Files Attribute**

A batch server that executes a batch job for which the value of the *Keep\_Files* attribute includes the value KEEP\_STD\_OUTPUT shall retain the standard output of the batch job on the host where execution occurs. The standard output shall be retained in the home directory of the user under whose user ID the batch job is executed and the filename shall be the default filename for the standard output as defined under the *-o* option of the *qsub* utility. The *Output\_Path* attribute is not modified.

A batch server that executes a batch job for which the value of the *Keep\_Files* attribute includes the value KEEP\_STD\_ERROR shall retain the standard error of the batch job on the host where execution occurs. The standard error shall be retained in the home directory of the user under whose user ID the batch job is executed and the filename shall be the default filename for

standard error as defined under the `-e` option of the *qsub* utility. The *Error\_Path* attribute is not modified.

A batch server that executes a batch job for which the value of the *Keep\_Files* attribute includes values other than `KEEP_STD_OUTPUT` and `KEEP_STD_ERROR` shall retain these other files on the host where execution occurs. These files (with implementation-defined names) shall be retained in the home directory of the user under whose user identifier the batch job is executed.

#### Mail\_Points and Mail\_Users Attributes

A batch server that executes a batch job for which one of the values of the *Mail\_Points* attribute is the value `MAIL_AT_BEGINNING` shall send a mail message to each user account listed in the *Mail\_Users* attribute of the batch job.

The mail message shall contain at least the batch job identifier, queue, and server at which the batch job currently resides, and the *Job\_Owner* attribute.

#### Output\_Path Attribute

The *Output\_Path* attribute of a running job cannot be changed by a *Modify Batch Job Request*. When the *Keep\_Files* attribute of the batch job does not contain the value `KEEP_STD_OUTPUT`, a batch server that executes a batch job shall either:

- Set the standard output stream of the session leader of the batch job to the destination described by the value of the *Output\_Path* attribute of the batch job.

or:

- Buffer the standard output of the session leader of the batch job until completion of the batch job, and when the batch job exits return the contents to the destination described by the value of the *Output\_Path* attribute of the batch job.

When the *Output\_Path* attribute does not specify a host name, then the batch server shall retain the standard output of the batch job on the host of execution.

When the *Keep\_Files* attribute does not contain the value `KEEP_STD_OUTPUT` and the *Output\_Path* attribute does specify a host name, then the final destination of the standard output of the batch job shall be on the host specified.

If the path specified in the *Output\_Path* attribute of the batch job is a relative path, the batch server shall expand the path relative to the home directory of the user on the host to which the file is being returned.

Whether or not the batch server buffers the standard output of the batch job until completion of the batch job is implementation-defined. Applications shall not rely on having access to the standard output of a batch job prior to the completion of the batch job.

When the batch server does buffer the standard output of the batch job and the file cannot be opened for write upon completion of the batch job, then the batch server shall place the standard output in an implementation-defined location and notify the user of the location via mail. It shall be possible for the user to process this mail using the *mailx* utility.

If a batch server that does not buffer the standard output cannot open the standard output path of the batch job for write access, then the batch server shall abort the batch job.



**Priority Attribute**

A batch server implementation may choose to preferentially execute a batch job based on the *Priority* attribute. The interpretation of the batch job *Priority* attribute by a batch server is implementation-defined. If an implementation uses the *Priority* attribute, it shall interpret larger values of the *Priority* attribute to mean the batch job shall be preferentially selected for execution.

**Rerunable Attribute**

A batch job that began execution but did not complete, because the batch server either shut down or terminated abnormally, shall be requeued if the *Rerunable* attribute of the batch job has the value TRUE.

If a batch job, which was requeued after beginning execution but prior to completion, has a valid checkpoint file and the batch server supports checkpointing, then the batch job shall be restarted from the last valid checkpoint.

If the batch job cannot be restarted from a checkpoint, then when a batch job has a *Rerunable* attribute value of TRUE and was requeued after beginning execution but prior to completion, the batch server shall place the batch job into execution at the beginning of the job.

When a batch job has a *Rerunable* attribute value other than TRUE and was requeued after beginning execution but prior to completion, and the batch job cannot be restarted from a checkpoint, then the batch server shall abort the batch job.

**Resource\_List Attribute**

A batch server that executes a batch job shall establish the resource limits of the session leader of the batch job according to the values of the *Resource\_List* attribute of the batch job. Resource limits shall be enforced by an implementation-defined method.

**Shell\_Path\_List Attribute**

The *Shell\_Path\_List* job attribute consists of a list of pairs of pathname and host name values. The host name component can be omitted, in which case the pathname serves as the default pathname when a batch server cannot find the name of the host on which it is running in the list.

A batch server that executes a batch job shall select, from the value of the *Shell\_Path\_List* attribute of the batch job, a pathname where the shell to execute the batch job shall be found. The batch server shall select the pathname, in order of preference, according to the following methods:

- Select the pathname that contains the name of the host on which the batch server is running.
- Select the pathname for which the host name has been omitted.
- Select the pathname for the login shell of the user under which the batch job is to execute.

If the shell path value selected is an invalid pathname, the batch server shall abort the batch job.

If the value of the selected pathname from the *Shell\_Path\_List* attribute of the batch job represents a partial path, the batch server shall expand the path relative to a path that is implementation-defined.

The batch server that executes the batch job shall execute the program that was selected from the *Shell\_Path\_List* attribute of the batch job. The batch server shall pass the path to the script of the batch job as the first argument to the shell program.

**User\_List Attribute**

The *User\_List* job attribute consists of a list of pairs of user name and host name values. The host name component can be omitted, in which case the user name serves as a default when a batch server cannot find the name of the host on which it is running in the list.

A batch server that executes a batch job shall select, from the value of the *User\_List* attribute of the batch job, a user name under which to create the session leader. The server shall select the user name, in order of preference, according to the following methods:

- Select the user name of a value that contains the name of the host on which the batch server executes.
- Select the user name of a value for which the host name has been omitted.
- Select the user name from the *Job\_Owner* attribute of the batch job.

**Variable\_List Attribute**

A batch server that executes a batch job shall create, in the environment of the session leader of the batch job, each environment variable listed in the *Variable\_List* attribute of the batch job, and set the value of each such environment variable to that of the corresponding variable in the variable list.

**3.2.2.2 Batch Job Routing**

To route a batch job is to select a queue from a list and move the batch job to that queue.

A batch server that has routing queues, which have been started, shall route the jobs in the routing queues owned by the batch server. A batch server may delay the routing of a batch job. The algorithm for selecting a batch job and the queue to which it will be routed is implementation-defined.

When a routing queue has multiple possible destinations specified, then the precedence of the destinations is implementation-defined.

A batch server that routes a batch job to a queue at another server shall move the batch job into the target queue with a *Queue Batch Job Request*.

If the target server rejects the *Queue Batch Job Request*, the routing server shall retry routing the batch job or abort the batch job. A batch server that retries failed routings shall provide a means for the batch administrator to specify the number of retries and the minimum period of time between retries. The means by which an administrator specifies the number of retries and the delay between retries is implementation-defined. When the number of retries specified by the batch administrator has been exhausted, the batch server shall abort the batch job and perform the functions of *Batch Job Exit*; see [Section 3.2.2.3](#).

**3.2.2.3 Batch Job Exit**

For each job in the EXITING state, the batch server that exited the batch job shall perform the following deferred services in the order specified:

1. If buffering standard error, move that file into the location specified by the *Error\_Path* attribute of the batch job.
2. If buffering standard output, move that file into the location specified by the *Output\_Path* attribute of the batch job.

3. If the *Mail\_Points* attribute of the batch job includes MAIL\_AT\_EXIT, send mail to the users listed in the *Mail\_Users* attribute of the batch job. The mail message shall contain at least the batch job identifier, queue, and server at which the batch job currently resides, and the *Job\_Owner* attribute.

4. Remove the batch job from the queue.

If a batch server that buffers the standard error output cannot return the standard error file to the standard error path at the time the batch job exits, the batch server shall do one of the following:

- Mail the standard error file to the batch job owner.
- Save the standard error file and mail the location and name of the file where the standard error is stored to the batch job owner.
- Save the standard error file and notify the user by other implementation-defined means.

If a batch server that buffers the standard output cannot return the standard output file to the standard output path at the time the batch job exits, the batch server shall do one of the following:

- Mail the standard output file to the batch job owner.
- Save the standard output file and mail the location and name of the file where the standard output is stored to the batch job owner.
- Save the standard output file and notify the user by other implementation-defined means.

At the conclusion of job exit processing, the batch job is no longer managed by a batch server.

#### 3.2.2.4 Batch Server Restart

A batch server that has been either shutdown or terminated abnormally, and has returned to operation, is said to have “restarted”.

Upon restarting, a batch server shall requeue those jobs managed by the batch server that were in the RUNNING state at the time the batch server shut down and for which the *Rerunable* attribute of the batch job has the value TRUE.

Queues are defined to be non-volatile. A batch server shall store the content of queues that it controls in such a way that server and system shutdowns do not erase the content of the queues.

#### 3.2.2.5 Batch Job Abort

A batch server that cannot perform a deferred service for a batch job shall abort the batch job.

A batch server that aborts a batch job shall perform the following services:

- Delete the batch job from the queue in which it resides.
- If the *Mail\_Points* attribute of the batch job includes the value MAIL\_AT\_ABORT, send mail to the users listed in the value of the *Mail\_Users* attribute of the job. The mail message shall contain at least the batch job identifier, queue, and server at which the batch job currently resides, the *Job\_Owner* attribute, and the reason for the abort.
- If the batch job was in the RUNNING state, terminate the session leader of the executing job by sending the session leader a SIGKILL, place the batch job in the EXITING state, and perform the actions of *Batch Job Exit*.

### 3.2.3 Requested Batch Services

This section describes the services provided by batch servers in response to requests from clients. Table 3-5 summarizes the current set of batch service requests and for each gives its type (deferred or not) and whether it is an optional function.

**Table 3-5** Batch Services Summary

| Batch Service                     | Deferred | Optional |
|-----------------------------------|----------|----------|
| <i>Batch Job Execution</i>        | Yes      | No       |
| <i>Batch Job Routing</i>          | Yes      | No       |
| <i>Batch Job Exit</i>             | Yes      | No       |
| <i>Batch Server Restart</i>       | Yes      | No       |
| <i>Batch Job Abort</i>            | Yes      | No       |
| <i>Delete Batch Job Request</i>   | No       | No       |
| <i>Hold Batch Job Request</i>     | No       | No       |
| <i>Batch Job Message Request</i>  | No       | Yes      |
| <i>Batch Job Status Request</i>   | No       | No       |
| <i>Locate Batch Job Request</i>   | No       | Yes      |
| <i>Modify Batch Job Request</i>   | No       | No       |
| <i>Move Batch Job Request</i>     | No       | No       |
| <i>Queue Batch Job Request</i>    | No       | No       |
| <i>Batch Queue Status Request</i> | No       | No       |
| <i>Release Batch Job Request</i>  | No       | No       |
| <i>Rerun Batch Job Request</i>    | No       | No       |
| <i>Select Batch Jobs Request</i>  | No       | No       |
| <i>Server Shutdown Request</i>    | No       | No       |
| <i>Server Status Request</i>      | No       | No       |
| <i>Signal Batch Job Request</i>   | No       | No       |
| <i>Track Batch Job Request</i>    | No       | Yes      |

If a request is rejected because the batch client is not authorized to perform the action, the batch server shall return the same status as when the batch job does not exist.

#### 3.2.3.1 Delete Batch Job Request

A batch job is defined to have been deleted when it has been removed from the queue in which it resides and not instantiated in another queue. A client requests that the server that manages a batch job delete the batch job. Such a request is called a *Delete Batch Job Request*.

A batch server shall reject a *Delete Batch Job Request* if any of the following statements are true:

- The user of the batch client is not authorized to delete the designated job.
- The designated job is not managed by the batch server.
- The designated job is in a state inconsistent with the delete request.

A batch server may reject a *Delete Batch Job Request* for other implementation-defined reasons. The method used to determine whether the user of a client is authorized to perform the requested action is implementation-defined.

A batch server requested to delete a batch job shall delete the batch job if the batch job exists and is not in the EXITING state.

A batch server that deletes a batch job in the RUNNING state shall send a SIGKILL signal to the

77937 session leader of the batch job. It is implementation-defined whether additional signals are sent  
77938 to the session leader of the job prior to sending the SIGKILL signal.

77939 A batch server that deletes a batch job in the RUNNING state shall place the batch job in the  
77940 EXITING state after it has killed the session leader of the batch job and shall perform the actions  
77941 of *Batch Job Exit*.

#### 77942 3.2.3.2 *Hold Batch Job Request*

77943 A batch client can request that the batch server add one or more holds to a batch job. Such a  
77944 request is called a *Hold Batch Job Request*.

77945 A batch server shall reject a *Hold Batch Job Request* if any of the following statements are true:

- 77946 • The batch server does not support one or more of the requested holds to be added to the  
77947 batch job.
- 77948 • The user of the batch client is not authorized to add one or more of the requested holds to  
77949 the batch job.
- 77950 • The batch server does not manage the specified job.
- 77951 • The designated job is in the EXITING state.

77952 A batch server may reject a *Hold Batch Job Request* for other implementation-defined reasons. The  
77953 method used to determine whether the user of a client is authorized to perform the requested  
77954 action is implementation-defined.

77955 A batch server that accepts a *Hold Batch Job Request* for a batch job in the RUNNING state shall  
77956 place a hold on the batch job. The effects, if any, the hold will have on a batch job in the  
77957 RUNNING state are implementation-defined.

77958 A batch server that accepts a *Hold Batch Job Request* shall add each type of hold listed in the *Hold*  
77959 *Batch Job Request*, that is not already present, to the value of the *Hold\_Types* attribute of the batch  
77960 job.

#### 77961 3.2.3.3 *Batch Job Message Request*

77962 *Batch Job Message Request* is an optional feature of batch servers. If an implementation supports  
77963 *Batch Job Message Request*, the statements in this section apply and the configuration variable  
77964 POSIX2\_PBS\_MESSAGE shall be set to 1.

77965 A batch client can request that a batch server write a message into certain output files of a batch  
77966 job. Such a request is called a *Batch Job Message Request*.

77967 A batch server shall reject a *Batch Job Message Request* if any of the following statements are true:

- 77968 • The batch server does not support sending messages to jobs.
- 77969 • The user of the batch client is not authorized to post a message to the designated job.
- 77970 • The designated job does not exist on the batch server.
- 77971 • The designated job is not in the RUNNING state.

77972 A batch server may reject a *Batch Job Message Request* for other implementation-defined reasons.  
77973 The method used to determine whether the user of a client is authorized to perform the  
77974 requested action is implementation-defined.

77975 A batch server that accepts a *Batch Job Message Request* shall write the message sent by the batch  
77976 client into the files indicated by the batch client.

77977 3.2.3.4 *Batch Job Status Request*

77978 A batch client can request that a batch server respond with the status and attributes of a batch  
77979 job. Such a request is called a *Batch Job Status Request*.

77980 A batch server shall reject a *Batch Job Status Request* if any of the following statements are true:

- 77981 • The user of the batch client is not authorized to query the status of the designated job.
- 77982 • The designated job is not managed by the batch server.

77983 A batch server may reject a *Batch Job Status Request* for other implementation-defined reasons.  
77984 The method used to determine whether the user of a client is authorized to perform the  
77985 requested action is implementation-defined.

77986 A batch server that accepts a *Batch Job Status Request* shall return a *Batch Job Status Message* to the  
77987 batch client.

77988 A batch server may return other information in response to a *Batch Job Status Request*.

77989 3.2.3.5 *Locate Batch Job Request*

77990 *Locate Batch Job Request* is an optional feature of batch servers. If an implementation supports  
77991 *Locate Batch Job Request*, the statements in this section apply and the configuration variable  
77992 POSIX2\_PBS\_LOCATE shall be set to 1.

77993 A batch client can ask a batch server to respond with the location of a batch job that was created  
77994 by the batch server. Such a request is called a *Locate Batch Job Request*.

77995 A batch server that accepts a *Locate Batch Job Request* shall return a *Batch Job Location Message* to  
77996 the batch client.

77997 A batch server may reject a *Locate Batch Job Request* for a batch job that was not created by that  
77998 server.

77999 A batch server may reject a *Locate Batch Job Request* for a batch job that is no longer managed by  
78000 that server; that is, for a batch job that is not in a queue owned by that server.

78001 A batch server may reject a *Locate Batch Job Request* for other implementation-defined reasons.

78002 3.2.3.6 *Modify Batch Job Request*

78003 Batch clients modify (alter) the attributes of a batch job by making a request to the server that  
78004 manages the batch job. Such a request is called a *Modify Batch Job Request*.

78005 A batch server shall reject a *Modify Batch Job Request* if any of the following statements are true:

- 78006 • The user of the batch client is not authorized to make the requested modification to the  
78007 batch job.
- 78008 • The designated job is not managed by the batch server.
- 78009 • The requested modification is inconsistent with the state of the batch job.
- 78010 • An unrecognized resource is requested for a batch job in an execution queue.

78011 A batch server may reject a *Modify Batch Job Request* for other implementation-defined reasons.  
78012 The method used to determine whether the user of a client is authorized to perform the  
78013 requested action is implementation-defined.

78014 A batch server that accepts a *Modify Batch Job Request* shall modify all the specified attributes of  
78015 the batch job. A batch server that rejects a *Modify Batch Job Request* shall modify none of the



78016 attributes of the batch job.

78017 If the servicing by a batch server of an otherwise valid request would result in no change, then  
78018 the batch server shall indicate successful completion of the request.

#### 78019 3.2.3.7 *Move Batch Job Request*

78020 A batch client can request that a batch server move a batch job to another destination. Such a  
78021 request is called a *Move Batch Job Request*.

78022 A batch server shall reject a *Move Batch Job Request* if any of the following statements are true:

- 78023 • The user of the batch client is not authorized to remove the designated job from the queue  
78024 in which the batch job resides.
- 78025 • The user of the batch client is not authorized to move the designated job to the destination.
- 78026 • The designated job is not managed by the batch server.
- 78027 • The designated job is in the EXITING state.
- 78028 • The destination is inaccessible.

78029 A batch server can reject a *Move Batch Job Request* for other implementation-defined reasons. The  
78030 method used to determine whether the user of a client is authorized to perform the requested  
78031 action is implementation-defined.

78032 A batch server that accepts a *Move Batch Job Request* shall perform the following services:

- 78033 • Queue the designated job at the destination.
- 78034 • Remove the designated job from the queue in which the batch job resides.

78035 If the destination resides on another batch server, the batch server shall queue the batch job at  
78036 the destination by sending a *Queue Batch Job Request* to the other server. If the *Queue Batch Job*  
78037 *Request* fails, the batch server shall reject the *Move Batch Job Request*. If the *Queue Batch Job*  
78038 *Request* succeeds, the batch server shall remove the batch job from its queue.

78039 The batch server shall not modify any attributes of the batch job.

#### 78040 3.2.3.8 *Queue Batch Job Request*

78041 A batch queue is controlled by one and only one batch server. A batch server is said to own the  
78042 queues that it controls. Batch clients make requests of batch servers to have jobs queued. Such a  
78043 request is called a *Queue Batch Job Request*.

78044 A batch server requested to queue a batch job for which the queue is not specified shall select an  
78045 implementation-defined queue for the batch job. Such a queue is called the “default queue” of  
78046 the batch server. The implementation shall provide the means for a batch administrator to  
78047 specify the default queue. The queue, whether specified or defaulted, is called the “target  
78048 queue”.

78049 A batch server shall reject a *Queue Batch Job Request* if any of the following statements are true:

- 78050 • The client is not authorized to create a batch job in the target queue.
- 78051 • The request specifies a queue that does not exist on the batch server.
- 78052 • The target queue is an execution queue and the batch server cannot satisfy a resource  
78053 requirement of the batch job.

- 78054 • The target queue is an execution queue and an unrecognized resource is requested.
- 78055 • The target queue is an execution queue, the batch server does not support checkpointing,
- 78056 and the value of the *Checkpoint* attribute of the batch job is not NO\_CHECKPOINT.
- 78057 • The job requires access to a user identifier that the batch client is not authorized to access.

78058 A batch server may reject a *Queue Batch Job Request* for other implementation-defined reasons.

78059 A batch server that accepts a *Queue Batch Job Request* for a batch job for which the  
78060 PBS\_O\_QUEUE value is missing from the value of the *Variable\_List* attribute of the batch job  
78061 shall add that variable to the list and set the value to the name of the target queue. Once set, no  
78062 server shall change the value of PBS\_O\_QUEUE, even if the batch job is moved to another  
78063 queue.

78064 A batch server that accepts a *Queue Batch Job Request* for a batch job for which the PBS\_JOBID  
78065 value is missing from the value of the *Variable\_List* attribute shall add that variable to the list and  
78066 set the value to the batch job identifier assigned by the server in the format:

78067 `sequence_number.server`

78068 A batch server that accepts a *Queue Batch Job Request* for a batch job for which the  
78069 PBS\_JOBNAME value is missing from the value of the *Variable\_List* attribute of the batch job  
78070 shall add that variable to the list and set the value to the *Job\_Name* attribute of the batch job.

#### 78071 3.2.3.9 *Batch Queue Status Request*

78072 A batch client can request that a batch server respond with the status and attributes of a queue.  
78073 Such a request is called a *Batch Queue Status Request*.

78074 A batch server shall reject a *Batch Queue Status Request* if any of the following statements are  
78075 true:

- 78076 • The user of the batch client is not authorized to query the status of the designated queue.
- 78077 • The designated queue does not exist on the batch server.

78078 A batch server may reject a *Batch Queue Status Request* for other implementation-defined reasons.  
78079 The method used to determine whether the user of a client is authorized to perform the  
78080 requested action is implementation-defined.

78081 A batch server that accepts a *Batch Queue Status Request* shall return a *Batch Queue Status Reply* to  
78082 the batch client.

#### 78083 3.2.3.10 *Release Batch Job Request*

78084 A batch client can request that the server remove one or more holds from a batch job. Such a  
78085 request is called a *Release Batch Job Request*.

78086 A batch server shall reject a *Release Batch Job Request* if any of the following statements are true:

- 78087 • The user of the batch client is not authorized to remove one or more of the requested holds
- 78088 from the batch job.
- 78089 • The batch server does not manage the specified job.

78090 A batch server may reject a *Release Batch Job Request* for other implementation-defined reasons.  
78091 The method used to determine whether the user of a client is authorized to perform the  
78092 requested action is implementation-defined.

78093 A batch server that accepts a *Release Batch Job Request* shall remove each type of hold listed in the



78094 *Release Batch Job Request*, that is present, from the value of the *Hold\_Types* attribute of the batch  
78095 job.

#### 78096 3.2.3.11 *Rerun Batch Job Request*

78097 To rerun a batch job is to kill the session leader of the batch job and leave the batch job eligible  
78098 for re-execution. A batch client can request that a batch server rerun a batch job. Such a request  
78099 is called *Rerun Batch Job Request*.

78100 A batch server shall reject a *Rerun Batch Job Request* if any of the following statements are true:

- 78101 • The user of the batch client is not authorized to rerun the designated job.
- 78102 • The *Rerunable* attribute of the designated job has the value FALSE.
- 78103 • The designated job is not in the RUNNING state.
- 78104 • The batch server does not manage the designated job.

78105 A batch server may reject a *Rerun Batch Job Request* for other implementation-defined reasons.  
78106 The method used to determine whether the user of a client is authorized to perform the  
78107 requested action is implementation-defined.

78108 A batch server that rejects a *Rerun Batch Job Request* shall in no way modify the execution of the  
78109 batch job.

78110 A batch server that accepts a request to rerun a batch job shall perform the following services:

- 78111 • Requeue the batch job in the execution queue in which it was executing.
- 78112 • Send a SIGKILL signal to the process group of the session leader of the batch job.

78113 An implementation may indicate to the batch job owner that the batch job has been rerun.  
78114 Whether and how the batch job owner is notified that a batch job is rerun is implementation-  
78115 defined.

78116 A batch server that reruns a batch job may send other implementation-defined signals to the  
78117 session leader of the batch job prior to sending the SIGKILL signal.

78118 A batch server may preferentially select a rerun job for execution. Whether rerun jobs shall be  
78119 selected for execution before other jobs is implementation-defined.

#### 78120 3.2.3.12 *Select Batch Jobs Request*

78121 A batch client can request from a batch server a list of jobs managed by that server that match a  
78122 list of selection criteria. Such a request is called a *Select Batch Jobs Request*. All the batch jobs  
78123 managed by the batch server that receives the request are candidates for selection.

78124 A batch server that accepts a *Select Batch Jobs Request* shall return a list of zero or more job  
78125 identifiers that correspond to jobs that meet the selection criteria.

78126 If the batch client is not authorized to query the status of a batch job, the batch server shall not  
78127 select the batch job.

78128 3.2.3.13 *Server Shutdown Request*

78129 A batch server is defined to have shut down when it does not respond to requests from clients  
78130 and does not perform deferred services for jobs. A batch client can request that a batch server  
78131 shut down. Such a request is called a *Server Shutdown Request*.

78132 A batch server shall reject a *Server Shutdown Request* from a client that is not authorized to shut  
78133 down the batch server. The method used to determine whether the user of a client is authorized  
78134 to perform the requested action is implementation-defined.

78135 A batch server may reject a *Server Shutdown Request* for other implementation-defined reasons.  
78136 The reasons for which a *Server Shutdown Request* may be rejected are implementation-defined.

78137 At server shutdown, a batch server shall do, in order of preference, one of the following:

- 78138 • If checkpointing is implemented and the batch job is checkpointable, then checkpoint the  
78139 batch job and requeue it.
- 78140 • If the batch job is rerunnable, then requeue the batch job to be rerun (restarted from the  
78141 beginning).
- 78142 • Abort the batch job.

78143 3.2.3.14 *Server Status Request*

78144 A batch client can request that a batch server respond with the status and attributes of the batch  
78145 server. Such a request is called a *Server Status Request*.

78146 A batch server shall reject a *Server Status Request* if the following statement is true:

- 78147 • The user of the batch client is not authorized to query the status of the designated server.

78148 A batch server may reject a *Server Status Request* for other implementation-defined reasons. The  
78149 method used to determine whether the user of a client is authorized to perform the requested  
78150 action is implementation-defined.

78151 A batch server that accepts a *Server Status Request* shall return a *Server Status Reply* to the batch  
78152 client.

78153 3.2.3.15 *Signal Batch Job Request*

78154 A batch client can request that a batch server signal the session leader of a batch job. Such a  
78155 request is called a *Signal Batch Job Request*.

78156 A batch server shall reject a *Signal Batch Job Request* if any of the following statements are true:

- 78157 • The user of the batch client is not authorized to signal the batch job.
- 78158 • The job is not in the RUNNING state.
- 78159 • The batch server does not manage the designated job.
- 78160 • The requested signal is not supported by the implementation.

78161 A batch server may reject a *Signal Batch Job Request* for other implementation-defined reasons.  
78162 The method used to determine whether the user of a client is authorized to perform the  
78163 requested action is implementation-defined.

78164 A batch server that accepts a request to signal a batch job shall send the signal requested by the  
78165 batch client to the process group of the session leader of the batch job.

78166 3.2.3.16 *Track Batch Job Request*

78167 *Track Batch Job Request* is an optional feature of batch servers. If an implementation supports  
 78168 *Track Batch Job Request*, the statements in this section apply and the configuration variable  
 78169 POSIX2\_PBS\_TRACK shall be set to 1.

78170 *Track Batch Job Request* provides a method for tracking the current location of a batch job. Clients  
 78171 may use the tracking information to determine the batch server that should receive a batch  
 78172 server request.

78173 If *Track Batch Job Request* is supported by a batch server, then when the batch server queues a  
 78174 batch job as a result of a *Queue Batch Job Request*, and the batch server is not the batch server that  
 78175 created the batch job, the batch server shall send a *Track Batch Job Request* to the batch server that  
 78176 created the job.

78177 If *Track Batch Job Request* is supported by a batch server, then the *Track Batch Job Request* may also  
 78178 be sent to other servers as a backup to the primary server. The method by which backup servers  
 78179 are specified is implementation-defined.

78180 If *Track Batch Job Request* is supported by a batch server that receives a *Track Batch Job Request*,  
 78181 then the batch server shall record the current location of the batch job as contained in the  
 78182 request.

78183 **3.3 Common Behavior for Batch Environment Utilities**78184 **3.3.1 Batch Job Identifier**

78185 A utility shall recognize *job\_identifiers* of the format:

78186 [ *sequence\_number* ] [ . *server\_name* ] [ @*server* ]

78187 where:

78188 *sequence\_number* An integer that, when combined with *server\_name*, provides a batch job  
 78189 identifier that is unique within the batch system.

78190 *server\_name* The name of the batch server to which the batch job was originally submitted.

78191 *server* The name of the batch server that is currently managing the batch job.

78192 If the application omits the batch *server\_name* portion of a batch job identifier, a utility shall use  
 78193 the name of a default batch server.

78194 If the application omits the batch *server* portion of a batch job identifier, a utility shall use:

- 78195 • The batch server indicated by *server\_name*, if present
- 78196 • The name of the default batch server
- 78197 • The name of the batch server that is currently managing the batch job

78198 If only @*server* is specified, then the status of all jobs owned by the user on the requested server  
 78199 is listed.

78200 The means by which a utility determines the default batch server is implementation-defined.

78201 If the application presents the batch *server* portion of a batch job identifier to a utility, the utility  
 78202 shall send the request to the specified server.

A strictly conforming application shall use the syntax described for the job identifier. Whenever a batch job identifier is specified whose syntax is not recognized by an implementation, then a message for each error that occurs shall be written to standard error and the utility shall exit with an exit status greater than zero.

When a batch job identifier is supplied as an argument to a batch utility and the *server\_name* portion of the batch job identifier is omitted, then the utility shall use the name of the default batch server.

When a batch job identifier is supplied as an argument to a batch utility and the batch *server* portion of the batch job identifier is omitted, then the utility shall use either:

- The name of the default batch server
- or:
- The name of the batch server that is currently managing the batch job

When a batch job identifier is supplied as an argument to a batch utility and the batch *server* portion of the batch job identifier is specified, then the utility shall send the required *Batch Server Request* to the specified server.

### 3.3.2 Destination

The utility shall recognize a *destination* of the format:

[ *queue* ] [ @*server* ]

where:

*queue*            The name of a valid execution or routing queue at the batch server denoted by @*server*, defined as a string of up to 15 alphanumeric characters in the portable character set (see XBD [Section 6.1](#), on page 125) where the first character is alphabetic.

*server*           The name of a batch server, defined as a string of alphanumeric characters in the portable character set.

If the application omits the batch *server* portion of a destination, then the utility shall use either:

- The name of the default batch server
- or:
- The name of the batch server that is currently managing the batch job

The means by which a utility determines the default batch server is implementation-defined.

If the application omits the *queue* portion of a destination, then the utility shall use the name of the default queue at the batch server chosen. The means by which a batch server determines its default queue is implementation-defined. If a destination is specified in the *queue@server* form, then the utility shall use the specified queue at the specified server.

A strictly conforming application shall use the syntax described for a destination. Whenever a destination is specified whose syntax is not recognized by an implementation, then a message shall be written to standard error and the utility shall exit with an exit status greater than zero.

### 3.3.3 Multiple Keyword-Value Pairs

For each option that can have multiple keyword-value pair arguments, the following rules shall apply. Examples of options that can have list-oriented option-arguments are `-u value@keyword` and `-l keyword=value`.

1. If a batch utility is presented with a list-oriented option-argument for which a keyword has a corresponding value that begins with a single or double-quote, then the utility shall stop interpreting the input stream for delimiters until a second single or double-quote, respectively, is encountered. This feature allows some flexibility for a <comma> ( ' , ' ) or <equals-sign> ( ' = ' ) to be part of the value string for a particular keyword; for example:

```
keywd1='val1,val2',keywd2="val3,val4"
```

**Note:** This may require the user to escape the quotes as in the following command:

```
foo -xkeywd1=\'val1,val2\',keywd2=\"val3,val4\"
```

2. If a batch server is presented with a list-oriented attribute that has a keyword that was encountered earlier in the list, then the later entry for that keyword shall replace the earlier entry.
3. If a batch server is presented with a list-oriented attribute that has a keyword without any corresponding value of the form *keyword=* or *@keyword* and the same keyword was encountered earlier in the list, then the prior entry for that keyword shall be ignored by the batch server.
4. If a batch utility is expecting a list-oriented option-argument entry of the form *keyword=value*, but is presented with an entry of the form *keyword* without any corresponding *value*, then the entry shall be treated as though a default value of NULL was assigned (that is, *keyword=NULL*) for entry parsing purposes. The utility shall include only the keyword, not the NULL value, in the associated job attribute.
5. If a batch utility is expecting a list-oriented option-argument entry of the form *value@keyword*, but is presented with an entry of the form *value* without any corresponding *keyword*, then the entry shall be treated as though a keyword of NULL was assigned (that is, *value@NULL*) for entry parsing purposes. The utility shall include only the value, not the NULL keyword, in the associated job attribute.
6. A batch server shall accept a list-oriented attribute that has multiple occurrences of the same keyword, interpreting the keywords, in order, with the last value encountered taking precedence over prior instances of the same keyword. This rule allows, but does not require, a batch utility to preprocess the attribute to remove duplicate keywords.
7. If a batch utility is presented with multiple list-oriented option-arguments on the command line or in script directives, or both, for a single option, then the utility shall concatenate, in order, any command line keyword and value pairs to the end of any directive keyword and value pairs separated by a single <comma> to produce a single string that is an equivalent, valid option-argument. The resulting string shall be assigned to the associated attribute of the batch job (after optionally removing duplicate entries as described in item 6).



78281

*Chapter 4*

78282

# *Utilities*

78283

This chapter contains the definitions of the utilities, as follows:

78284

- Mandatory utilities that are present on every conformant system

78285

- Optional utilities that are present only on systems supporting the associated option; see [Section 1.7.1](#) (on page 7) for information on the options in this volume of POSIX.1-2008

78286

## 78287 NAME

78288 admin — create and administer SCCS files (DEVELOPMENT)

## 78289 SYNOPSIS

```

78290 XSI  admin -i[name] [-n] [-a login] [-d flag] [-e login] [-f flag]
78291         [-m mrlist] [-r rel] [-t[name] [-y[comment]] newfile
78292
78292         admin -n [-a login] [-d flag] [-e login] [-f flag] [-m mrlist]
78293         [-t[name]] [-y[comment]] newfile...
78294
78294         admin [-a login] [-d flag] [-m mrlist] [-r rel] [-t[name]] file...
78295
78295         admin -h file...
78296
78296         admin -z file...

```

## 78297 DESCRIPTION

78298 The *admin* utility shall create new SCCS files or change parameters of existing ones. If a named  
 78299 file does not exist, it shall be created, and its parameters shall be initialized according to the  
 78300 specified options. Parameters not initialized by an option shall be assigned a default value. If a  
 78301 named file does exist, parameters corresponding to specified options shall be changed, and other  
 78302 parameters shall be left as is.

78303 All SCCS filenames supplied by the application shall be of the form *s.filename*. New SCCS files  
 78304 shall be given read-only permission mode. Write permission in the parent directory is required  
 78305 to create a file. All writing done by *admin* shall be to a temporary *x-file*, named *x.filename* (see *get*)  
 78306 created with read-only mode if *admin* is creating a new SCCS file, or created with the same mode  
 78307 as that of the SCCS file if the file already exists. After successful execution of *admin*, the SCCS file  
 78308 shall be removed (if it exists), and the *x-file* shall be renamed with the name of the SCCS file. This  
 78309 ensures that changes are made to the SCCS file only if no errors occur.

78310 The *admin* utility shall also use a transient lock file (named *z.filename*), which is used to prevent  
 78311 simultaneous updates to the SCCS file; see *get*.

## 78312 OPTIONS

78313 The *admin* utility shall conform to XBD [Section 12.2](#) (on page 216), except that the *-i*, *-t*, and *-y*  
 78314 options have optional option-arguments. These optional option-arguments shall not be  
 78315 presented as separate arguments. The following options are supported:

78316 **-n** Create a new SCCS file. When *-n* is used without *-i*, the SCCS file shall be created  
 78317 with control information but without any file data.

78318 **-i[name]** Specify the *name* of a file from which the text for a new SCCS file shall be taken.  
 78319 The text constitutes the first delta of the file (see the *-r* option for the delta  
 78320 numbering scheme). If the *-i* option is used, but the *name* option-argument is  
 78321 omitted, the text shall be obtained by reading the standard input. If this option is  
 78322 omitted, the SCCS file shall be created with control information but without any  
 78323 file data. The *-i* option implies the *-n* option.

78324 **-r SID** Specify the SID of the initial delta to be inserted. This SID shall be a trunk SID; that  
 78325 is, the branch and sequence numbers shall be zero or missing. The level number is  
 78326 optional, and defaults to 1.

78327 **-t[name]** Specify the *name* of a file from which descriptive text for the SCCS file shall be  
 78328 taken. In the case of existing SCCS files (neither *-i* nor *-n* is specified):



|       |                |                                                                                                     |
|-------|----------------|-----------------------------------------------------------------------------------------------------|
| 78329 |                | • A <b>-t</b> option without a <i>name</i> option-argument shall cause the removal of               |
| 78330 |                | descriptive text (if any) currently in the SCCS file.                                               |
| 78331 |                | • A <b>-t</b> option with a <i>name</i> option-argument shall cause the text (if any) in the        |
| 78332 |                | named file to replace the descriptive text (if any) currently in the SCCS file.                     |
| 78333 | <b>-f flag</b> | Specify a <i>flag</i> , and, possibly, a value for the <i>flag</i> , to be placed in the SCCS file. |
| 78334 |                | Several <b>-f</b> options may be supplied on a single <i>admin</i> command line.                    |
| 78335 |                | Implementations shall recognize the following flags and associated values:                          |
| 78336 | <b>b</b>       | Allow use of the <b>-b</b> option on a <i>get</i> command to create branch deltas.                  |
| 78337 | <b>cceil</b>   | Specify the highest release (that is, ceiling), a number less than or equal to                      |
| 78338 |                | 9999, which may be retrieved by a <i>get</i> command for editing. The default                       |
| 78339 |                | value for an unspecified <b>c</b> flag shall be 9999.                                               |
| 78340 | <b>ffloor</b>  | Specify the lowest release (that is, floor), a number greater than 0 but less                       |
| 78341 |                | than 9999, which may be retrieved by a <i>get</i> command for editing. The                          |
| 78342 |                | default value for an unspecified <b>f</b> flag shall be 1.                                          |
| 78343 | <b>dSID</b>    | Specify the default delta number (SID) to be used by a <i>get</i> command.                          |
| 78344 | <b>istr</b>    | Treat the “No ID keywords” message issued by <i>get</i> or <i>delta</i> as a fatal error.           |
| 78345 |                | In the absence of this flag, the message is only a warning. The message is                          |
| 78346 |                | issued if no SCCS identification keywords (see <i>get</i> ) are found in the text                   |
| 78347 |                | retrieved or stored in the SCCS file. If a value is supplied, the application                       |
| 78348 |                | shall ensure that the keywords exactly match the given string; however,                             |
| 78349 |                | the string shall contain a keyword, and no embedded <newline>                                       |
| 78350 |                | characters.                                                                                         |
| 78351 | <b>j</b>       | Allow concurrent <i>get</i> commands for editing on the same SID of an SCCS                         |
| 78352 |                | file. This allows multiple concurrent updates to the same version of the                            |
| 78353 |                | SCCS file.                                                                                          |
| 78354 | <b>l list</b>  | Specify a <i>list</i> of releases to which deltas can no longer be made (that is, <i>get</i>        |
| 78355 |                | <b>-e</b> against one of these locked releases fails). Conforming applications                      |
| 78356 |                | shall use the following syntax to specify a <i>list</i> . Implementations may                       |
| 78357 |                | accept additional forms as an extension:                                                            |
| 78358 |                | <list> ::= a   <range-list>                                                                         |
| 78359 |                | <range-list> ::= <range>   <range-list>, <range>                                                    |
| 78360 |                | <range> ::= <SID>                                                                                   |
| 78361 |                | The character <i>a</i> in the <i>list</i> shall be equivalent to specifying all releases for        |
| 78362 |                | the named SCCS file. The non-terminal <SID> in range shall be the delta                             |
| 78363 |                | number of an existing delta associated with the SCCS file.                                          |
| 78364 | <b>n</b>       | Cause <i>delta</i> to create a null delta in each of those releases (if any) being                  |
| 78365 |                | skipped when a delta is made in a new release (for example, in making                               |
| 78366 |                | delta 5.1 after delta 2.7, releases 3 and 4 are skipped). These null deltas                         |
| 78367 |                | shall serve as anchor points so that branch deltas may later be created                             |
| 78368 |                | from them. The absence of this flag shall cause skipped releases to be                              |
| 78369 |                | nonexistent in the SCCS file, preventing branch deltas from being created                           |
| 78370 |                | from them in the future. During the initial creation of an SCCS file, the <b>n</b>                  |
| 78371 |                | flag may be ignored; that is, if the <b>-r</b> option is used to set the release                    |
| 78372 |                | number of the initial SID to a value greater than 1, null deltas need not be                        |
| 78373 |                | created for the “skipped” releases.                                                                 |

|       |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-------|--------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 78374 | <b>qtext</b>       | Substitute user-definable <i>text</i> for all occurrences of the %Q% keyword in the SCCS file text retrieved by <i>get</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 78375 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78376 | <b>mmod</b>        | Specify the module name of the SCCS file substituted for all occurrences of the %M% keyword in the SCCS file text retrieved by <i>get</i> . If the <b>m</b> flag is not specified, the value assigned shall be the name of the SCCS file with the leading ' . ' removed.                                                                                                                                                                                                                                                                                                                      |
| 78377 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78378 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78379 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78380 | <b>ttype</b>       | Specify the <i>type</i> of module in the SCCS file substituted for all occurrences of the %Y% keyword in the SCCS file text retrieved by <i>get</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 78381 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78382 | <b>vpgm</b>        | Cause <i>delta</i> to prompt for modification request (MR) numbers as the reason for creating a delta. The optional value specifies the name of an MR number validation program. (If this flag is set when creating an SCCS file, the application shall ensure that the <b>m</b> option is also used even if its value is null.)                                                                                                                                                                                                                                                              |
| 78383 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78384 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78385 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78386 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78387 | <b>-d flag</b>     | Remove (delete) the specified <i>flag</i> from an SCCS file. Several <b>-d</b> options may be supplied on a single <i>admin</i> command. See the <b>-f</b> option for allowable <i>flag</i> names. (The <b>l</b> ist flag gives a <i>list</i> of releases to be unlocked. See the <b>-f</b> option for further description of the <b>l</b> flag and the syntax of a <i>list</i> .)                                                                                                                                                                                                            |
| 78388 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78389 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78390 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78391 | <b>-a login</b>    | Specify a <i>login</i> name, or numerical group ID, to be added to the list of users who may make deltas (changes) to the SCCS file. A group ID shall be equivalent to specifying all <i>login</i> names common to that group ID. Several <b>-a</b> options may be used on a single <i>admin</i> command line. As many <i>logins</i> , or numerical group IDs, as desired may be on the list simultaneously. If the list of users is empty, then anyone may add deltas. If <i>login</i> or group ID is preceded by a ' ! ', the users so specified shall be denied permission to make deltas. |
| 78392 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78393 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78394 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78395 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78396 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78397 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78398 | <b>-e login</b>    | Specify a <i>login</i> name, or numerical group ID, to be erased from the list of users allowed to make deltas (changes) to the SCCS file. Specifying a group ID is equivalent to specifying all <i>login</i> names common to that group ID. Several <b>-e</b> options may be used on a single <i>admin</i> command line.                                                                                                                                                                                                                                                                     |
| 78399 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78400 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78401 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78402 | <b>-y[comment]</b> | Insert the <i>comment</i> text into the SCCS file as a comment for the initial delta in a manner identical to that of <i>delta</i> . In the POSIX locale, omission of the <b>-y</b> option shall result in a default comment line being inserted in the form:                                                                                                                                                                                                                                                                                                                                 |
| 78403 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78404 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78405 |                    | "date and time created %s %s by %s", <date>, <time>, <login>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 78406 |                    | where <date> is expressed in the format of the <i>date</i> utility's %Y/%m/%d conversion specification, <time> in the format of the <i>date</i> utility's %T conversion specification format, and <login> is the login name of the user creating the file.                                                                                                                                                                                                                                                                                                                                    |
| 78407 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78408 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78409 | <b>-m mrlist</b>   | Insert the list of modification request (MR) numbers into the SCCS file as the reason for creating the initial delta in a manner identical to <i>delta</i> . The application shall ensure that the <b>v</b> flag is set and the MR numbers are validated if the <b>v</b> flag has a value (the name of an MR number validation program). A diagnostic message shall be written if the <b>v</b> flag is not set or MR validation fails.                                                                                                                                                        |
| 78410 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78411 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78412 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78413 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78414 | <b>-h</b>          | Check the structure of the SCCS file and compare the newly computed checksum with the checksum that is stored in the SCCS file. If the newly computed checksum does not match the checksum in the SCCS file, a diagnostic message shall be written.                                                                                                                                                                                                                                                                                                                                           |
| 78415 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78416 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78417 |                    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |

78418            **-z**            Recompute the SCCS file checksum and store it in the first line of the SCCS file (see  
 78419                            the **-h** option above). Note that use of this option on a truly corrupted file may  
 78420                            prevent future detection of the corruption.

## 78421 OPERANDS

78422            The following operands shall be supported:

78423            *file*            A pathname of an existing SCCS file or a directory. If *file* is a directory, the *admin*  
 78424                            utility shall behave as though each file in the directory were specified as a named  
 78425                            file, except that non-SCCS files (last component of the pathname does not begin  
 78426                            with **s**.) and unreadable files shall be silently ignored.

78427            *newfile*          A pathname of an SCCS file to be created.

78428            If exactly one *file* or *newfile* operand appears, and it is **'-'**, the standard input shall be read; each  
 78429                            line of the standard input shall be taken to be the name of an SCCS file to be processed. Non-  
 78430                            SCCS files and unreadable files shall be silently ignored.

## 78431 STDIN

78432            The standard input shall be a text file used only if **-i** is specified without an option-argument or  
 78433                            if a *file* or *newfile* operand is specified as **'-'**. If the first character of any standard input line is  
 78434                            <SOH> in the POSIX locale, the results are unspecified.

## 78435 INPUT FILES

78436            The existing SCCS files shall be text files of an unspecified format.

78437            The application shall ensure that the file named by the **-i** option's *name* option-argument shall  
 78438                            be a text file; if the first character of any line in this file is <SOH> in the POSIX locale, the results  
 78439                            are unspecified. If this file contains more than 99 999 lines, the number of lines recorded in the  
 78440                            header for this file shall be 99 999 for this delta.

## 78441 ENVIRONMENT VARIABLES

78442            The following environment variables shall affect the execution of *admin*:

78443            *LANG*            Provide a default value for the internationalization variables that are unset or null.  
 78444                            (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 78445                            variables used to determine the values of locale categories.)

78446            *LC\_ALL*            If set to a non-empty string value, override the values of all the other  
 78447                            internationalization variables.

78448            *LC\_CTYPE*          Determine the locale for the interpretation of sequences of bytes of text data as  
 78449                            characters (for example, single-byte as opposed to multi-byte characters in  
 78450                            arguments and input files).

78451            *LC\_MESSAGES*

78452                            Determine the locale that should be used to affect the format and contents of  
 78453                            diagnostic messages written to standard error and the contents of the default **-y**  
 78454                            comment.

78455            *NLSPATH*          Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 78456 ASYNCHRONOUS EVENTS

78457            Default.

## 78458 STDOUT

78459            Not used.

**78460 STDERR**

78461 The standard error shall be used only for diagnostic messages.

**78462 OUTPUT FILES**

78463 Any SCCS files created shall be text files of an unspecified format. During processing of a *file*, a  
78464 locking *z-file*, as described in *get* (on page 2812), may be created and deleted.

**78465 EXTENDED DESCRIPTION**

78466 None.

**78467 EXIT STATUS**

78468 The following exit values shall be returned:

78469 0 Successful completion.

78470 >0 An error occurred.

**78471 CONSEQUENCES OF ERRORS**

78472 Default.

**78473 APPLICATION USAGE**

78474 It is recommended that directories containing SCCS files be writable by the owner only, and that  
78475 SCCS files themselves be read-only. The mode of the directories should allow only the owner to  
78476 modify SCCS files contained in the directories. The mode of the SCCS files prevents any  
78477 modification at all except by SCCS commands.

**78478 EXAMPLES**

78479 None.

**78480 RATIONALE**

78481 None.

**78482 FUTURE DIRECTIONS**

78483 None.

**78484 SEE ALSO**

78485 *delta, get, prs, what*

78486 XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

**78487 CHANGE HISTORY**

78488 First released in Issue 2.

**78489 Issue 6**

78490 The normative text is reworded to avoid use of the term “must” for application requirements,  
78491 and to emphasize the term “shall” for implementation requirements.

78492 The grammar is updated.

78493 The Open Group Base Resolution bwg2001-007 is applied, adding new text to the INPUT FILES  
78494 section warning that the maximum lines recorded in the file is 99 999.

78495 The Open Group Base Resolution bwg2001-009 is applied, amending the description of the **-h**  
78496 option.

78497 **NAME**

78498 alias — define or display aliases

78499 **SYNOPSIS**78500 alias [*alias-name*[=*string*]]...78501 **DESCRIPTION**

78502 The *alias* utility shall create or redefine alias definitions or write the values of existing alias  
 78503 definitions to standard output. An alias definition provides a string value that shall replace a  
 78504 command name when it is encountered; see [Section 2.3.1](#) (on page 2338).

78505 An alias definition shall affect the current shell execution environment and the execution  
 78506 environments of the subshells of the current shell. When used as specified by this volume of  
 78507 POSIX.1-2008, the alias definition shall not affect the parent process of the current shell nor any  
 78508 utility environment invoked by the shell; see [Section 2.12](#) (on page 2371).

78509 **OPTIONS**

78510 None.

78511 **OPERANDS**

78512 The following operands shall be supported:

78513 *alias-name* Write the alias definition to standard output.

78514 *alias-name=string*  
 78515 Assign the value of *string* to the alias *alias-name*.

78516 If no operands are given, all alias definitions shall be written to standard output.

78517 **STDIN**

78518 Not used.

78519 **INPUT FILES**

78520 None.

78521 **ENVIRONMENT VARIABLES**78522 The following environment variables shall affect the execution of *alias*:

78523 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 78524 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 78525 variables used to determine the values of locale categories.)

78526 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 78527 internationalization variables.

78528 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 78529 characters (for example, single-byte as opposed to multi-byte characters in  
 78530 arguments).

78531 *LC\_MESSAGES*  
 78532 Determine the locale that should be used to affect the format and contents of  
 78533 diagnostic messages written to standard error.

78534 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.78535 **ASYNCHRONOUS EVENTS**

78536 Default.

**STDOUT**

The format for displaying aliases (when no operands or only *name* operands are specified) shall be:

```
"%s=%s\n", name, value
```

The *value* string shall be written with appropriate quoting so that it is suitable for reinput to the shell. See the description of shell quoting in [Section 2.2](#) (on page 2336).

**STDERR**

The standard error shall be used only for diagnostic messages.

**OUTPUT FILES**

None.

**EXTENDED DESCRIPTION**

None.

**EXIT STATUS**

The following exit values shall be returned:

0 Successful completion.

>0 One of the *name* operands specified did not have an alias definition, or an error occurred.

**CONSEQUENCES OF ERRORS**

Default.

**APPLICATION USAGE**

None.

**EXAMPLES**

1. Create a short alias for a commonly used *ls* command:

```
alias lf="ls -CF"
```

2. Create a simple “redo” command to repeat previous entries in the command history file:

```
alias r='fc -s'
```

3. Use 1K units for *du*:

```
alias du=du\ -k
```

4. Set up *nohup* so that it can deal with an argument that is itself an alias name:

```
alias nohup="nohup "
```

**RATIONALE**

The *alias* description is based on historical KornShell implementations. Known differences exist between that and the C shell. The KornShell version was adopted to be consistent with all the other KornShell features in this volume of POSIX.1-2008, such as command line editing.

Since *alias* affects the current shell execution environment, it is generally provided as a shell regular built-in.

Historical versions of the KornShell have allowed aliases to be exported to scripts that are invoked by the same shell. This is triggered by the *alias* *-x* flag; it is allowed by this volume of POSIX.1-2008 only when an explicit extension such as *-x* is used. The standard developers considered that aliases were of use primarily to interactive users and that they should normally not affect shell scripts called by those users; functions are available to such scripts.

78577 Historical versions of the KornShell had not written aliases in a quoted manner suitable for  
78578 reentry to the shell, but this volume of POSIX.1-2008 has made this a requirement for all similar  
78579 output. Therefore, consistency was chosen over this detail of historical practice.

78580 **FUTURE DIRECTIONS**

78581 None.

78582 **SEE ALSO**

78583 [Section 2.9.5](#) (on page 2364)

78584 XBD [Chapter 8](#) (on page 173)

78585 **CHANGE HISTORY**

78586 First released in Issue 4.

78587 **Issue 6**

78588 This utility is marked as part of the User Portability Utilities option.

78589 The APPLICATION USAGE section is added.

78590 **Issue 7**

78591 The *alias* utility is moved from the User Portability Utilities option to the Base. User Portability  
78592 Utilities is now an option for interactive utilities.

78593 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

78594 The first example is changed to remove the creation of an alias for a standard utility that alters  
78595 its behavior to be non-conforming.



78596 **NAME**78597 **ar** — create and maintain library archives78598 **SYNOPSIS**78599 SD **ar -d [-v] archive file...**78600 XSI **ar -m [-v] archive file...**78601 **ar -m -a [-v] posname archive file...**78602 **ar -m -b [-v] posname archive file...**78603 **ar -m -i [-v] posname archive file...**78604 XSI **ar -p [-v] [-s] archive [file...]**78605 XSI **ar -q [-cv] archive file...**78606 **ar -r [-cuv] archive file...**78607 XSI **ar -r -a [-cuv] posname archive file...**78608 **ar -r -b [-cuv] posname archive file...**78609 **ar -r -i [-cuv] posname archive file...**78610 XSI **ar -t [-v] [-s] archive [file...]**78611 XSI **ar -x [-v] [-sCT] archive [file...]**78612 **DESCRIPTION**78613 The *ar* utility is part of the Software Development Utilities option.

78614 The *ar* utility can be used to create and maintain groups of files combined into an archive. Once  
 78615 an archive has been created, new files can be added, and existing files in an archive can be  
 78616 extracted, deleted, or replaced. When an archive consists entirely of valid object files, the  
 78617 implementation shall format the archive so that it is usable as a library for link editing (see *c99*  
 78618 and *fort77*). When some of the archived files are not valid object files, the suitability of the  
 78619 XSI archive for library use is undefined. If an archive consists entirely of printable files, the entire  
 78620 archive shall be printable.

78621 When *ar* creates an archive, it creates administrative information indicating whether a symbol  
 78622 table is present in the archive. When there is at least one object file that *ar* recognizes as such in  
 78623 the archive, an archive symbol table shall be created in the archive and maintained by *ar*; it is  
 78624 used by the link editor to search the archive. Whenever the *ar* utility is used to create or update  
 78625 the contents of such an archive, the symbol table shall be rebuilt. The *-s* option shall force the  
 78626 symbol table to be rebuilt.

78627 All *file* operands can be pathnames. However, files within archives shall be named by a filename,  
 78628 which is the last component of the pathname used when the file was entered into the archive.  
 78629 The comparison of *file* operands to the names of files in archives shall be performed by  
 78630 comparing the last component of the operand to the name of the file in the archive.

78631 It is unspecified whether multiple files in the archive may be identically named. In the case of  
 78632 XSI such files, however, each *file* and *posname* operand shall match only the first file in the archive  
 78633 having a name that is the same as the last component of the operand.



## 78634 OPTIONS

78635 The *ar* utility shall conform to XBD Section 12.2 (on page 216), except for Guideline 9.

78636 The following options shall be supported:

- |       |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------|-----|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 78637 | XSI | <b>-a</b> | Position new files in the archive after the file named by the <i>posname</i> operand.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 78638 | XSI | <b>-b</b> | Position new files in the archive before the file named by the <i>posname</i> operand.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 78639 |     | <b>-c</b> | Suppress the diagnostic message that is written to standard error by default when the archive <i>archive</i> is created.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 78640 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78641 | XSI | <b>-C</b> | Prevent extracted files from replacing like-named files in the file system. This option is useful when <b>-T</b> is also used, to prevent truncated filenames from replacing files with the same prefix.                                                                                                                                                                                                                                                                                                                                                                                                       |
| 78642 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78643 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78644 |     | <b>-d</b> | Delete one or more <i>files</i> from <i>archive</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 78645 | XSI | <b>-i</b> | Position new files in the archive before the file in the archive named by the <i>posname</i> operand (equivalent to <b>-b</b> ).                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 78646 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78647 | XSI | <b>-m</b> | Move the named files in the archive. The <b>-a</b> , <b>-b</b> , or <b>-i</b> options with the <i>posname</i> operand indicate the position; otherwise, move the names files in the archive to the end of the archive.                                                                                                                                                                                                                                                                                                                                                                                         |
| 78648 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78649 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78650 |     | <b>-p</b> | Write the contents of the <i>files</i> in the archive named by <i>file</i> operands from <i>archive</i> to the standard output. If no <i>file</i> operands are specified, the contents of all files in the archive shall be written in the order of the archive.                                                                                                                                                                                                                                                                                                                                               |
| 78651 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78652 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78653 | XSI | <b>-q</b> | Append the named files to the end of the archive. In this case <i>ar</i> does not check whether the added files are already in the archive. This is useful to bypass the searching otherwise done when creating a large archive piece by piece.                                                                                                                                                                                                                                                                                                                                                                |
| 78654 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78655 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78656 |     | <b>-r</b> | Replace or add <i>files</i> to <i>archive</i> . If the archive named by <i>archive</i> does not exist, a new archive shall be created and a diagnostic message shall be written to standard error (unless the <b>-c</b> option is specified). If no <i>files</i> are specified and the <i>archive</i> exists, the results are undefined. Files that replace existing files in the archive shall not change the order of the archive. Files that do not replace existing files in the archive shall be appended to the archive unless a <b>-a</b> , <b>-b</b> , or <b>-i</b> option specifies another position. |
| 78657 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78658 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78659 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78660 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78661 | XSI |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78662 | XSI | <b>-s</b> | Force the regeneration of the archive symbol table even if <i>ar</i> is not invoked with an option that modifies the archive contents. This option is useful to restore the archive symbol table after it has been stripped; see <i>strip</i> .                                                                                                                                                                                                                                                                                                                                                                |
| 78663 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78664 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78665 |     | <b>-t</b> | Write a table of contents of <i>archive</i> to the standard output. Only the files specified by the <i>file</i> operands shall be included in the written list. If no <i>file</i> operands are specified, all files in <i>archive</i> shall be included in the order of the archive.                                                                                                                                                                                                                                                                                                                           |
| 78666 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78667 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78668 | XSI | <b>-T</b> | Allow filename truncation of extracted files whose archive names are longer than the file system can support. By default, extracting a file with a name that is too long shall be an error; a diagnostic message shall be written and the file shall not be extracted.                                                                                                                                                                                                                                                                                                                                         |
| 78669 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78670 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78671 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78672 |     | <b>-u</b> | Update older files in the archive. When used with the <b>-r</b> option, files in the archive shall be replaced only if the corresponding <i>file</i> has a modification time that is at least as new as the modification time of the file in the archive.                                                                                                                                                                                                                                                                                                                                                      |
| 78673 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 78674 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

78675        **-v**            Give verbose output. When used with the option characters **-d**, **-r**, or **-x**, write a  
 78676 detailed file-by-file description of the archive creation and maintenance activity, as  
 78677 described in the STDOUT section.

78678                    When used with **-p**, write the name of the file in the archive to the standard output  
 78679 before writing the file in the archive itself to the standard output, as described in  
 78680 the STDOUT section.

78681                    When used with **-t**, include a long listing of information about the files in the  
 78682 archive, as described in the STDOUT section.

78683        **-x**            Extract the files in the archive named by the *file* operands from *archive*. The  
 78684 contents of the archive shall not be changed. If no *file* operands are given, all files  
 78685 in the archive shall be extracted. The modification time of each file extracted shall  
 78686 be set to the time the file is extracted from the archive.

## 78687 OPERANDS

78688            The following operands shall be supported:

78689        *archive*        A pathname of the archive.

78690        *file*            A pathname. Only the last component shall be used when comparing against the  
 78691 names of files in the archive. If two or more *file* operands have the same last  
 78692 pathname component (basename), the results are unspecified. The  
 78693 implementation's archive format shall not truncate valid filenames of files added  
 78694 to or replaced in the archive.

78695 XSI        *posname*        The name of a file in the archive, used for relative positioning; see options **-m** and  
 78696 **-r**.

## 78697 STDIN

78698            Not used.

## 78699 INPUT FILES

78700            The archive named by *archive* shall be a file in the format created by *ar -r*.

## 78701 ENVIRONMENT VARIABLES

78702            The following environment variables shall affect the execution of *ar*:

78703        *LANG*            Provide a default value for the internationalization variables that are unset or null.  
 78704 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 78705 variables used to determine the values of locale categories.)

78706        *LC\_ALL*          If set to a non-empty string value, override the values of all the other  
 78707 internationalization variables.

78708        *LC\_CTYPE*        Determine the locale for the interpretation of sequences of bytes of text data as  
 78709 characters (for example, single-byte as opposed to multi-byte characters in  
 78710 arguments and input files).

78711        *LC\_MESSAGES*

78712                    Determine the locale that should be used to affect the format and contents of  
 78713 diagnostic messages written to standard error.

78714        *LC\_TIME*        Determine the format and content for date and time strings written by *ar -tv*.

78715 XSI        *NLSPATH*        Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

78716 *TMPDIR* Determine the pathname that overrides the default directory for temporary files, if  
78717 any.

78718 *TZ* Determine the timezone used to calculate date and time strings written by *ar -tv*.  
78719 If *TZ* is unset or null, an unspecified default timezone shall be used.

## 78720 ASYNCHRONOUS EVENTS

78721 Default.

## 78722 STDOUT

78723 If the *-d* option is used with the *-v* option, the standard output format shall be:

78724 "d - %s\n", <file>

78725 where *file* is the operand specified on the command line.

78726 If the *-p* option is used with the *-v* option, *ar* shall precede the contents of each file with:

78727 "\n<%s>\n\n", <file>

78728 where *file* is the operand specified on the command line, if *file* operands were specified, and the  
78729 name of the file in the archive if they were not.

78730 If the *-r* option is used with the *-v* option:

- 78731 • If *file* is already in the archive, the standard output format shall be:

78732 "r - %s\n", <file>

78733 where <file> is the operand specified on the command line.

- 78734 • If *file* is not already in the archive, the standard output format shall be:

78735 "a - %s\n", <file>

78736 where <file> is the operand specified on the command line.

78737 If the *-t* option is used, *ar* shall write the names of the files in the archive to the standard output  
78738 in the format:

78739 "%s\n", <file>

78740 where *file* is the operand specified on the command line, if *file* operands were specified, or the  
78741 name of the file in the archive if they were not.

78742 If the *-t* option is used with the *-v* option, the standard output format shall be:

78743 "%s %u/%u %u %s %d %d:%d %d %s\n", <member mode>, <user ID>,  
78744 <group ID>, <number of bytes in member>,  
78745 <abbreviated month>, <day-of-month>, <hour>,  
78746 <minute>, <year>, <file>

78747 where:

78748 <file> Shall be the operand specified on the command line, if *file* operands were specified,  
78749 or the name of the file in the archive if they were not.

78750 <member mode>

78751 Shall be formatted the same as the <file mode> string defined in the STDOUT  
78752 section of *ls*, except that the first character, the <entry type>, is not used; the string  
78753 represents the file mode of the file in the archive at the time it was added to or  
78754 replaced in the archive.

78755 The following represent the last-modification time of a file when it was most recently added to  
78756 or replaced in the archive:

78757 <*abbreviated month*>

78758 Equivalent to the format of the %b conversion specification format in *date*.

78759 <*day-of-month*>

78760 Equivalent to the format of the %e conversion specification format in *date*.

78761 <*hour*> Equivalent to the format of the %H conversion specification format in *date*.

78762 <*minute*> Equivalent to the format of the %M conversion specification format in *date*.

78763 <*year*> Equivalent to the format of the %Y conversion specification format in *date*.

78764 When *LC\_TIME* does not specify the POSIX locale, a different format and order of presentation  
78765 of these fields relative to each other may be used in a format appropriate in the specified locale.

78766 If the *-x* option is used with the *-v* option, the standard output format shall be:

78767 "x - %s\n", <*file*>

78768 where *file* is the operand specified on the command line, if *file* operands were specified, or the  
78769 name of the file in the archive if they were not.

## 78770 STDERR

78771 The standard error shall be used only for diagnostic messages. The diagnostic message about  
78772 creating a new archive when *-c* is not specified shall not modify the exit status.

## 78773 OUTPUT FILES

78774 Archives are files with unspecified formats.

## 78775 EXTENDED DESCRIPTION

78776 None.

## 78777 EXIT STATUS

78778 The following exit values shall be returned:

78779 0 Successful completion.

78780 >0 An error occurred.

## 78781 CONSEQUENCES OF ERRORS

78782 Default.

## 78783 APPLICATION USAGE

78784 None.

## 78785 EXAMPLES

78786 None.

## 78787 RATIONALE

78788 The archive format is not described. It is recognized that there are several known *ar* formats,  
78789 which are not compatible. The *ar* utility is included, however, to allow creation of archives that  
78790 are intended for use only on one machine. The archive is specified as a file, and it can be moved  
78791 as a file. This does allow an archive to be moved from one machine to another machine that uses  
78792 the same implementation of *ar*.

78793 Utilities such as *pax* (and its forebears *tar* and *cpio*) also provide portable “archives”. This is a not  
78794 a duplication; the *ar* utility is included to provide an interface primarily for *make* and the  
78795 compilers, based on a historical model.

In historical implementations, the `-q` option (available on XSI-conforming systems) is known to execute quickly because *ar* does not check on whether the added members are already in the archive. This is useful to bypass the searching otherwise done when creating a large archive piece-by-piece. These remarks may but need not remain true for a brand new implementation of this utility; hence, these remarks have been moved into the RATIONALE.

BSD implementations historically required applications to provide the `-s` option whenever the archive was supposed to contain a symbol table. As in this volume of POSIX.1-2008, System V historically creates or updates an archive symbol table whenever an object file is removed from, added to, or updated in the archive.

The OPERANDS section requires what might seem to be true without specifying it: the archive cannot truncate the filenames below `{NAME_MAX}`. Some historical implementations do so, however, causing unexpected results for the application. Therefore, this volume of POSIX.1-2008 makes the requirement explicit to avoid misunderstandings.

According to the System V documentation, the options `-dmpqrtx` are not required to begin with a `<hyphen-minus>` ('-'). This volume of POSIX.1-2008 requires that a conforming application use the leading `<hyphen-minus>`.

The archive format used by the 4.4 BSD implementation is documented in this RATIONALE as an example:

A file created by *ar* begins with the “magic” string `!<arch>\n`. The rest of the archive is made up of objects, each of which is composed of a header for a file, a possible filename, and the file contents. The header is portable between machine architectures, and, if the file contents are printable, the archive is itself printable.

The header is made up of six ASCII fields, followed by a two-character trailer. The fields are the object name (16 characters), the file last modification time (12 characters), the user and group IDs (each 6 characters), the file mode (8 characters), and the file size (10 characters). All numeric fields are in decimal, except for the file mode, which is in octal.

The modification time is the file `st_mtime` field. The user and group IDs are the file `st_uid` and `st_gid` fields. The file mode is the file `st_mode` field. The file size is the file `st_size` field. The two-byte trailer is the string `"<newline>"`.

Only the name field has any provision for overflow. If any filename is more than 16 characters in length or contains an embedded space, the string `"#1/"` followed by the ASCII length of the name is written in the name field. The file size (stored in the archive header) is incremented by the length of the name. The name is then written immediately following the archive header.

Any unused characters in any of these fields are written as `<space>` characters. If any fields are their particular maximum number of characters in length, there is no separation between the fields.

Objects in the archive are always an even number of bytes long; files that are an odd number of bytes long are padded with a `<newline>`, although the size in the header does not reflect this.

The *ar* utility description requires that (when all its members are valid object files) *ar* produce an object code library, which the linkage editor can use to extract object modules. If the linkage editor needs a symbol table to permit random access to the archive, *ar* must provide it; however, *ar* does not require a symbol table.

The BSD `-o` option was omitted. It is a rare conforming application that uses *ar* to extract object

code from a library with concern for its modification time, since this can only be of importance to *make*. Hence, since this functionality is not deemed important for applications portability, the modification time of the extracted files is set to the current time.

There is at least one known implementation (for a small computer) that can accommodate only object files for that system, disallowing mixed object and other files. The ability to handle any type of file is not only historical practice for most implementations, but is also a reasonable expectation.

Consideration was given to changing the output format of *ar -tv* to the same format as the output of *ls -l*. This would have made parsing the output of *ar* the same as that of *ls*. This was rejected in part because the current *ar* format is commonly used and changes would break historical usage. Second, *ar* gives the user ID and group ID in numeric format separated by a `<slash>`. Changing this to be the user name and group name would not be correct if the archive were moved to a machine that contained a different user database. Since *ar* cannot know whether the archive was generated on the same machine, it cannot tell what to report.

The text on the *-ur* option combination is historical practice—since one filename can easily represent two different files (for example, */a/foo* and */b/foo*), it is reasonable to replace the file in the archive even when the modification time in the archive is identical to that in the file system.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*c99*, *date*, *fort77*, *pax*, *strip*

XBD Chapter 8 (on page 173), Section 12.2 (on page 216), `<unistd.h>`, description of `{POSIX_NO_TRUNC}`

## CHANGE HISTORY

First released in Issue 2.

### Issue 5

The FUTURE DIRECTIONS section is added.

### Issue 6

This utility is marked as part of the Software Development Utilities option.

The STDOUT description is changed for the *-v* option to align with the IEEE P1003.2b draft standard.

The normative text is reworded to avoid use of the term “must” for application requirements.

The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

IEEE PASC Interpretation 1003.2 #198 is applied, changing the description to consistently use “file” to refer to a file in the file system hierarchy, “archive” to refer to the archive being operated upon by the *ar* utility, and “file in the archive” to refer to a copy of a file that is contained in the archive.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/10 is applied, making corrections to the SYNOPSIS. The change was needed since the *-a*, *-b*, and *-i* options are mutually-exclusive, and *posname* is required if any of these options is specified.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/11 is applied, correcting the description of the two-byte trailer in RATIONALE which had missed out a backquote. The correct trailer is a backquote followed by a `<newline>`.

78884 **Issue 7**

78885 SD5-XCU-ERN-6 is applied, clarifying that Guideline 9 of the Utility Syntax Guidelines does not  
78886 apply.

78887 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

78888 The description of the `-t` option is changed to say “Only the files specified ...”.



78889 **NAME**

78890       asa — interpret carriage-control characters

78891 **SYNOPSIS**78892 FR       asa [*file...*]78893 **DESCRIPTION**78894       The *asa* utility shall write its input files to standard output, mapping carriage-control characters  
78895       from the text files to line-printer control sequences in an implementation-defined manner.78896       The first character of every line shall be removed from the input, and the following actions are  
78897       performed.

78898       If the character removed is:

78899       &lt;space&gt;       The rest of the line is output without change.

78900       0            A &lt;newline&gt; is output, then the rest of the input line.

78901       1            One or more implementation-defined characters that causes an advance to the next  
78902       page shall be output, followed by the rest of the input line.78903       +            The <newline> of the previous line shall be replaced with one or more  
78904       implementation-defined characters that causes printing to return to column  
78905       position 1, followed by the rest of the input line. If the '+' is the first character in  
78906       the input, it shall be equivalent to <space>.78907       The action of the *asa* utility is unspecified upon encountering any character other than those  
78908       listed above as the first character in a line.78909 **OPTIONS**

78910       None.

78911 **OPERANDS**78912       *file*        A pathname of a text file used for input. If no *file* operands are specified, the  
78913       standard input shall be used.78914 **STDIN**78915       The standard input shall be used if no *file* operands are specified, and shall be used if a *file*  
78916       operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
78917       the standard input shall not be used. See the INPUT FILES section.78918 **INPUT FILES**

78919       The input files shall be text files.

78920 **ENVIRONMENT VARIABLES**78921       The following environment variables shall affect the execution of *asa*:78922       LANG        Provide a default value for the internationalization variables that are unset or null.  
78923       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
78924       variables used to determine the values of locale categories.)78925       LC\_ALL       If set to a non-empty string value, override the values of all the other  
78926       internationalization variables.78927       LC\_CTYPE    Determine the locale for the interpretation of sequences of bytes of text data as  
78928       characters (for example, single-byte as opposed to multi-byte characters in  
78929       arguments and input files).



78930 **LC\_MESSAGES**

78931 Determine the locale that should be used to affect the format and contents of  
 78932 diagnostic messages written to standard error.

78933 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

78934 **ASYNCHRONOUS EVENTS**

78935 Default.

78936 **STDOUT**

78937 The standard output shall be the text from the input file modified as described in the  
 78938 DESCRIPTION section.

78939 **STDERR**

78940 None.

78941 **OUTPUT FILES**

78942 None.

78943 **EXTENDED DESCRIPTION**

78944 None.

78945 **EXIT STATUS**

78946 The following exit values shall be returned:

78947 0 All input files were output successfully.

78948 >0 An error occurred.

78949 **CONSEQUENCES OF ERRORS**

78950 Default.

78951 **APPLICATION USAGE**

78952 None.

78953 **EXAMPLES**

78954 1. The following command:

78955 *asa file*

78956 permits the viewing of *file* (created by a program using FORTRAN-style carriage-control  
 78957 characters) on a terminal.

78958 2. The following command:

78959 *a.out | asa | lp*

78960 formats the FORTRAN output of **a.out** and directs it to the printer.

78961 **RATIONALE**

78962 The *asa* utility is needed to map “standard” FORTRAN 77 output into a form acceptable to  
 78963 contemporary printers. Usually, *asa* is used to pipe data to the *lp* utility; see *lp*.

78964 This utility is generally used only by FORTRAN programs. The standard developers decided to  
 78965 retain *asa* to avoid breaking the historical large base of FORTRAN applications that put carriage-  
 78966 control characters in their output files. There is no requirement that a system have a FORTRAN  
 78967 compiler in order to run applications that need *asa*.

78968 Historical implementations have used an ASCII <form-feed> in response to a 1 and an ASCII  
 78969 <carriage-return> in response to a '+'. It is suggested that implementations treat characters  
 78970 other than 0, 1, and '+' as <space> in the absence of any compelling reason to do otherwise.

78971 However, the action is listed here as “unspecified”, permitting an implementation to provide  
78972 extensions to access fast multiple-line slewing and channel seeking in a non-portable manner.

78973 **FUTURE DIRECTIONS**

78974 None.

78975 **SEE ALSO**

78976 *fort77, lp*

78977 XBD *Chapter 8* (on page 173)

78978 **CHANGE HISTORY**

78979 First released in Issue 4.

78980 **Issue 6**

78981 This utility is marked as part of the FORTRAN Runtime Utilities option.

78982 The normative text is reworded to avoid use of the term “must” for application requirements.

78983 **Issue 7**

78984 Austin Group Interpretation 1003.1-2001 #092 is applied.

78985 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

78986 **NAME**78987 `at` — execute commands at a later time78988 **SYNOPSIS**78989 `at [-m] [-f file] [-q queuenam] -t time_arg`78990 `at [-m] [-f file] [-q queuenam] timespec...`78991 `at -r at_job_id...`78992 `at -l -q queuenam`78993 `at -l [at_job_id...]`78994 **DESCRIPTION**78995 The *at* utility shall read commands from standard input and group them together as an *at-job*, to  
78996 be executed at a later time.78997 The *at-job* shall be executed in a separate invocation of the shell, running in a separate process  
78998 group with no controlling terminal, except that the environment variables, current working  
78999 directory, file creation mask, and other implementation-defined execution-time attributes in  
79000 effect when the *at* utility is executed shall be retained and used when the *at-job* is executed.79001 When the *at-job* is submitted, the *at\_job\_id* and scheduled time shall be written to standard error.  
79002 The *at\_job\_id* is an identifier that shall be a string consisting solely of alphanumeric characters  
79003 and the <period> character. The *at\_job\_id* shall be assigned by the system when the job is  
79004 scheduled such that it uniquely identifies a particular job.79005 User notification and the processing of the job's standard output and standard error are  
79006 described under the **-m** option.79007 XSI Users shall be permitted to use *at* if their name appears in the file **at.allow** which is located in an  
79008 implementation-defined directory. If that file does not exist, the file **at.deny**, which is located in  
79009 an implementation-defined directory, shall be checked to determine whether the user shall be  
79010 denied access to *at*. If neither file exists, only a process with appropriate privileges shall be  
79011 allowed to submit a job. If only **at.deny** exists and is empty, global usage shall be permitted. The  
79012 **at.allow** and **at.deny** files shall consist of one user name per line.79013 **OPTIONS**79014 The *at* utility shall conform to XBD [Section 12.2](#) (on page 216).

79015 The following options shall be supported:

79016 **-f *file*** Specify the pathname of a file to be used as the source of the *at-job*, instead of  
79017 standard input.79018 **-l** (The letter ell.) Report all jobs scheduled for the invoking user if no *at\_job\_id*  
79019 operands are specified. If *at\_job\_ids* are specified, report only information for these  
79020 jobs. The output shall be written to standard output.79021 **-m** Send mail to the invoking user after the *at-job* has run, announcing its completion.  
79022 Standard output and standard error produced by the *at-job* shall be mailed to the  
79023 user as well, unless redirected elsewhere. Mail shall be sent even if the job  
79024 produces no output.79025 If **-m** is not used, the job's standard output and standard error shall be provided to  
79026 the user by means of mail, unless they are redirected elsewhere; if there is no such  
79027 output to provide, the implementation need not notify the user of the job's  
79028 completion.

|       |                            |                                                                                                            |
|-------|----------------------------|------------------------------------------------------------------------------------------------------------|
| 79029 | <b>-q</b> <i>queuename</i> |                                                                                                            |
| 79030 |                            | Specify in which queue to schedule a job for submission. When used with the <b>-l</b>                      |
| 79031 |                            | option, limit the search to that particular queue. By default, at-jobs shall be                            |
| 79032 |                            | scheduled in queue <i>a</i> . In contrast, queue <i>b</i> shall be reserved for batch jobs; see            |
| 79033 |                            | <i>batch</i> . The meanings of all other <i>queuenames</i> are implementation-defined. If <b>-q</b> is     |
| 79034 |                            | specified along with either of the <b>-t</b> <i>time_arg</i> or <i>timespec</i> arguments, the results are |
| 79035 |                            | unspecified.                                                                                               |
| 79036 | <b>-r</b>                  | Remove the jobs with the specified <i>at_job_id</i> operands that were previously                          |
| 79037 |                            | scheduled by the <i>at</i> utility.                                                                        |
| 79038 | <b>-t</b> <i>time_arg</i>  | Submit the job to be run at the time specified by the <i>time</i> option-argument, which                   |
| 79039 |                            | the application shall ensure has the format as specified by the <i>touch -t time</i> utility.              |

## 79040 OPERANDS

79041 The following operands shall be supported:

|       |                  |                                                                                              |
|-------|------------------|----------------------------------------------------------------------------------------------|
| 79042 | <i>at_job_id</i> | The name reported by a previous invocation of the <i>at</i> utility at the time the job was  |
| 79043 |                  | scheduled.                                                                                   |
| 79044 | <i>timespec</i>  | Submit the job to be run at the date and time specified. All of the <i>timespec</i> operands |
| 79045 |                  | are interpreted as if they were separated by <space> characters and concatenated,            |
| 79046 |                  | and shall be parsed as described in the grammar at the end of this section. The date         |
| 79047 |                  | and time shall be interpreted as being in the timezone of the user (as determined            |
| 79048 |                  | by the <i>TZ</i> variable), unless a timezone name appears as part of <i>time</i> , below.   |
| 79049 |                  | In the POSIX locale, the following describes the three parts of the time specification       |
| 79050 |                  | string. All of the values from the <i>LC_TIME</i> categories in the POSIX locale shall be    |
| 79051 |                  | recognized in a case-insensitive manner.                                                     |
| 79052 | <i>time</i>      | The time can be specified as one, two, or four digits. One-digit and                         |
| 79053 |                  | two-digit numbers shall be taken to be hours; four-digit numbers to                          |
| 79054 |                  | be hours and minutes. The time can alternatively be specified as two                         |
| 79055 |                  | numbers separated by a <colon>, meaning <i>hour:minute</i> . An AM/PM                        |
| 79056 |                  | indication (one of the values from the <b>am_pm</b> keywords in the                          |
| 79057 |                  | <i>LC_TIME</i> locale category) can follow the time; otherwise, a 24-hour                    |
| 79058 |                  | clock time shall be understood. A timezone name can also follow to                           |
| 79059 |                  | further qualify the time. The acceptable timezone names are                                  |
| 79060 |                  | implementation-defined, except that they shall be case-insensitive                           |
| 79061 |                  | and the string <b>utc</b> is supported to indicate the time is in Coordinated                |
| 79062 |                  | Universal Time. In the POSIX locale, the <i>time</i> field can also be one of                |
| 79063 |                  | the following tokens:                                                                        |
| 79064 | <b>midnight</b>  | Indicates the time 12:00 am (00:00).                                                         |
| 79065 | <b>noon</b>      | Indicates the time 12:00 pm.                                                                 |
| 79066 | <b>now</b>       | Indicates the current day and time. Invoking <i>at &lt;now&gt;</i>                           |
| 79067 |                  | shall submit an at-job for potentially immediate                                             |
| 79068 |                  | execution (that is, subject only to unspecified                                              |
| 79069 |                  | scheduling delays).                                                                          |
| 79070 | <i>date</i>      | An optional <i>date</i> can be specified as either a month name (one of the                  |
| 79071 |                  | values from the <b>mon</b> or <b>abmon</b> keywords in the <i>LC_TIME</i> locale             |
| 79072 |                  | category) followed by a day number (and possibly year number                                 |
| 79073 |                  | preceded by a comma), or a day of the week (one of the values from                           |
| 79074 |                  | the <b>day</b> or <b>abday</b> keywords in the <i>LC_TIME</i> locale category). In the       |

79075 POSIX locale, two special days shall be recognized:

79076 **today** Indicates the current day.

79077 **tomorrow** Indicates the day following the current day.

79078 If no *date* is given, **today** shall be assumed if the given time is greater

79079 than the current time, and **tomorrow** shall be assumed if it is less. If

79080 the given month is less than the current month (and no year is given),

79081 next year shall be assumed.

79082 *increment* The optional *increment* shall be a number preceded by a <plus-sign>

79083 ('+') and suffixed by one of the following: **minutes**, **hours**, **days**,

79084 **weeks**, **months**, or **years**. (The singular forms shall also be accepted.)

79085 The keyword **next** shall be equivalent to an increment number of +1.

79086 For example, the following are equivalent commands:

79087 at 2pm + 1 week

79088 at 2pm next week

79089 The following grammar describes the precise format of *timespec* in the POSIX locale. The general

79090 conventions for this style of grammar are described in [Section 1.3](#) (on page 2325). This formal

79091 syntax shall take precedence over the preceding text syntax description. The longest possible

79092 token or delimiter shall be recognized at a given point. When used in a *timespec*, white space

79093 shall also delimit tokens.

79094 %token hr24clock\_hr\_min

79095 %token hr24clock\_hour

79096 /\*

79097 An hr24clock\_hr\_min is a one, two, or four-digit number. A one-digit

79098 or two-digit number constitutes an hr24clock\_hour. An hr24clock\_hour

79099 may be any of the single digits [0,9], or may be double digits, ranging

79100 from [00,23]. If an hr24clock\_hr\_min is a four-digit number, the

79101 first two digits shall be a valid hr24clock\_hour, while the last two

79102 represent the number of minutes, from [00,59].

79103 \*/

79104 %token wallclock\_hr\_min

79105 %token wallclock\_hour

79106 /\*

79107 A wallclock\_hr\_min is a one, two-digit, or four-digit number.

79108 A one-digit or two-digit number constitutes a wallclock\_hour.

79109 A wallclock\_hour may be any of the single digits [1,9], or may

79110 be double digits, ranging from [01,12]. If a wallclock\_hr\_min

79111 is a four-digit number, the first two digits shall be a valid

79112 wallclock\_hour, while the last two represent the number of

79113 minutes, from [00,59].

79114 \*/

79115 %token minute

79116 /\*

79117 A minute is a one or two-digit number whose value can be [0,9]

79118 or [00,59].

79119 \*/

79120 %token day\_number

79121 /\*

```

79122     A day_number is a number in the range appropriate for the particular
79123     month and year specified by month_name and year_number, respectively.
79124     If no year_number is given, the current year is assumed if the given
79125     date and time are later this year. If no year_number is given and
79126     the date and time have already occurred this year and the month is
79127     not the current month, next year is the assumed year.
79128     */
79129     %token year_number
79130     /*
79131     A year_number is a four-digit number representing the year A.D., in
79132     which the at_job is to be run.
79133     */
79134     %token inc_number
79135     /*
79136     The inc_number is the number of times the succeeding increment
79137     period is to be added to the specified date and time.
79138     */
79139     %token timezone_name
79140     /*
79141     The name of an optional timezone suffix to the time field, in an
79142     implementation-defined format.
79143     */
79144     %token month_name
79145     /*
79146     One of the values from the mon or abmon keywords in the LC_TIME
79147     locale category.
79148     */
79149     %token day_of_week
79150     /*
79151     One of the values from the day or abday keywords in the LC_TIME
79152     locale category.
79153     */
79154     %token am_pm
79155     /*
79156     One of the values from the am_pm keyword in the LC_TIME locale
79157     category.
79158     */
79159     %start timespec
79160     %%
79161     timespec      : time
79162                   | time date
79163                   | time increment
79164                   | time date increment
79165                   | nowspec
79166                   ;
79167     nowspec       : "now"
79168                   | "now" increment
79169                   ;

```

```

79170         time          : hr24clock_hr_min
79171                        | hr24clock_hr_min timezone_name
79172                        | hr24clock_hour ":" minute
79173                        | hr24clock_hour ":" minute timezone_name
79174                        | wallclock_hr_min am_pm
79175                        | wallclock_hr_min am_pm timezone_name
79176                        | wallclock_hour ":" minute am_pm
79177                        | wallclock_hour ":" minute am_pm timezone_name
79178                        | "noon"
79179                        | "midnight"
79180                        ;

79181         date           : month_name day_number
79182                        | month_name day_number "," year_number
79183                        | day_of_week
79184                        | "today"
79185                        | "tomorrow"
79186                        ;

79187         increment      : "+" inc_number inc_period
79188                        | "next" inc_period
79189                        ;

79190         inc_period      : "minute" | "minutes"
79191                        | "hour" | "hours"
79192                        | "day" | "days"
79193                        | "week" | "weeks"
79194                        | "month" | "months"
79195                        | "year" | "years"
79196                        ;

```

## 79197 STDIN

79198 The standard input shall be a text file consisting of commands acceptable to the shell command  
 79199 language described in [Chapter 2](#) (on page 2335). The standard input shall only be used if no `-f`  
 79200 *file* option is specified.

## 79201 INPUT FILES

79202 See the STDIN section.

79203 XSI The text files **at.allow** and **at.deny**, which are located in an implementation-defined directory,  
 79204 shall contain zero or more user names, one per line, of users who are, respectively, authorized or  
 79205 denied access to the *at* and *batch* utilities.

## 79206 ENVIRONMENT VARIABLES

79207 The following environment variables shall affect the execution of *at*:

|                         |                 |                                                                                                                                                                                                                                                          |
|-------------------------|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 79208<br>79209<br>79210 | <b>LANG</b>     | Provide a default value for the internationalization variables that are unset or null.<br>(See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization<br>variables used to determine the values of locale categories.) |
| 79211<br>79212          | <b>LC_ALL</b>   | If set to a non-empty string value, override the values of all the other<br>internationalization variables.                                                                                                                                              |
| 79213<br>79214<br>79215 | <b>LC_CTYPE</b> | Determine the locale for the interpretation of sequences of bytes of text data as<br>characters (for example, single-byte as opposed to multi-byte characters in<br>arguments and input files).                                                          |

|       |     |                             |                                                                                                              |
|-------|-----|-----------------------------|--------------------------------------------------------------------------------------------------------------|
| 79216 |     | <b>LC_MESSAGES</b>          |                                                                                                              |
| 79217 |     |                             | Determine the locale that should be used to affect the format and contents of                                |
| 79218 |     |                             | diagnostic messages written to standard error and informative messages written to                            |
| 79219 |     |                             | standard output.                                                                                             |
| 79220 | XSI | <b>NLSPATH</b>              | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                        |
| 79221 |     | <b>LC_TIME</b>              | Determine the format and contents for date and time strings written and accepted                             |
| 79222 |     |                             | by <i>at</i> .                                                                                               |
| 79223 |     | <b>SHELL</b>                | Determine a name of a command interpreter to be used to invoke the at-job. If the                            |
| 79224 |     |                             | variable is unset or null, <i>sh</i> shall be used. If it is set to a value other than a name for            |
| 79225 |     |                             | <i>sh</i> , the implementation shall do one of the following: use that shell; use <i>sh</i> ; use the        |
| 79226 |     |                             | login shell from the user database; or any of the preceding accompanied by a                                 |
| 79227 |     |                             | warning diagnostic about which was chosen.                                                                   |
| 79228 |     | <b>TZ</b>                   | Determine the timezone. The job shall be submitted for execution at the time                                 |
| 79229 |     |                             | specified by <i>timespec</i> or <i>-t time</i> relative to the timezone specified by the <i>TZ</i>           |
| 79230 |     |                             | variable. If <i>timespec</i> specifies a timezone, it shall override <i>TZ</i> . If <i>timespec</i> does not |
| 79231 |     |                             | specify a timezone and <i>TZ</i> is unset or null, an unspecified default timezone shall                     |
| 79232 |     |                             | be used.                                                                                                     |
| 79233 |     | <b>ASYNCHRONOUS EVENTS</b>  |                                                                                                              |
| 79234 |     |                             | Default.                                                                                                     |
| 79235 |     | <b>STDOUT</b>               |                                                                                                              |
| 79236 |     |                             | When standard input is a terminal, prompts of unspecified format for each line of the user input             |
| 79237 |     |                             | described in the STDIN section may be written to standard output.                                            |
| 79238 |     |                             | In the POSIX locale, the following shall be written to the standard output for each job when jobs            |
| 79239 |     |                             | are listed in response to the <i>-l</i> option:                                                              |
| 79240 |     |                             | "%s\t%s\n", <i>at_job_id</i> , < <i>date</i> >                                                               |
| 79241 |     |                             | where <i>date</i> shall be equivalent in format to the output of:                                            |
| 79242 |     |                             | date +"%a %b %e %T %Y"                                                                                       |
| 79243 |     |                             | The date and time written shall be adjusted so that they appear in the timezone of the user (as              |
| 79244 |     |                             | determined by the <i>TZ</i> variable).                                                                       |
| 79245 |     | <b>STDERR</b>               |                                                                                                              |
| 79246 |     |                             | In the POSIX locale, the following shall be written to standard error when a job has been                    |
| 79247 |     |                             | successfully submitted:                                                                                      |
| 79248 |     |                             | "job %s at %s\n", <i>at_job_id</i> , < <i>date</i> >                                                         |
| 79249 |     |                             | where <i>date</i> has the same format as that described in the STDOUT section. Neither this, nor             |
| 79250 |     |                             | warning messages concerning the selection of the command interpreter, shall be considered a                  |
| 79251 |     |                             | diagnostic that changes the exit status.                                                                     |
| 79252 |     |                             | Diagnostic messages, if any, shall be written to standard error.                                             |
| 79253 |     | <b>OUTPUT FILES</b>         |                                                                                                              |
| 79254 |     |                             | None.                                                                                                        |
| 79255 |     | <b>EXTENDED DESCRIPTION</b> |                                                                                                              |
| 79256 |     |                             | None.                                                                                                        |



**EXIT STATUS**

The following exit values shall be returned:

- 0 The *at* utility successfully submitted, removed, or listed a job or jobs.
- >0 An error occurred.

**CONSEQUENCES OF ERRORS**

The job shall not be scheduled, removed, or listed.

**APPLICATION USAGE**

The format of the *at* command line shown here is guaranteed only for the POSIX locale. Other cultures may be supported with substantially different interfaces, although implementations are encouraged to provide comparable levels of functionality.

Since the commands run in a separate shell invocation, running in a separate process group with no controlling terminal, open file descriptors, traps, and priority inherited from the invoking environment are lost.

Some implementations do not allow substitution of different shells using *SHELL*. System V systems, for example, have used the login shell value for the user in */etc/passwd*. To select reliably another command interpreter, the user must include it as part of the script, such as:

```
$ at 1800
myshell myscript
EOT
job ... at ...
$
```

**EXAMPLES**

1. This sequence can be used at a terminal:

```
at -m 0730 tomorrow
sort < file >outfile
EOT
```

2. This sequence, which demonstrates redirecting standard error to a pipe, is useful in a command procedure (the sequence of output redirection specifications is significant):

```
at now + 1 hour <<!
diff file1 file2 2>&1 >outfile | mailx mygroup
!
```

3. To have a job reschedule itself, *at* can be invoked from within the *at*-job. For example, this daily processing script named **my.daily** runs every day (although *crontab* is a more appropriate vehicle for such work):

```
# my.daily runs every day
daily processing
at now tomorrow < my.daily
```

4. The spacing of the three portions of the POSIX locale *timespec* is quite flexible as long as there are no ambiguities. Examples of various times and operand presentation include:

```
at 0815am Jan 24
at 8 :15amjan24
at now "+ 1day"
at 5 pm FRIday
at '17
```

```
79301         utc+
79302         30minutes'
```

## 79303 RATIONALE

79304 The *at* utility reads from standard input the commands to be executed at a later time. It may be  
79305 useful to redirect standard output and standard error within the specified commands.

79306 The *-t time* option was added as a new capability to support an internationalized way of  
79307 specifying a time for execution of the submitted job.

79308 Early proposals added a “jobname” concept as a way of giving submitted jobs names that are  
79309 meaningful to the user submitting them. The historical, system-specified *at\_job\_id* gives no  
79310 indication of what the job is. Upon further reflection, it was decided that the benefit of this was  
79311 not worth the change in historical interface. The *at* functionality is useful in simple  
79312 environments, but in large or complex situations, the functionality provided by the Batch  
79313 Services option is more suitable.

79314 The *-q* option historically has been an undocumented option, used mainly by the *batch* utility.

79315 The System V *-m* option was added to provide a method for informing users that an *at*-job had  
79316 completed. Otherwise, users are only informed when output to standard error or standard  
79317 output are not redirected.

79318 The behavior of *at <now>* was changed in an early proposal from being unspecified to  
79319 submitting a job for potentially immediate execution. Historical BSD *at* implementations support  
79320 this. Historical System V implementations give an error in that case, but a change to the System  
79321 V versions should have no backwards-compatibility ramifications.

79322 On BSD-based systems, a *-u user* option has allowed those with appropriate privileges to access  
79323 the work of other users. Since this is primarily a system administration feature and is not  
79324 universally implemented, it has been omitted. Similarly, a specification for the output format for  
79325 a user with appropriate privileges viewing the queues of other users has been omitted.

79326 The *-f file* option from System V is used instead of the BSD method of using the last operand as  
79327 the pathname. The BSD method is ambiguous—does:

```
79328 at 1200 friday
```

79329 mean the same thing if there is a file named **friday** in the current directory?

79330 The *at\_job\_id* is composed of a limited character set in historical practice, and it is mandated here  
79331 to invalidate systems that might try using characters that require shell quoting or that could not  
79332 be easily parsed by shell scripts.

79333 The *at* utility varies between System V and BSD systems in the way timezones are used. On  
79334 System V systems, the *TZ* variable affects the *at*-job submission times and the times displayed  
79335 for the user. On BSD systems, *TZ* is not taken into account. The BSD behavior is easily achieved  
79336 with the current specification. If the user wishes to have the timezone default to that of the  
79337 system, they merely need to issue the *at* command immediately following an unsetting or null  
79338 assignment to *TZ*. For example:

```
79339 TZ= at noon ...
```

79340 gives the desired BSD result.

79341 While the *yacc*-like grammar specified in the OPERANDS section is lexically unambiguous with  
79342 respect to the digit strings, a lexical analyzer would probably be written to look for and return  
79343 digit strings in those cases. The parser could then check whether the digit string returned is a  
79344 valid *day\_number*, *year\_number*, and so on, based on the context.

79345 **FUTURE DIRECTIONS**

79346 None.

79347 **SEE ALSO**79348 *batch, crontab*79349 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)79350 **CHANGE HISTORY**

79351 First released in Issue 2.

79352 **Issue 6**

79353 This utility is marked as part of the User Portability Utilities option.

79354 The following new requirements on POSIX implementations derive from alignment with the  
79355 Single UNIX Specification:

- 79356
- If **-m** is not used, the job's standard output and standard error are provided to the user by
- 
- 79357 mail.

79358 The effects of using the **-q** and **-t** options as defined in the IEEE P1003.2b draft standard are  
79359 specified.

79360 The normative text is reworded to avoid use of the term “must” for application requirements.

79361 **Issue 7**79362 The *at* utility is moved from the User Portability Utilities option to the Base. User Portability  
79363 Utilities is now an option for interactive utilities.79364 SD5-XCU-ERN-95 is applied, removing the references to fixed locations for the files referenced  
79365 by the *at* utility.

79366 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

**NAME**

awk — pattern scanning and processing language

**SYNOPSIS**

awk [-F *sepstring*] [-v *assignment*]... *program* [*argument*...]

awk [-F *sepstring*] -f *progfile* [-f *progfile*]... [-v *assignment*]...  
[*argument*...]

**DESCRIPTION**

The *awk* utility shall execute programs written in the *awk* programming language, which is specialized for textual data manipulation. An *awk* program is a sequence of patterns and corresponding actions. When input is read that matches a pattern, the action associated with that pattern is carried out.

Input shall be interpreted as a sequence of records. By default, a record is a line, less its terminating <newline>, but this can be changed by using the **RS** built-in variable. Each record of input shall be matched in turn against each pattern in the program. For each pattern matched, the associated action shall be executed.

The *awk* utility shall interpret each input record as a sequence of fields where, by default, a field is a string of non-<blank> non-<newline> characters. This default <blank> and <newline> field delimiter can be changed by using the **FS** built-in variable or the **-F *sepstring*** option. The *awk* utility shall denote the first field in a record \$1, the second \$2, and so on. The symbol \$0 shall refer to the entire record; setting any other field causes the re-evaluation of \$0. Assigning to \$0 shall reset the values of all other fields and the **NF** built-in variable.

**OPTIONS**

The *awk* utility shall conform to XBD [Section 12.2](#) (on page 216).

The following options shall be supported:

**-F *sepstring*** Define the input field separator. This option shall be equivalent to:

**-v FS=*sepstring***

except that if **-F *sepstring*** and **-v FS=*sepstring*** are both used, it is unspecified whether the **FS** assignment resulting from **-F *sepstring*** is processed in command line order or is processed after the last **-v FS=*sepstring***. See the description of the **FS** built-in variable, and how it is used, in the EXTENDED DESCRIPTION section.

**-f *progfile*** Specify the pathname of the file *progfile* containing an *awk* program. A pathname of '-' shall denote the standard input. If multiple instances of this option are specified, the concatenation of the files specified as *progfile* in the order specified shall be the *awk* program. The *awk* program can alternatively be specified in the command line as a single argument.

**-v *assignment***

The application shall ensure that the *assignment* argument is in the same form as an *assignment* operand. The specified variable assignment shall occur prior to executing the *awk* program, including the actions associated with **BEGIN** patterns (if any). Multiple occurrences of this option can be specified.

**OPERANDS**

The following operands shall be supported:

*program* If no **-f** option is specified, the first operand to *awk* shall be the text of the *awk* program. The application shall supply the *program* operand as a single argument to *awk*. If the text does not end in a <newline>, *awk* shall interpret the text as if it did.

79412 *argument* Either of the following two types of *argument* can be intermixed:

79413 *file* A pathname of a file that contains the input to be read, which is

79414 matched against the set of patterns in the program. If no *file* operands

79415 are specified, or if a *file* operand is *'-'*, the standard input shall be

79416 used.

79417 *assignment* An operand that begins with an <underscore> or alphabetic

79418 character from the portable character set (see the table in XBD [Section](#)

79419 [6.1](#), on page 125), followed by a sequence of underscores, digits, and

79420 alphabetic characters from the portable character set, followed by the *'='*

79421 character, shall specify a variable assignment rather than a pathname.

79422 The characters before the *'='* represent the name of an *awk* variable;

79423 if that name is an *awk* reserved word (see [Grammar](#), on page 2490)

79424 the behavior is undefined. The characters following the <equals-

79425 sign> shall be interpreted as if they appeared in the *awk* program

79426 preceded and followed by a double-quote (*'"*) character, as a

79427 **STRING** token (see [Grammar](#), on page 2490), except that if the last

79428 character is an unescaped <backslash>, it shall be interpreted as a

79429 literal <backslash> rather than as the first character of the sequence

79430 *"\"*. The variable shall be assigned the value of that **STRING**

79431 token and, if appropriate, shall be considered a *numeric string* (see

79432 [Expressions in awk](#), on page 2475), the variable shall also be assigned

79433 its numeric value. Each such variable assignment shall occur just

79434 prior to the processing of the following *file*, if any. Thus, an

79435 assignment before the first *file* argument shall be executed after the

79436 **BEGIN** actions (if any), while an assignment after the last *file*

79437 argument shall occur before the **END** actions (if any). If there are no

79438 *file* arguments, assignments shall be executed before processing the

79439 standard input.

#### 79440 STDIN

79441 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is *'-'*,

79442 or if a *progfile* option-argument is *'-'*; see the INPUT FILES section. If the *awk* program contains

79443 no actions and no patterns, but is otherwise a valid *awk* program, standard input and any *file*

79444 operands shall not be read and *awk* shall exit with a return status of zero.

#### 79445 INPUT FILES

79446 Input files to the *awk* program from any of the following sources shall be text files:

- 79447 • Any *file* operands or their equivalents, achieved by modifying the *awk* variables **ARGV**
- 79448 and **ARGC**
- 79449 • Standard input in the absence of any *file* operands
- 79450 • Arguments to the **getline** function

79451 Whether the variable **RS** is set to a value other than a <newline> or not, for these files,

79452 implementations shall support records terminated with the specified separator up to

79453 {**LINE\_MAX**} bytes and may support longer records.

79454 If *-f progfile* is specified, the application shall ensure that the files named by each of the *progfile*

79455 option-arguments are text files and their concatenation, in the same order as they appear in the

79456 arguments, is an *awk* program.

**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *awk*:

**LANG** Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

**LC\_ALL** If set to a non-empty string value, override the values of all the other internationalization variables.

**LC\_COLLATE**

Determine the locale for the behavior of ranges, equivalence classes, and multi-character collating elements within regular expressions and in comparisons of string values.

**LC\_CTYPE**

Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files), the behavior of character classes within regular expressions, the identification of characters as letters, and the mapping of uppercase and lowercase characters for the **toupper** and **tolower** functions.

**LC\_MESSAGES**

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

**LC\_NUMERIC**

Determine the radix character used when interpreting numeric input, performing conversions between numeric and string values, and formatting numeric output. Regardless of locale, the <period> character (the decimal-point character of the POSIX locale) is the decimal-point character recognized in processing *awk* programs (including assignments in command line arguments).

**NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

**PATH** Determine the search path when looking for commands executed by *system(expr)*, or input and output pipes; see XBD [Chapter 8](#) (on page 173).

In addition, all environment variables shall be visible via the *awk* variable **ENVIRON**.

**ASYNCHRONOUS EVENTS**

Default.

**STDOUT**

The nature of the output files depends on the *awk* program.

**STDERR**

The standard error shall be used only for diagnostic messages.

**OUTPUT FILES**

The nature of the output files depends on the *awk* program.

## EXTENDED DESCRIPTION

## Overall Program Structure

An *awk* program is composed of pairs of the form:

```
pattern { action }
```

Either the pattern or the action (including the enclosing brace characters) can be omitted.

A missing pattern shall match any record of input, and a missing action shall be equivalent to:

```
{ print }
```

Execution of the *awk* program shall start by first executing the actions associated with all **BEGIN** patterns in the order they occur in the program. Then each *file* operand (or standard input if no files were specified) shall be processed in turn by reading data from the file until a record separator is seen (<newline> by default). Before the first reference to a field in the record is evaluated, the record shall be split into fields, according to the rules in [Regular Expressions](#) (on page 2481), using the value of **FS** that was current at the time the record was read. Each pattern in the program then shall be evaluated in the order of occurrence, and the action associated with each pattern that matches the current record executed. The action for a matching pattern shall be executed before evaluating subsequent patterns. Finally, the actions associated with all **END** patterns shall be executed in the order they occur in the program.

## Expressions in awk

Expressions describe computations used in *patterns* and *actions*. In the following table, valid expression operations are given in groups from highest precedence first to lowest precedence last, with equal-precedence operators grouped between horizontal lines. In expression evaluation, where the grammar is formally ambiguous, higher precedence operators shall be evaluated before lower precedence operators. In this table *expr*, *expr1*, *expr2*, and *expr3* represent any expression, while *lvalue* represents any entity that can be assigned to (that is, on the left side of an assignment operator). The precise syntax of expressions is given in [Grammar](#) (on page 2490).

Table 4-1 Expressions in Decreasing Precedence in *awk*

| Syntax                    | Name            | Type of Result      | Associativity |
|---------------------------|-----------------|---------------------|---------------|
| ( <i>expr</i> )           | Grouping        | Type of <i>expr</i> | N/A           |
| <i>\$expr</i>             | Field reference | String              | N/A           |
| <i>lvalue</i> ++          | Post-increment  | Numeric             | N/A           |
| <i>lvalue</i> --          | Post-decrement  | Numeric             | N/A           |
| ++ <i>lvalue</i>          | Pre-increment   | Numeric             | N/A           |
| -- <i>lvalue</i>          | Pre-decrement   | Numeric             | N/A           |
| <i>expr</i> ^ <i>expr</i> | Exponentiation  | Numeric             | Right         |
| ! <i>expr</i>             | Logical not     | Numeric             | N/A           |
| + <i>expr</i>             | Unary plus      | Numeric             | N/A           |
| - <i>expr</i>             | Unary minus     | Numeric             | N/A           |
| <i>expr</i> * <i>expr</i> | Multiplication  | Numeric             | Left          |
| <i>expr</i> / <i>expr</i> | Division        | Numeric             | Left          |
| <i>expr</i> % <i>expr</i> | Modulus         | Numeric             | Left          |



| Syntax                                     | Name                             | Type of Result                                   | Associativity |
|--------------------------------------------|----------------------------------|--------------------------------------------------|---------------|
| <i>expr</i> + <i>expr</i>                  | Addition                         | Numeric                                          | Left          |
| <i>expr</i> - <i>expr</i>                  | Subtraction                      | Numeric                                          | Left          |
| <i>expr</i> <i>expr</i>                    | String concatenation             | String                                           | Left          |
| <i>expr</i> < <i>expr</i>                  | Less than                        | Numeric                                          | None          |
| <i>expr</i> <= <i>expr</i>                 | Less than or equal to            | Numeric                                          | None          |
| <i>expr</i> != <i>expr</i>                 | Not equal to                     | Numeric                                          | None          |
| <i>expr</i> == <i>expr</i>                 | Equal to                         | Numeric                                          | None          |
| <i>expr</i> > <i>expr</i>                  | Greater than                     | Numeric                                          | None          |
| <i>expr</i> >= <i>expr</i>                 | Greater than or equal to         | Numeric                                          | None          |
| <i>expr</i> ~ <i>expr</i>                  | ERE match                        | Numeric                                          | None          |
| <i>expr</i> !~ <i>expr</i>                 | ERE non-match                    | Numeric                                          | None          |
| <i>expr</i> in array                       | Array membership                 | Numeric                                          | Left          |
| ( <i>index</i> ) in array                  | Multi-dimension array membership | Numeric                                          | Left          |
| <i>expr</i> && <i>expr</i>                 | Logical AND                      | Numeric                                          | Left          |
| <i>expr</i>    <i>expr</i>                 | Logical OR                       | Numeric                                          | Left          |
| <i>expr1</i> ? <i>expr2</i> : <i>expr3</i> | Conditional expression           | Type of selected<br><i>expr2</i> or <i>expr3</i> | Right         |
| <i>lvalue</i> ^= <i>expr</i>               | Exponentiation assignment        | Numeric                                          | Right         |
| <i>lvalue</i> %= <i>expr</i>               | Modulus assignment               | Numeric                                          | Right         |
| <i>lvalue</i> *= <i>expr</i>               | Multiplication assignment        | Numeric                                          | Right         |
| <i>lvalue</i> /= <i>expr</i>               | Division assignment              | Numeric                                          | Right         |
| <i>lvalue</i> += <i>expr</i>               | Addition assignment              | Numeric                                          | Right         |
| <i>lvalue</i> -= <i>expr</i>               | Subtraction assignment           | Numeric                                          | Right         |
| <i>lvalue</i> = <i>expr</i>                | Assignment                       | Type of <i>expr</i>                              | Right         |

Each expression shall have either a string value, a numeric value, or both. Except as stated for specific contexts, the value of an expression shall be implicitly converted to the type needed for the context in which it is used. A string value shall be converted to a numeric value either by the equivalent of the following calls to functions defined by the ISO C standard:

```
setlocale(LC_NUMERIC, "");
numeric_value = atof(string_value);
```

or by converting the initial portion of the string to type **double** representation as follows:

The input string is decomposed into two parts: an initial, possibly empty, sequence of white-space characters (as specified by *isspace()*) and a subject sequence interpreted as a floating-point constant.

The expected form of the subject sequence is an optional '+' or '-' sign, then a non-empty sequence of digits optionally containing a <period>, then an optional exponent part. An exponent part consists of 'e' or 'E', followed by an optional sign, followed by one or more decimal digits.

The sequence starting with the first digit or the <period> (whichever occurs first) is interpreted as a floating constant of the C language, and if neither an exponent part nor a <period> appears, a <period> is assumed to follow the last digit in the string. If the subject sequence begins with a <hyphen-minus>, the value resulting from the conversion is negated.

A numeric value that is exactly equal to the value of an integer (see [Section 1.1.2](#), on page 2321)



shall be converted to a string by the equivalent of a call to the **sprintf** function (see [String Functions](#), on page 2487) with the string "%d" as the *fmt* argument and the numeric value being converted as the first and only *expr* argument. Any other numeric value shall be converted to a string by the equivalent of a call to the **sprintf** function with the value of the variable **CONVFMT** as the *fmt* argument and the numeric value being converted as the first and only *expr* argument. The result of the conversion is unspecified if the value of **CONVFMT** is not a floating-point format specification. This volume of POSIX.1-2008 specifies no explicit conversions between numbers and strings. An application can force an expression to be treated as a number by adding zero to it, or can force it to be treated as a string by concatenating the null string (" ") to it.

A string value shall be considered a *numeric string* if it comes from one of the following:

1. Field variables
2. Input from the *getline()* function
3. **FILENAME**
4. **ARGV** array elements
5. **ENVIRON** array elements
6. Array elements created by the *split()* function
7. A command line variable assignment
8. Variable assignment from another numeric string variable

and an implementation-dependent condition corresponding to either case (a) or (b) below is met.

- a. After the equivalent of the following calls to functions defined by the ISO C standard, *string\_value\_end* would differ from *string\_value*, and any characters before the terminating null character in *string\_value\_end* would be <blank> characters:

```
char *string_value_end;
setlocale(LC_NUMERIC, "");
numeric_value = strtod (string_value, &string_value_end);
```

- b. After all the following conversions have been applied, the resulting string would lexically be recognized as a **NUMBER** token as described by the lexical conventions in [Grammar](#) (on page 2490):

- All leading and trailing <blank> characters are discarded.
- If the first non-<blank> is '+' or '-', it is discarded.
- Each occurrence of the decimal point character from the current locale is changed to a <period>.

In case (a) the numeric value of the *numeric string* shall be the value that would be returned by the *strtod()* call. In case (b) if the first non-<blank> is '-', the numeric value of the *numeric string* shall be the negation of the numeric value of the recognized **NUMBER** token; otherwise, the numeric value of the *numeric string* shall be the numeric value of the recognized **NUMBER** token. Whether or not a string is a *numeric string* shall be relevant only in contexts where that term is used in this section.

When an expression is used in a Boolean context, if it has a numeric value, a value of zero shall be treated as false and any other value shall be treated as true. Otherwise, a string value of the null string shall be treated as false and any other value shall be treated as true. A Boolean

context shall be one of the following:

- The first subexpression of a conditional expression
- An expression operated on by logical NOT, logical AND, or logical OR
- The second expression of a **for** statement
- The expression of an **if** statement
- The expression of the **while** clause in either a **while** or **do...while** statement
- An expression used as a pattern (as in Overall Program Structure)

All arithmetic shall follow the semantics of floating-point arithmetic as specified by the ISO C standard (see [Section 1.1.2](#), on page 2321).

The value of the expression:

```
expr1 ^ expr2
```

shall be equivalent to the value returned by the ISO C standard function call:

```
pow(expr1, expr2)
```

The expression:

```
lvalue ^= expr
```

shall be equivalent to the ISO C standard expression:

```
lvalue = pow(lvalue, expr)
```

except that lvalue shall be evaluated only once. The value of the expression:

```
expr1 % expr2
```

shall be equivalent to the value returned by the ISO C standard function call:

```
fmod(expr1, expr2)
```

The expression:

```
lvalue %= expr
```

shall be equivalent to the ISO C standard expression:

```
lvalue = fmod(lvalue, expr)
```

except that lvalue shall be evaluated only once.

Variables and fields shall be set by the assignment statement:

```
lvalue = expression
```

and the type of *expression* shall determine the resulting variable type. The assignment includes the arithmetic assignments ("**+=**", "**-=**", "**\*=**", "**/=**", "**%=**", "**^=**", "**++**", "**--**") all of which shall produce a numeric result. The left-hand side of an assignment and the target of increment and decrement operators can be one of a variable, an array with index, or a field selector.

The *awk* language supplies arrays that are used for storing numbers or strings. Arrays need not be declared. They shall initially be empty, and their sizes shall change dynamically. The subscripts, or element identifiers, are strings, providing a type of associative array capability. An array name followed by a subscript within square brackets can be used as an lvalue and thus as an expression, as described in the grammar; see [Grammar](#) (on page 2490). Unsubscripted array names can be used in only the following contexts:

- A parameter in a function definition or function call
- The **NAME** token following any use of the keyword **in** as specified in the grammar (see [Grammar](#), on page 2490); if the name used in this context is not an array name, the behavior is undefined

A valid array *index* shall consist of one or more <comma>-separated expressions, similar to the way in which multi-dimensional arrays are indexed in some programming languages. Because *awk* arrays are really one-dimensional, such a <comma>-separated list shall be converted to a single string by concatenating the string values of the separate expressions, each separated from the other by the value of the **SUBSEP** variable. Thus, the following two index operations shall be equivalent:

```
var[expr1, expr2, ... exprn]
```

```
var[expr1 SUBSEP expr2 SUBSEP ... SUBSEP exprn]
```

The application shall ensure that a multi-dimensioned *index* used with the **in** operator is parenthesized. The **in** operator, which tests for the existence of a particular array element, shall not cause that element to exist. Any other reference to a nonexistent array element shall automatically create it.

Comparisons (with the '<', '<=', '!=', '==', '>', and '>=' operators) shall be made numerically if both operands are numeric, if one is numeric and the other has a string value that is a numeric string, or if one is numeric and the other has the uninitialized value. Otherwise, operands shall be converted to strings as required and a string comparison shall be made as follows:

- For the '!=' and '==' operators, the strings should be compared to check if they are identical but may be compared using the locale-specific collation sequence to check if they collate equally.
- For the other operators, the strings shall be compared using the locale-specific collation sequence.

The value of the comparison expression shall be 1 if the relation is true, or 0 if the relation is false.

## Variables and Special Variables

Variables can be used in an *awk* program by referencing them. With the exception of function parameters (see [User-Defined Functions](#), on page 2489), they are not explicitly declared. Function parameter names shall be local to the function; all other variable names shall be global. The same name shall not be used as both a function parameter name and as the name of a function or a special *awk* variable. The same name shall not be used both as a variable name with global scope and as the name of a function. The same name shall not be used within the same scope both as a scalar variable and as an array. Uninitialized variables, including scalar variables, array elements, and field variables, shall have an uninitialized value. An uninitialized value shall have both a numeric value of zero and a string value of the empty string. Evaluation of variables with an uninitialized value, to either string or numeric, shall be determined by the context in which they are used.

Field variables shall be designated by a '\$' followed by a number or numerical expression. The effect of the field number *expression* evaluating to anything other than a non-negative integer is unspecified; uninitialized variables or string values need not be converted to numeric values in this context. New field variables can be created by assigning a value to them. References to nonexistent fields (that is, fields after \$NF), shall evaluate to the uninitialized value. Such

references shall not create new fields. However, assigning to a nonexistent field (for example,  $\$(NF+2)=5$ ) shall increase the value of **NF**; create any intervening fields with the uninitialized value; and cause the value of **\$0** to be recomputed, with the fields being separated by the value of **OFS**. Each field variable shall have a string value or an uninitialized value when created. Field variables shall have the uninitialized value when created from **\$0** using **FS** and the variable does not contain any characters. If appropriate, the field variable shall be considered a numeric string (see [Expressions in awk](#), on page 2475).

Implementations shall support the following other special variables that are set by *awk*:

**ARGC** The number of elements in the **ARGV** array.

**ARGV** An array of command line arguments, excluding options and the *program* argument, numbered from zero to **ARGC**–1.

The arguments in **ARGV** can be modified or added to; **ARGC** can be altered. As each input file ends, *awk* shall treat the next non-null element of **ARGV**, up to the current value of **ARGC**–1, inclusive, as the name of the next input file. Thus, setting an element of **ARGV** to null means that it shall not be treated as an input file. The name ‘–’ indicates the standard input. If an argument matches the format of an *assignment* operand, this argument shall be treated as an *assignment* rather than a *file* argument.

**CONVFMT** The **printf** format for converting numbers to strings (except for output statements, where **OFMT** is used); “%.6g” by default.

**ENVIRON** An array representing the value of the environment, as described in the *exec* functions defined in the System Interfaces volume of POSIX.1-2008. The indices of the array shall be strings consisting of the names of the environment variables, and the value of each array element shall be a string consisting of the value of that variable. If appropriate, the environment variable shall be considered a *numeric string* (see [Expressions in awk](#), on page 2475); the array element shall also have its numeric value.

In all cases where the behavior of *awk* is affected by environment variables (including the environment of any commands that *awk* executes via the **system** function or via pipeline redirections with the **print** statement, the **printf** statement, or the **getline** function), the environment used shall be the environment at the time *awk* began executing; it is implementation-defined whether any modification of **ENVIRON** affects this environment.

**FILENAME** A pathname of the current input file. Inside a **BEGIN** action the value is undefined. Inside an **END** action the value shall be the name of the last input file processed.

**FNR** The ordinal number of the current record in the current file. Inside a **BEGIN** action the value shall be zero. Inside an **END** action the value shall be the number of the last record processed in the last file processed.

**FS** Input field separator regular expression; a <space> by default.

**NF** The number of fields in the current record. Inside a **BEGIN** action, the use of **NF** is undefined unless a **getline** function without a *var* argument is executed previously. Inside an **END** action, **NF** shall retain the value it had for the last record read, unless a subsequent, redirected, **getline** function without a *var* argument is performed prior to entering the **END** action.

|       |                |                                                                                                      |
|-------|----------------|------------------------------------------------------------------------------------------------------|
| 79752 | <b>NR</b>      | The ordinal number of the current record from the start of input. Inside a <b>BEGIN</b>              |
| 79753 |                | action the value shall be zero. Inside an <b>END</b> action the value shall be the number            |
| 79754 |                | of the last record processed.                                                                        |
| 79755 | <b>OFMT</b>    | The <b>printf</b> format for converting numbers to strings in output statements (see                 |
| 79756 |                | <a href="#">Output Statements</a> , on page 2485); "% . 6g" by default. The result of the conversion |
| 79757 |                | is unspecified if the value of <b>OFMT</b> is not a floating-point format specification.             |
| 79758 | <b>OFS</b>     | The <b>print</b> statement output field separator; <space> by default.                               |
| 79759 | <b>ORS</b>     | The <b>print</b> statement output record separator; a <newline> by default.                          |
| 79760 | <b>RLENGTH</b> | The length of the string matched by the <b>match</b> function.                                       |
| 79761 | <b>RS</b>      | The first character of the string value of <b>RS</b> shall be the input record separator; a          |
| 79762 |                | <newline> by default. If <b>RS</b> contains more than one character, the results are                 |
| 79763 |                | unspecified. If <b>RS</b> is null, then records are separated by sequences consisting of a           |
| 79764 |                | <newline> plus one or more blank lines, leading or trailing blank lines shall not                    |
| 79765 |                | result in empty records at the beginning or end of the input, and a <newline> shall                  |
| 79766 |                | always be a field separator, no matter what the value of <b>FS</b> is.                               |
| 79767 | <b>RSTART</b>  | The starting position of the string matched by the <b>match</b> function, numbering from             |
| 79768 |                | 1. This shall always be equivalent to the return value of the <b>match</b> function.                 |
| 79769 | <b>SUBSEP</b>  | The subscript separator string for multi-dimensional arrays; the default value is                    |
| 79770 |                | implementation-defined.                                                                              |

#### 79771 **Regular Expressions**

79772 The *awk* utility shall make use of the extended regular expression notation (see XBD [Section 9.4](#),  
 79773 on page 188) except that it shall allow the use of C-language conventions for escaping special  
 79774 characters within the EREs, as specified in the table in XBD [Chapter 5](#) (on page 121) ('\\',  
 79775 '\\a', '\\b', '\\f', '\\n', '\\r', '\\t', '\\v') and the following table; these escape sequences  
 79776 shall be recognized both inside and outside bracket expressions. Note that records need not be  
 79777 separated by <newline> characters and string constants can contain <newline> characters, so  
 79778 even the "\\n" sequence is valid in *awk* EREs. Using a <slash> character within an ERE requires  
 79779 the escaping shown in the following table.

Table 4-2 Escape Sequences in *awk*

| Escape Sequence | Description                                                                                                                                                                                                             | Meaning                                                                                                                                                                                                                         |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \ "             | <backslash> <quotation-mark>                                                                                                                                                                                            | <quotation-mark> character                                                                                                                                                                                                      |
| \ /             | <backslash> <slash>                                                                                                                                                                                                     | <slash> character                                                                                                                                                                                                               |
| \ ddd           | A <backslash> character followed by the longest sequence of one, two, or three octal-digit characters (01234567). If all of the digits are 0 (that is, representation of the NUL character), the behavior is undefined. | The character whose encoding is represented by the one, two, or three-digit octal integer. Multi-byte characters require multiple, concatenated escape sequences of this type, including the leading <backslash> for each byte. |
| \ c             | A <backslash> character followed by any character not described in this table or in the table in XBD Chapter 5 (on page 121) ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v').                                          | Undefined                                                                                                                                                                                                                       |

A regular expression can be matched against a specific field or string by using one of the two regular expression matching operators, '~' and '!~'. These operators shall interpret their right-hand operand as a regular expression and their left-hand operand as a string. If the regular expression matches the string, the '~' expression shall evaluate to a value of 1, and the '!~' expression shall evaluate to a value of 0. (The regular expression matching operation is as defined by the term matched in XBD Section 9.1 (on page 181), where a match occurs on any part of the string unless the regular expression is limited with the <circumflex> or <dollar-sign> special characters.) If the regular expression does not match the string, the '~' expression shall evaluate to a value of 0, and the '!~' expression shall evaluate to a value of 1. If the right-hand operand is any expression other than the lexical token **ERE**, the string value of the expression shall be interpreted as an extended regular expression, including the escape conventions described above. Note that these same escape conventions shall also be applied in determining the value of a string literal (the lexical token **STRING**), and thus shall be applied a second time when a string literal is used in this context.

When an **ERE** token appears as an expression in any context other than as the right-hand of the '~' or '!~' operator or as one of the built-in function arguments described below, the value of the resulting expression shall be the equivalent of:

```
$0 ~ /ere/
```

The *ere* argument to the **gsub**, **match**, **sub** functions, and the *fs* argument to the **split** function (see [String Functions](#), on page 2487) shall be interpreted as extended regular expressions. These can be either **ERE** tokens or arbitrary expressions, and shall be interpreted in the same manner as the right-hand side of the '~' or '!~' operator.

An extended regular expression can be used to separate fields by assigning a string containing the expression to the built-in variable **FS**, either directly or as a consequence of using the **-F sepstring** option. The default value of the **FS** variable shall be a single <space>. The following describes **FS** behavior:

1. If **FS** is a null string, the behavior is unspecified.



2. If **FS** is a single character:

- a. If **FS** is <space>, skip leading and trailing <blank> and <newline> characters; fields shall be delimited by sets of one or more <blank> or <newline> characters.
- b. Otherwise, if **FS** is any other character *c*, fields shall be delimited by each single occurrence of *c*.

3. Otherwise, the string value of **FS** shall be considered to be an extended regular expression. Each occurrence of a sequence matching the extended regular expression shall delimit fields.

Except for the '*~*' and '!~' operators, and in the **gsub**, **match**, **split**, and **sub** built-in functions, ERE matching shall be based on input records; that is, record separator characters (the first character of the value of the variable **RS**, <newline> by default) cannot be embedded in the expression, and no expression shall match the record separator character. If the record separator is not <newline>, <newline> characters embedded in the expression can be matched. For the '*~*' and '!~' operators, and in those four built-in functions, ERE matching shall be based on text strings; that is, any character (including <newline> and the record separator) can be embedded in the pattern, and an appropriate pattern shall match any character. However, in all *awk* ERE matching, the use of one or more NUL characters in the pattern, input record, or text string produces undefined results.

## Patterns

A *pattern* is any valid *expression*, a range specified by two expressions separated by a comma, or one of the two special patterns **BEGIN** or **END**.

## Special Patterns

The *awk* utility shall recognize two special patterns, **BEGIN** and **END**. Each **BEGIN** pattern shall be matched once and its associated action executed before the first record of input is read—except possibly by use of the **getline** function (see [Input/Output and General Functions](#), on page 2488) in a prior **BEGIN** action—and before command line assignment is done. Each **END** pattern shall be matched once and its associated action executed after the last record of input has been read. These two patterns shall have associated actions.

**BEGIN** and **END** shall not combine with other patterns. Multiple **BEGIN** and **END** patterns shall be allowed. The actions associated with the **BEGIN** patterns shall be executed in the order specified in the program, as are the **END** actions. An **END** pattern can precede a **BEGIN** pattern in a program.

If an *awk* program consists of only actions with the pattern **BEGIN**, and the **BEGIN** action contains no **getline** function, *awk* shall exit without reading its input when the last statement in the last **BEGIN** action is executed. If an *awk* program consists of only actions with the pattern **END** or only actions with the patterns **BEGIN** and **END**, the input shall be read before the statements in the **END** actions are executed.

## Expression Patterns

An expression pattern shall be evaluated as if it were an expression in a Boolean context. If the result is true, the pattern shall be considered to match, and the associated action (if any) shall be executed. If the result is false, the action shall not be executed.

## Pattern Ranges

A pattern range consists of two expressions separated by a comma; in this case, the action shall be performed for all records between a match of the first expression and the following match of the second expression, inclusive. At this point, the pattern range can be repeated starting at input records subsequent to the end of the matched range.

## Actions

An action is a sequence of statements as shown in the grammar in [Grammar](#) (on page 2490). Any single statement can be replaced by a statement list enclosed in curly braces. The application shall ensure that statements in a statement list are separated by <newline> or <semicolon> characters. Statements in a statement list shall be executed sequentially in the order that they appear.

The *expression* acting as the conditional in an **if** statement shall be evaluated and if it is non-zero or non-null, the following statement shall be executed; otherwise, if **else** is present, the statement following the **else** shall be executed.

The **if**, **while**, **do...while**, **for**, **break**, and **continue** statements are based on the ISO C standard (see [Section 1.1.2](#), on page 2321), except that the Boolean expressions shall be treated as described in [Expressions in awk](#) (on page 2475), and except in the case of:

```
for (variable in array)
```

which shall iterate, assigning each *index* of *array* to *variable* in an unspecified order. The results of adding new elements to *array* within such a **for** loop are undefined. If a **break** or **continue** statement occurs outside of a loop, the behavior is undefined.

The **delete** statement shall remove an individual array element. Thus, the following code deletes an entire array:

```
for (index in array)  
    delete array[index]
```

The **next** statement shall cause all further processing of the current input record to be abandoned. The behavior is undefined if a **next** statement appears or is invoked in a **BEGIN** or **END** action.

The **exit** statement shall invoke all **END** actions in the order in which they occur in the program source and then terminate the program without reading further input. An **exit** statement inside an **END** action shall terminate the program without further execution of **END** actions. If an expression is specified in an **exit** statement, its numeric value shall be the exit status of *awk*, unless subsequent errors are encountered or a subsequent **exit** statement with an expression is executed.



## Output Statements

Both **print** and **printf** statements shall write to standard output by default. The output shall be written to the location specified by *output\_redirection* if one is supplied, as follows:

```
> expression
>> expression
| expression
```

In all cases, the *expression* shall be evaluated to produce a string that is used as a pathname into which to write (for '*>*' or "*>>*") or as a command to be executed (for '*|*'). Using the first two forms, if the file of that name is not currently open, it shall be opened, creating it if necessary and using the first form, truncating the file. The output then shall be appended to the file. As long as the file remains open, subsequent calls in which *expression* evaluates to the same string value shall simply append output to the file. The file remains open until the **close** function (see [Input/Output and General Functions](#), on page 2488) is called with an expression that evaluates to the same string value.

The third form shall write output onto a stream piped to the input of a command. The stream shall be created if no stream is currently open with the value of *expression* as its command name. The stream created shall be equivalent to one created by a call to the *popen()* function defined in the System Interfaces volume of POSIX.1-2008 with the value of *expression* as the *command* argument and a value of *w* as the *mode* argument. As long as the stream remains open, subsequent calls in which *expression* evaluates to the same string value shall write output to the existing stream. The stream shall remain open until the **close** function (see [Input/Output and General Functions](#), on page 2488) is called with an expression that evaluates to the same string value. At that time, the stream shall be closed as if by a call to the *pclose()* function defined in the System Interfaces volume of POSIX.1-2008.

As described in detail by the grammar in [Grammar](#) (on page 2490), these output statements shall take a <comma>-separated list of *expressions* referred to in the grammar by the non-terminal symbols **expr\_list**, **print\_expr\_list**, or **print\_expr\_list\_opt**. This list is referred to here as the *expression list*, and each member is referred to as an *expression argument*.

The **print** statement shall write the value of each expression argument onto the indicated output stream separated by the current output field separator (see variable **OFS** above), and terminated by the output record separator (see variable **ORS** above). All expression arguments shall be taken as strings, being converted if necessary; this conversion shall be as described in [Expressions in awk](#) (on page 2475), with the exception that the **printf** format in **OFMT** shall be used instead of the value in **CONVFMT**. An empty expression list shall stand for the whole input record (\$0).

The **printf** statement shall produce output based on a notation similar to the File Format Notation used to describe file formats in this volume of POSIX.1-2008 (see XBD [Chapter 5](#), on page 121). Output shall be produced as specified with the first *expression* argument as the string *format* and subsequent *expression* arguments as the strings *arg1* to *argn*, inclusive, with the following exceptions:

1. The *format* shall be an actual character string rather than a graphical representation. Therefore, it cannot contain empty character positions. The <space> in the *format* string, in any context other than a *flag* of a conversion specification, shall be treated as an ordinary character that is copied to the output.
2. If the character set contains a 'Δ' character and that character appears in the *format* string, it shall be treated as an ordinary character that is copied to the output.

3. The *escape sequences* beginning with a <backslash> character shall be treated as sequences of ordinary characters that are copied to the output. Note that these same sequences shall be interpreted lexically by *awk* when they appear in literal strings, but they shall not be treated specially by the **printf** statement.
4. A *field width* or *precision* can be specified as the ' \* ' character instead of a digit string. In this case the next argument from the expression list shall be fetched and its numeric value taken as the field width or precision.
5. The implementation shall not precede or follow output from the d or u conversion specifier characters with <blank> characters not specified by the *format* string.
6. The implementation shall not precede output from the o conversion specifier character with leading zeros not specified by the *format* string.
7. For the c conversion specifier character: if the argument has a numeric value, the character whose encoding is that value shall be output. If the value is zero or is not the encoding of any character in the character set, the behavior is undefined. If the argument does not have a numeric value, the first character of the string value shall be output; if the string does not contain any characters, the behavior is undefined.
8. For each conversion specification that consumes an argument, the next expression argument shall be evaluated. With the exception of the c conversion specifier character, the value shall be converted (according to the rules specified in [Expressions in awk](#), on page 2475) to the appropriate type for the conversion specification.
9. If there are insufficient expression arguments to satisfy all the conversion specifications in the *format* string, the behavior is undefined.
10. If any character sequence in the *format* string begins with a ' % ' character, but does not form a valid conversion specification, the behavior is unspecified.

Both **print** and **printf** can output at least {LINE\_MAX} bytes.

## Functions

The *awk* language has a variety of built-in functions: arithmetic, string, input/output, and general.

### Arithmetic Functions

The arithmetic functions, except for **int**, shall be based on the ISO C standard (see [Section 1.1.2](#), on page 2321). The behavior is undefined in cases where the ISO C standard specifies that an error be returned or that the behavior is undefined. Although the grammar (see [Grammar](#), on page 2490) permits built-in functions to appear with no arguments or parentheses, unless the argument or parentheses are indicated as optional in the following list (by displaying them within the " [ ] " brackets), such use is undefined.

- |                                      |                                                                                  |
|--------------------------------------|----------------------------------------------------------------------------------|
| <b>atan2</b> ( <i>y</i> , <i>x</i> ) | Return arctangent of <i>y</i> / <i>x</i> in radians in the range $[-\pi, \pi]$ . |
| <b>cos</b> ( <i>x</i> )              | Return cosine of <i>x</i> , where <i>x</i> is in radians.                        |
| <b>sin</b> ( <i>x</i> )              | Return sine of <i>x</i> , where <i>x</i> is in radians.                          |
| <b>exp</b> ( <i>x</i> )              | Return the exponential function of <i>x</i> .                                    |
| <b>log</b> ( <i>x</i> )              | Return the natural logarithm of <i>x</i> .                                       |

79985        **sqrt**(*x*)        Return the square root of *x*.

79986        **int**(*x*)        Return the argument truncated to an integer. Truncation shall be toward 0 when  
79987                    *x*>0.

79988        **rand**()        Return a random number *n*, such that  $0 \leq n < 1$ .

79989        **srand**(*[expr]*) Set the seed value for *rand* to *expr* or use the time of day if *expr* is omitted. The  
79990                    previous seed value shall be returned.

## 79991        **String Functions**

79992        The string functions in the following list shall be supported. Although the grammar (see  
79993        [Grammar](#), on page 2490) permits built-in functions to appear with no arguments or parentheses,  
79994        unless the argument or parentheses are indicated as optional in the following list (by displaying  
79995        them within the "[ ]" brackets), such use is undefined.

79996        **gsub**(*ere, repl[, in]*)  
79997                    Behave like **sub** (see below), except that it shall replace all occurrences of the  
79998                    regular expression (like the *ed* utility global substitute) in *\$0* or in the *in* argument,  
79999                    when specified.

80000        **index**(*s, t*)        Return the position, in characters, numbering from 1, in string *s* where string *t* first  
80001                    occurs, or zero if it does not occur at all.

80002        **length**(*[(*s*)]*)        Return the length, in characters, of its argument taken as a string, or of the whole  
80003                    record, *\$0*, if there is no argument.

80004        **match**(*s, ere*)        Return the position, in characters, numbering from 1, in string *s* where the  
80005                    extended regular expression *ere* occurs, or zero if it does not occur at all. RSTART  
80006                    shall be set to the starting position (which is the same as the returned value), zero  
80007                    if no match is found; RLENGTH shall be set to the length of the matched string, -1  
80008                    if no match is found.

80009        **split**(*s, a[, fs ]*)  
80010                    Split the string *s* into array elements *a*[1], *a*[2], ..., *a*[*n*], and return *n*. All elements  
80011                    of the array shall be deleted before the split is performed. The separation shall be  
80012                    done with the ERE *fs* or with the field separator **FS** if *fs* is not given. Each array  
80013                    element shall have a string value when created and, if appropriate, the array  
80014                    element shall be considered a numeric string (see [Expressions in awk](#), on page  
80015                    2475). The effect of a null string as the value of *fs* is unspecified.

80016        **sprintf**(*fnt, expr, expr, ...*)  
80017                    Format the expressions according to the **printf** format given by *fnt* and return the  
80018                    resulting string.

80019        **sub**(*ere, repl[, in ]*)  
80020                    Substitute the string *repl* in place of the first instance of the extended regular  
80021                    expression *ERE* in string *in* and return the number of substitutions. An  
80022                    <ampersand> ('&') appearing in the string *repl* shall be replaced by the string  
80023                    from *in* that matches the ERE. An <ampersand> preceded with a <backslash> shall  
80024                    be interpreted as the literal <ampersand> character. An occurrence of two  
80025                    consecutive <backslash> characters shall be interpreted as just a single literal  
80026                    <backslash> character. Any other occurrence of a <backslash> (for example,  
80027                    preceding any other character) shall be treated as a literal <backslash> character.  
80028                    Note that if *repl* is a string literal (the lexical token **STRING**; see [Grammar](#), on page  
80029                    2490), the handling of the <ampersand> character occurs after any lexical

processing, including any lexical <backslash>-escape sequence processing. If *in* is specified and it is not an lvalue (see [Expressions in awk](#), on page 2475), the behavior is undefined. If *in* is omitted, *awk* shall use the current record (\$0) in its place.

**substr**(*s*, *m* [, *n* ])

Return the at most *n*-character substring of *s* that begins at position *m*, numbering from 1. If *n* is omitted, or if *n* specifies more characters than are left in the string, the length of the substring shall be limited by the length of the string *s*.

**tolower**(*s*) Return a string based on the string *s*. Each character in *s* that is an uppercase letter specified to have a **tolower** mapping by the *LC\_CTYPE* category of the current locale shall be replaced in the returned string by the lowercase letter specified by the mapping. Other characters in *s* shall be unchanged in the returned string.

**toupper**(*s*) Return a string based on the string *s*. Each character in *s* that is a lowercase letter specified to have a **toupper** mapping by the *LC\_CTYPE* category of the current locale is replaced in the returned string by the uppercase letter specified by the mapping. Other characters in *s* are unchanged in the returned string.

All of the preceding functions that take *ERE* as a parameter expect a pattern or a string valued expression that is a regular expression as defined in [Regular Expressions](#) (on page 2481).

## Input/Output and General Functions

The input/output and general functions are:

**close**(*expression*)

Close the file or pipe opened by a **print** or **printf** statement or a call to **getline** with the same string-valued *expression*. The limit on the number of open *expression* arguments is implementation-defined. If the close was successful, the function shall return zero; otherwise, it shall return non-zero.

*expression* | **getline** [*var*]

Read a record of input from a stream piped from the output of a command. The stream shall be created if no stream is currently open with the value of *expression* as its command name. The stream created shall be equivalent to one created by a call to the *popen*() function with the value of *expression* as the *command* argument and a value of *r* as the *mode* argument. As long as the stream remains open, subsequent calls in which *expression* evaluates to the same string value shall read subsequent records from the stream. The stream shall remain open until the **close** function is called with an expression that evaluates to the same string value. At that time, the stream shall be closed as if by a call to the *pclose*() function. If *var* is omitted, \$0 and **NF** shall be set; otherwise, *var* shall be set and, if appropriate, it shall be considered a numeric string (see [Expressions in awk](#), on page 2475).

The **getline** operator can form ambiguous constructs when there are unparenthesized operators (including concatenate) to the left of the ' | ' (to the beginning of the expression containing **getline**). In the context of the '\$' operator, ' | ' shall behave as if it had a lower precedence than '\$'. The result of evaluating other operators is unspecified, and conforming applications shall parenthesize properly all such usages.

**getline** Set \$0 to the next input record from the current input file. This form of **getline** shall set the **NF**, **NR**, and **FNR** variables.

**getline** *var* Set variable *var* to the next input record from the current input file and, if appropriate, *var* shall be considered a numeric string (see [Expressions in awk](#), on page 2475). This form of **getline** shall set the **FNR** and **NR** variables.

**getline** [*var*] < *expression*

Read the next record of input from a named file. The *expression* shall be evaluated to produce a string that is used as a pathname. If the file of that name is not currently open, it shall be opened. As long as the stream remains open, subsequent calls in which *expression* evaluates to the same string value shall read subsequent records from the file. The file shall remain open until the **close** function is called with an expression that evaluates to the same string value. If *var* is omitted, **\$0** and **NF** shall be set; otherwise, *var* shall be set and, if appropriate, it shall be considered a numeric string (see [Expressions in awk](#), on page 2475).

The **getline** operator can form ambiguous constructs when there are unparenthesized binary operators (including concatenate) to the right of the '**<**' (up to the end of the expression containing the **getline**). The result of evaluating such a construct is unspecified, and conforming applications shall parenthesize properly all such usages.

**system**(*expression*)

Execute the command given by *expression* in a manner equivalent to the *system()* function defined in the System Interfaces volume of POSIX.1-2008 and return the exit status of the command.

All forms of **getline** shall return 1 for successful input, zero for end-of-file, and -1 for an error.

Where strings are used as the name of a file or pipeline, the application shall ensure that the strings are textually identical. The terminology “same string value” implies that “equivalent strings”, even those that differ only by <space> characters, represent different files.

## User-Defined Functions

The *awk* language also provides user-defined functions. Such functions can be defined as:

```
function name([parameter, ...]) { statements }
```

A function can be referred to anywhere in an *awk* program; in particular, its use can precede its definition. The scope of a function is global.

Function parameters, if present, can be either scalars or arrays; the behavior is undefined if an array name is passed as a parameter that the function uses as a scalar, or if a scalar expression is passed as a parameter that the function uses as an array. Function parameters shall be passed by value if scalar and by reference if array name.

The number of parameters in the function definition need not match the number of parameters in the function call. Excess formal parameters can be used as local variables. If fewer arguments are supplied in a function call than are in the function definition, the extra parameters that are used in the function body as scalars shall evaluate to the uninitialized value until they are otherwise initialized, and the extra parameters that are used in the function body as arrays shall be treated as uninitialized arrays where each element evaluates to the uninitialized value until otherwise initialized.

When invoking a function, no white space can be placed between the function name and the opening parenthesis. Function calls can be nested and recursive calls can be made upon functions. Upon return from any nested or recursive function call, the values of all of the calling function's parameters shall be unchanged, except for array parameters passed by reference. The

**return** statement can be used to return a value. If a **return** statement appears outside of a function definition, the behavior is undefined.

In the function definition, <newline> characters shall be optional before the opening brace and after the closing brace. Function definitions can appear anywhere in the program where a *pattern-action* pair is allowed.

## Grammar

The grammar in this section and the lexical conventions in the following section shall together describe the syntax for *awk* programs. The general conventions for this style of grammar are described in [Section 1.3](#) (on page 2325). A valid program can be represented as the non-terminal symbol *program* in the grammar. This formal syntax shall take precedence over the preceding text syntax description.

```
%token NAME NUMBER STRING ERE
%token FUNC_NAME /* Name followed by '(' without white space. */

/* Keywords */
%token      Begin      End
/*          'BEGIN'    'END'                                */

%token      Break      Continue      Delete      Do      Else
/*          'break'    'continue'    'delete'    'do'    'else' */

%token      Exit      For      Function      If      In
/*          'exit'    'for'    'function'    'if'    'in'      */

%token      Next      Print      Printf      Return      While
/*          'next'    'print'    'printf'    'return'    'while' */

/* Reserved function names */
%token BUILTIN_FUNC_NAME
/* One token for the following:
 * atan2 cos sin exp log sqrt int rand srand
 * gsub index length match split sprintf sub
 * substr tolower toupper close system
 */
%token GETLINE
/* Syntactically different from other built-ins. */

/* Two-character tokens. */
%token ADD_ASSIGN SUB_ASSIGN MUL_ASSIGN DIV_ASSIGN MOD_ASSIGN POW_ASSIGN
/*      '+='      '-='      '*='      '/='      '%='      '^=' */

%token OR      AND      NO_MATCH      EQ      LE      GE      NE      INCR      DECR      APPEND
/*      '|'      '&&'    '!~'    '=='    '<='    '>='    '!='    '++'    '--'    '>>' */

/* One-character tokens. */
%token '{' '}' '(' ')' '[' ']' ',' ';' NEWLINE
%token '+' '-' '*' '%' '^' '!' '>' '<' '|' '?' ':' '~' '$' '='

%start program
%%

program          : item_list
                  | item_list item
                  ;
```



```

80164     item_list      : /* empty */
80165                     | item_list item terminator
80166                     ;

80167     item            : action
80168                     | pattern action
80169                     | normal_pattern
80170                     | Function NAME      '(' param_list_opt ')'
80171                       newline_opt action
80172                     | Function FUNC_NAME '(' param_list_opt ')'
80173                       newline_opt action
80174                     ;

80175     param_list_opt  : /* empty */
80176                     | param_list
80177                     ;

80178     param_list      : NAME
80179                     | param_list ',' NAME
80180                     ;

80181     pattern         : normal_pattern
80182                     | special_pattern
80183                     ;

80184     normal_pattern  : expr
80185                     | expr ',' newline_opt expr
80186                     ;

80187     special_pattern : Begin
80188                     | End
80189                     ;

80190     action          : '{' newline_opt
80191                     | '{' newline_opt terminated_statement_list
80192                     | '{' newline_opt unterminated_statement_list
80193                     ;

80194     terminator      : terminator NEWLINE
80195                     | ';'
80196                     | NEWLINE
80197                     ;

80198     terminated_statement_list : terminated_statement
80199                               | terminated_statement_list terminated_statement
80200                               ;

80201     unterminated_statement_list : unterminated_statement
80202                                  | terminated_statement_list unterminated_statement
80203                                  ;

80204     terminated_statement : action newline_opt
80205                           | If '(' expr ')' newline_opt terminated_statement
80206                           | If '(' expr ')' newline_opt terminated_statement
80207                             Else newline_opt terminated_statement
80208                           | While '(' expr ')' newline_opt terminated_statement
80209                           | For '(' simple_statement_opt ';'

```

```

80210             expr_opt ';' simple_statement_opt ')' newline_opt
80211             terminated_statement
80212         | For '(' NAME In NAME ')' newline_opt
80213             terminated_statement
80214         | ';' newline_opt
80215         | terminatable_statement NEWLINE newline_opt
80216         | terminatable_statement ';' newline_opt
80217         ;

80218     unterminated_statement : terminatable_statement
80219         | If '(' expr ')' newline_opt unterminated_statement
80220         | If '(' expr ')' newline_opt terminated_statement
80221         | Else newline_opt unterminated_statement
80222         | While '(' expr ')' newline_opt unterminated_statement
80223         | For '(' simple_statement_opt ';'
80224         |   expr_opt ';' simple_statement_opt ')' newline_opt
80225         |   unterminated_statement
80226         | For '(' NAME In NAME ')' newline_opt
80227         |   unterminated_statement
80228         ;

80229     terminatable_statement : simple_statement
80230         | Break
80231         | Continue
80232         | Next
80233         | Exit expr_opt
80234         | Return expr_opt
80235         | Do newline_opt terminated_statement While '(' expr ')'
80236         ;

80237     simple_statement_opt : /* empty */
80238         | simple_statement
80239         ;

80240     simple_statement : Delete NAME '[' expr_list ']'
80241         | expr
80242         | print_statement
80243         ;

80244     print_statement : simple_print_statement
80245         | simple_print_statement output_redirection
80246         ;

80247     simple_print_statement : Print print_expr_list_opt
80248         | Print '(' multiple_expr_list ')'
80249         | Printf print_expr_list
80250         | Printf '(' multiple_expr_list ')'
80251         ;

80252     output_redirection : '>' expr
80253         | APPEND expr
80254         | '|' expr
80255         ;

80256     expr_list_opt : /* empty */
80257         | expr_list

```



```

80258             ;
80259     expr_list      : expr
80260                     | multiple_expr_list
80261                     ;
80262     multiple_expr_list : expr ',' newline_opt expr
80263                       | multiple_expr_list ',' newline_opt expr
80264                       ;
80265     expr_opt       : /* empty */
80266                     | expr
80267                     ;
80268     expr           : unary_expr
80269                     | non_unary_expr
80270                     ;
80271     unary_expr     : '+' expr
80272                     | '-' expr
80273                     | unary_expr '^'      expr
80274                     | unary_expr '*'      expr
80275                     | unary_expr '/'      expr
80276                     | unary_expr '%'      expr
80277                     | unary_expr '+'      expr
80278                     | unary_expr '-'      expr
80279                     | unary_expr          non_unary_expr
80280                     | unary_expr '<'      expr
80281                     | unary_expr LE      expr
80282                     | unary_expr NE      expr
80283                     | unary_expr EQ      expr
80284                     | unary_expr '>'      expr
80285                     | unary_expr GE      expr
80286                     | unary_expr '~'      expr
80287                     | unary_expr NO_MATCH expr
80288                     | unary_expr In NAME
80289                     | unary_expr AND newline_opt expr
80290                     | unary_expr OR  newline_opt expr
80291                     | unary_expr '?' expr ':' expr
80292                     | unary_input_function
80293                     ;
80294     non_unary_expr : '(' expr ')'
80295                     | '!' expr
80296                     | non_unary_expr '^'      expr
80297                     | non_unary_expr '*'      expr
80298                     | non_unary_expr '/'      expr
80299                     | non_unary_expr '%'      expr
80300                     | non_unary_expr '+'      expr
80301                     | non_unary_expr '-'      expr
80302                     | non_unary_expr          non_unary_expr
80303                     | non_unary_expr '<'      expr
80304                     | non_unary_expr LE      expr
80305                     | non_unary_expr NE      expr
80306                     | non_unary_expr EQ      expr

```

```

80307         | non_unary_expr '>'      expr
80308         | non_unary_expr GE      expr
80309         | non_unary_expr '~'    expr
80310         | non_unary_expr NO_MATCH expr
80311         | non_unary_expr In NAME
80312         | '(' multiple_expr_list ')' In NAME
80313         | non_unary_expr AND newline_opt expr
80314         | non_unary_expr OR  newline_opt expr
80315         | non_unary_expr '?' expr ':' expr
80316         | NUMBER
80317         | STRING
80318         | lvalue
80319         | ERE
80320         | lvalue INCR
80321         | lvalue DECR
80322         | INCR lvalue
80323         | DECR lvalue
80324         | lvalue POW_ASSIGN expr
80325         | lvalue MOD_ASSIGN expr
80326         | lvalue MUL_ASSIGN expr
80327         | lvalue DIV_ASSIGN expr
80328         | lvalue ADD_ASSIGN expr
80329         | lvalue SUB_ASSIGN expr
80330         | lvalue '=' expr
80331         | FUNC_NAME '(' expr_list_opt ')'
80332         | /* no white space allowed before '(' */
80333         | BUILTIN_FUNC_NAME '(' expr_list_opt ')'
80334         | BUILTIN_FUNC_NAME
80335         | non_unary_input_function
80336         ;

80337 print_expr_list_opt : /* empty */
80338         | print_expr_list
80339         ;

80340 print_expr_list : print_expr
80341         | print_expr_list ',' newline_opt print_expr
80342         ;

80343 print_expr : unary_print_expr
80344         | non_unary_print_expr
80345         ;

80346 unary_print_expr : '+' print_expr
80347         | '-' print_expr
80348         | unary_print_expr '^'      print_expr
80349         | unary_print_expr '*'      print_expr
80350         | unary_print_expr '/'      print_expr
80351         | unary_print_expr '%'      print_expr
80352         | unary_print_expr '+'      print_expr
80353         | unary_print_expr '-'      print_expr
80354         | unary_print_expr          non_unary_print_expr
80355         | unary_print_expr '~'      print_expr
80356         | unary_print_expr NO_MATCH print_expr

```

```

80357         unary_print_expr In NAME
80358         unary_print_expr AND newline_opt print_expr
80359         unary_print_expr OR  newline_opt print_expr
80360         unary_print_expr '?' print_expr ':' print_expr
80361     ;

80362     non_unary_print_expr : '(' expr ')'
80363         | '!' print_expr
80364         | non_unary_print_expr '^'      print_expr
80365         | non_unary_print_expr '*'      print_expr
80366         | non_unary_print_expr '/'      print_expr
80367         | non_unary_print_expr '%'      print_expr
80368         | non_unary_print_expr '+'      print_expr
80369         | non_unary_print_expr '-'      print_expr
80370         | non_unary_print_expr          non_unary_print_expr
80371         | non_unary_print_expr '~'      print_expr
80372         | non_unary_print_expr NO_MATCH print_expr
80373         | non_unary_print_expr In NAME
80374         | '(' multiple_expr_list ')' In NAME
80375         | non_unary_print_expr AND newline_opt print_expr
80376         | non_unary_print_expr OR  newline_opt print_expr
80377         | non_unary_print_expr '?' print_expr ':' print_expr
80378         | NUMBER
80379         | STRING
80380         | lvalue
80381         | ERE
80382         | lvalue INCR
80383         | lvalue DECR
80384         | INCR lvalue
80385         | DECR lvalue
80386         | lvalue POW_ASSIGN print_expr
80387         | lvalue MOD_ASSIGN print_expr
80388         | lvalue MUL_ASSIGN print_expr
80389         | lvalue DIV_ASSIGN print_expr
80390         | lvalue ADD_ASSIGN print_expr
80391         | lvalue SUB_ASSIGN print_expr
80392         | lvalue '=' print_expr
80393         | FUNC_NAME '(' expr_list_opt ')'
80394             /* no white space allowed before '(' */
80395         | BUILTIN_FUNC_NAME '(' expr_list_opt ')'
80396         | BUILTIN_FUNC_NAME
80397     ;

80398     lvalue
80399         : NAME
80400         | NAME '[' expr_list ']'
80401         | '$' expr
80402     ;

80402     non_unary_input_function : simple_get
80403         | simple_get '<' expr
80404         | non_unary_expr '|' simple_get
80405     ;

80406     unary_input_function : unary_expr '|' simple_get

```

```

80407             ;
80408 simple_get      : GETLINE
80409                 | GETLINE lvalue
80410             ;
80411 newline_opt     : /* empty */
80412                 | newline_opt NEWLINE
80413             ;

```

This grammar has several ambiguities that shall be resolved as follows:

- Operator precedence and associativity shall be as described in [Table 4-1](#) (on page 2475).
- In case of ambiguity, an **else** shall be associated with the most immediately preceding **if** that would satisfy the grammar.
- In some contexts, a `<slash>` (`'/'`) that is used to surround an ERE could also be the division operator. This shall be resolved in such a way that wherever the division operator could appear, a `<slash>` is assumed to be the division operator. (There is no unary division operator.)

Each expression in an *awk* program shall conform to the precedence and associativity rules, even when this is not needed to resolve an ambiguity. For example, because `'$'` has higher precedence than `'++'`, the string `"$x++--"` is not a valid *awk* expression, even though it is unambiguously parsed by the grammar as `"$(x++)--"`.

One convention that might not be obvious from the formal grammar is where `<newline>` characters are acceptable. There are several obvious placements such as terminating a statement, and a `<backslash>` can be used to escape `<newline>` characters between any lexical tokens. In addition, `<newline>` characters without `<backslash>` characters can follow a comma, an open brace, logical AND operator (`"&&"`), logical OR operator (`"||"`), the **do** keyword, the **else** keyword, and the closing parenthesis of an **if**, **for**, or **while** statement. For example:

```

80432 { print $1,
80433       $2 }

```

## Lexical Conventions

The lexical conventions for *awk* programs, with respect to the preceding grammar, shall be as follows:

1. Except as noted, *awk* shall recognize the longest possible token or delimiter beginning at a given point.
2. A comment shall consist of any characters beginning with the `<number-sign>` character and terminated by, but excluding the next occurrence of, a `<newline>`. Comments shall have no effect, except to delimit lexical tokens.
3. The `<newline>` shall be recognized as the token **NEWLINE**.
4. A `<backslash>` character immediately followed by a `<newline>` shall have no effect.
5. The token **STRING** shall represent a string constant. A string constant shall begin with the character `'`. Within a string constant, a `<backslash>` character shall be considered to begin an escape sequence as specified in the table in [XBD Chapter 5](#) (on page 121) (`'\\'`, `'\a'`, `'\b'`, `'\f'`, `'\n'`, `'\r'`, `'\t'`, `'\v'`). In addition, the escape sequences in [Table 4-2](#) (on page 2482) shall be recognized. A `<newline>` shall not occur within a string constant. A string constant shall be terminated by the first unescaped occurrence of

the character ' ' ' after the one that begins the string constant. The value of the string shall be the sequence of all unescaped characters and values of escape sequences between, but not including, the two delimiting ' ' ' characters.

6. The token **ERE** represents an extended regular expression constant. An ERE constant shall begin with the <slash> character. Within an ERE constant, a <backslash> character shall be considered to begin an escape sequence as specified in the table in XBD Chapter 5 (on page 121). In addition, the escape sequences in Table 4-2 (on page 2482) shall be recognized. The application shall ensure that a <newline> does not occur within an ERE constant. An ERE constant shall be terminated by the first unescaped occurrence of the <slash> character after the one that begins the ERE constant. The extended regular expression represented by the ERE constant shall be the sequence of all unescaped characters and values of escape sequences between, but not including, the two delimiting <slash> characters.

7. A <blank> shall have no effect, except to delimit lexical tokens or within **STRING** or **ERE** tokens.

8. The token **NUMBER** shall represent a numeric constant. Its form and numeric value shall either be equivalent to the **decimal-floating-constant** token as specified by the ISO C standard, or it shall be a sequence of decimal digits and shall be evaluated as an integer constant in decimal. In addition, implementations may accept numeric constants with the form and numeric value equivalent to the **hexadecimal-constant** and **hexadecimal-floating-constant** tokens as specified by the ISO C standard.

If the value is too large or too small to be representable (see Section 1.1.2, on page 2321), the behavior is undefined.

9. A sequence of underscores, digits, and alphabetic characters from the portable character set (see XBD Section 6.1, on page 125), beginning with an <underscore> or alphabetic character, shall be considered a word.

10. The following words are keywords that shall be recognized as individual tokens; the name of the token is the same as the keyword:

|                 |               |             |                 |              |               |
|-----------------|---------------|-------------|-----------------|--------------|---------------|
| <b>BEGIN</b>    | <b>delete</b> | <b>END</b>  | <b>function</b> | <b>in</b>    | <b>printf</b> |
| <b>break</b>    | <b>do</b>     | <b>exit</b> | <b>getline</b>  | <b>next</b>  | <b>return</b> |
| <b>continue</b> | <b>else</b>   | <b>for</b>  | <b>if</b>       | <b>print</b> | <b>while</b>  |

11. The following words are names of built-in functions and shall be recognized as the token **BUILTIN\_FUNC\_NAME**:

|              |               |              |                |                |                |
|--------------|---------------|--------------|----------------|----------------|----------------|
| <b>atan2</b> | <b>gsub</b>   | <b>log</b>   | <b>split</b>   | <b>sub</b>     | <b>toupper</b> |
| <b>close</b> | <b>index</b>  | <b>match</b> | <b>sprintf</b> | <b>substr</b>  |                |
| <b>cos</b>   | <b>int</b>    | <b>rand</b>  | <b>sqrt</b>    | <b>system</b>  |                |
| <b>exp</b>   | <b>length</b> | <b>sin</b>   | <b>srand</b>   | <b>tolower</b> |                |

The above-listed keywords and names of built-in functions are considered reserved words.

12. The token **NAME** shall consist of a word that is not a keyword or a name of a built-in function and is not followed immediately (without any delimiters) by the ' ( ' character.
13. The token **FUNC\_NAME** shall consist of a word that is not a keyword or a name of a built-in function, followed immediately (without any delimiters) by the ' ( ' character. The ' ( ' character shall not be included as part of the token.

14. The following two-character sequences shall be recognized as the named tokens:

| Token Name | Sequence | Token Name | Sequence |
|------------|----------|------------|----------|
| ADD_ASSIGN | +=       | NO_MATCH   | !~       |
| SUB_ASSIGN | --       | EQ         | ==       |
| MUL_ASSIGN | *=       | LE         | <=       |
| DIV_ASSIGN | /=       | GE         | >=       |
| MOD_ASSIGN | %=       | NE         | !=       |
| POW_ASSIGN | ^=       | INCR       | ++       |
| OR         |          | DECR       | --       |
| AND        | &&       | APPEND     | >>       |

15. The following single characters shall be recognized as tokens whose names are the character:

<newline> { } ( ) [ ] , ; + - \* % ^ ! > < | ? : ~ \$ =

There is a lexical ambiguity between the token **ERE** and the tokens **'/'** and **DIV\_ASSIGN**. When an input sequence begins with a <slash> character in any syntactic context where the token **'/'** or **DIV\_ASSIGN** could appear as the next token in a valid program, the longer of those two tokens that can be recognized shall be recognized. In any other syntactic context where the token **ERE** could appear as the next token in a valid program, the token **ERE** shall be recognized.

## EXIT STATUS

The following exit values shall be returned:

0 All input files were processed successfully.

>0 An error occurred.

The exit status can be altered within the program by using an **exit** expression.

## CONSEQUENCES OF ERRORS

If any *file* operand is specified and the named file cannot be accessed, *awk* shall write a diagnostic message to standard error and terminate without any further action.

If the program specified by either the *program* operand or a *progfile* operand is not a valid *awk* program (as specified in the EXTENDED DESCRIPTION section), the behavior is undefined.

## APPLICATION USAGE

The **index**, **length**, **match**, and **substr** functions should not be confused with similar functions in the ISO C standard; the *awk* versions deal with characters, while the ISO C standard deals with bytes.

Because the concatenation operation is represented by adjacent expressions rather than an explicit operator, it is often necessary to use parentheses to enforce the proper evaluation precedence.

When using *awk* to process pathnames, it is recommended that **LC\_ALL**, or at least **LC\_CTYPE** and **LC\_COLLATE**, are set to **POSIX** or **C** in the environment, since pathnames can contain byte sequences that do not form valid characters in some locales, in which case the utility's behavior would be undefined. In the **POSIX** locale each byte is a valid single-byte character, and therefore this problem is avoided.

On implementations where the **"=="** operator checks if strings collate equally, applications needing to check whether strings are identical can use:

```
length(a) == length(b) && index(a,b) == 1
```

On implementations where the "==" operator checks if strings are identical, applications needing to check whether strings collate equally can use:

```
a <= b && a >= b
```

## EXAMPLES

The *awk* program specified in the command line is most easily specified within single-quotes (for example, '*program*') for applications using *sh*, because *awk* programs commonly contain characters that are special to the shell, including double-quotes. In the cases where an *awk* program contains single-quote characters, it is usually easiest to specify most of the program as strings within single-quotes concatenated by the shell with quoted single-quote characters. For example:

```
awk '/'\''/ { print "quote:", $0 }'
```

prints all lines from the standard input containing a single-quote character, prefixed with *quote*..

The following are examples of simple *awk* programs:

1. Write to the standard output all input lines for which field 3 is greater than 5:

```
$3 > 5
```

2. Write every tenth line:

```
(NR % 10) == 0
```

3. Write any line with a substring matching the regular expression:

```
/(G|D)(2[0-9][[:alpha:]]*)/
```

4. Print any line with a substring containing a 'G' or 'D', followed by a sequence of digits and characters. This example uses character classes **digit** and **alpha** to match language-independent digit and alphabetic characters respectively:

```
/(G|D)([[:digit:]][[:alpha:]]*)/
```

5. Write any line in which the second field matches the regular expression and the fourth field does not:

```
$2 ~ /xyz/ && $4 !~ /xyz/
```

6. Write any line in which the second field contains a <backslash>:

```
$2 ~ /\\"/>

```

7. Write any line in which the second field contains a <backslash>. Note that <backslash>-escapes are interpreted twice; once in lexical processing of the string and once in processing the regular expression:

```
$2 ~ "\\\"/>

```

8. Write the second to the last and the last field in each line. Separate the fields by a <colon>:

```
{OFS=":";print $(NF-1), $NF}
```

9. Write the line number and number of fields in each line. The three strings representing the line number, the <colon>, and the number of fields are concatenated and that string is written to standard output:

```
{print NR ":" NF}
```

10. Write lines longer than 72 characters:
- ```
length($0) > 72
```
11. Write the first two fields in opposite order separated by **OFS**:
- ```
{ print $2, $1 }
```
12. Same, with input fields separated by a <comma> or <space> and <tab> characters, or both:
- ```
BEGIN { FS = ",[ \t]*|[ \t]+" }
{ print $2, $1 }
```
13. Add up the first column, print sum, and average:
- ```
{ s += $1 }
END { print "sum is ", s, " average is", s/NR }
```
14. Write fields in reverse order, one per line (many lines out for each line in):
- ```
{ for (i = NF; i > 0; --i) print $i }
```
15. Write all lines between occurrences of the strings **start** and **stop**:
- ```
/start/, /stop/
```
16. Write all lines whose first field is different from the previous one:
- ```
$1 != prev { print; prev = $1 }
```
17. Simulate *echo*:
- ```
BEGIN {
    for (i = 1; i < ARGV; ++i)
        printf("%s%s", ARGV[i], i==ARGC-1?"\n":" ")
}
```
18. Write the path prefixes contained in the *PATH* environment variable, one per line:
- ```
BEGIN {
    n = split (ENVIRON["PATH"], path, ":")
    for (i = 1; i <= n; ++i)
        print path[i]
}
```
19. If there is a file named **input** containing page headers of the form:
- Page #
- and a file named **program** that contains:
- ```
/Page/    { $2 = n++; }
          { print }
```
- then the command line:
- ```
awk -f program n=5 input
```
- prints the file **input**, filling in page numbers starting at 5.



## RATIONALE

This description is based on the new *awk*, “nawk”, (see the referenced *The AWK Programming Language*), which introduced a number of new features to the historical *awk*:

1. New keywords: **delete**, **do**, **function**, **return**
2. New built-in functions: **atan2**, **close**, **cos**, **gsub**, **match**, **rand**, **sin**, **srand**, **sub**, **system**
3. New predefined variables: **FNR**, **ARGC**, **ARGV**, **RSTART**, **RLENGTH**, **SUBSEP**
4. New expression operators: **?**, **:**, **..**, **^**
5. The **FS** variable and the third argument to **split**, now treated as extended regular expressions.
6. The operator precedence, changed to more closely match the C language. Two examples of code that operate differently are:

```
while ( n /= 10 > 1 ) ...
if ( !"wk" ~ /bwk/ ) ...
```

Several features have been added based on newer implementations of *awk*:

- Multiple instances of **-f profile** are permitted.
- The new option **-v assignment**.
- The new predefined variable **ENVIRON**.
- New built-in functions **toupper** and **tolower**.
- More formatting capabilities are added to **printf** to match the ISO C standard.

Earlier versions of this standard required implementations to support multiple adjacent `<semicolon>`s, lines with one or more `<semicolon>` before a rule (*pattern-action* pairs), and lines with only `<semicolon>`(s). These are not required by this standard and are considered poor programming practice, but can be accepted by an implementation of *awk* as an extension.

The overall *awk* syntax has always been based on the C language, with a few features from the shell command language and other sources. Because of this, it is not completely compatible with any other language, which has caused confusion for some users. It is not the intent of the standard developers to address such issues. A few relatively minor changes toward making the language more compatible with the ISO C standard were made; most of these changes are based on similar changes in recent implementations, as described above. There remain several C-language conventions that are not in *awk*. One of the notable ones is the `<comma>` operator, which is commonly used to specify multiple expressions in the C language **for** statement. Also, there are various places where *awk* is more restrictive than the C language regarding the type of expression that can be used in a given context. These limitations are due to the different features that the *awk* language does provide.

Regular expressions in *awk* have been extended somewhat from historical implementations to make them a pure superset of extended regular expressions, as defined by POSIX.1-2008 (see XBD Section 9.4, on page 188). The main extensions are internationalization features and interval expressions. Historical implementations of *awk* have long supported `<backslash>`-escape sequences as an extension to extended regular expressions, and this extension has been retained despite inconsistency with other utilities. The number of escape sequences recognized in both extended regular expressions and strings has varied (generally increasing with time) among implementations. The set specified by POSIX.1-2008 includes most sequences known to be supported by popular implementations and by the ISO C standard. One sequence that is not supported is hexadecimal value escapes beginning with `'\x'`. This would

allow values expressed in more than 9 bits to be used within *awk* as in the ISO C standard. However, because this syntax has a non-deterministic length, it does not permit the subsequent character to be a hexadecimal digit. This limitation can be dealt with in the C language by the use of lexical string concatenation. In the *awk* language, concatenation could also be a solution for strings, but not for extended regular expressions (either lexical ERE tokens or strings used dynamically as regular expressions). Because of this limitation, the feature has not been added to POSIX.1-2008.

When a string variable is used in a context where an extended regular expression normally appears (where the lexical token ERE is used in the grammar) the string does not contain the literal `<slash>` characters.

Some versions of *awk* allow the form:

```
func name(args, ... ) { statements }
```

This has been deprecated by the authors of the language, who asked that it not be specified.

Historical implementations of *awk* produce an error if a **next** statement is executed in a **BEGIN** action, and cause *awk* to terminate if a **next** statement is executed in an **END** action. This behavior has not been documented, and it was not believed that it was necessary to standardize it.

The specification of conversions between string and numeric values is much more detailed than in the documentation of historical implementations or in the referenced *The AWK Programming Language*. Although most of the behavior is designed to be intuitive, the details are necessary to ensure compatible behavior from different implementations. This is especially important in relational expressions since the types of the operands determine whether a string or numeric comparison is performed. From the perspective of an application developer, it is usually sufficient to expect intuitive behavior and to force conversions (by adding zero or concatenating a null string) when the type of an expression does not obviously match what is needed. The intent has been to specify historical practice in almost all cases. The one exception is that, in historical implementations, variables and constants maintain both string and numeric values after their original value is converted by any use. This means that referencing a variable or constant can have unexpected side-effects. For example, with historical implementations the following program:

```
{
    a = "+2"
    b = 2
    if (NR % 2)
        c = a + b
    if (a == b)
        print "numeric comparison"
    else
        print "string comparison"
}
```

would perform a numeric comparison (and output numeric comparison) for each odd-numbered line, but perform a string comparison (and output string comparison) for each even-numbered line. POSIX.1-2008 ensures that comparisons will be numeric if necessary. With historical implementations, the following program:

```
BEGIN {
    OFMT = "%e"
    print 3.14
```

```

80703         OFMT = "%f"
80704         print 3.14
80705     }

```

80706 would output "3.140000e+00" twice, because in the second **print** statement the constant  
 80707 "3.14" would have a string value from the previous conversion. POSIX.1-2008 requires that the  
 80708 output of the second **print** statement be "3.140000". The behavior of historical  
 80709 implementations was seen as too unintuitive and unpredictable.

80710 It was pointed out that with the rules contained in early drafts, the following script would print  
 80711 nothing:

```

80712 BEGIN {
80713     y[1.5] = 1
80714     OFMT = "%e"
80715     print y[1.5]
80716 }

```

80717 Therefore, a new variable, **CONVFMT**, was introduced. The **OFMT** variable is now restricted to  
 80718 affecting output conversions of numbers to strings and **CONVFMT** is used for internal  
 80719 conversions, such as comparisons or array indexing. The default value is the same as that for  
 80720 **OFMT**, so unless a program changes **CONVFMT** (which no historical program would do), it  
 80721 will receive the historical behavior associated with internal string conversions.

80722 The POSIX *awk* lexical and syntactic conventions are specified more formally than in other  
 80723 sources. Again the intent has been to specify historical practice. One convention that may not be  
 80724 obvious from the formal grammar as in other verbal descriptions is where <newline> characters  
 80725 are acceptable. There are several obvious placements such as terminating a statement, and a  
 80726 <backslash> can be used to escape <newline> characters between any lexical tokens. In addition,  
 80727 <newline> characters without <backslash> characters can follow a comma, an open brace, a  
 80728 logical AND operator ("&&"), a logical OR operator (" || "), the **do** keyword, the **else** keyword,  
 80729 and the closing parenthesis of an **if**, **for**, or **while** statement. For example:

```

80730 { print $1,
80731     $2 }

```

80732 The requirement that *awk* add a trailing <newline> to the program argument text is to simplify  
 80733 the grammar, making it match a text file in form. There is no way for an application or test suite  
 80734 to determine whether a literal <newline> is added or whether *awk* simply acts as if it did.

80735 POSIX.1-2008 requires several changes from historical implementations in order to support  
 80736 internationalization. Probably the most subtle of these is the use of the decimal-point character,  
 80737 defined by the *LC\_NUMERIC* category of the locale, in representations of floating-point  
 80738 numbers. This locale-specific character is used in recognizing numeric input, in converting  
 80739 between strings and numeric values, and in formatting output. However, regardless of locale,  
 80740 the <period> character (the decimal-point character of the POSIX locale) is the decimal-point  
 80741 character recognized in processing *awk* programs (including assignments in command line  
 80742 arguments). This is essentially the same convention as the one used in the ISO C standard. The  
 80743 difference is that the C language includes the *setlocale()* function, which permits an application  
 80744 to modify its locale. Because of this capability, a C application begins executing with its locale set  
 80745 to the C locale, and only executes in the environment-specified locale after an explicit call to  
 80746 *setlocale()*. However, adding such an elaborate new feature to the *awk* language was seen as  
 80747 inappropriate for POSIX.1-2008. It is possible to execute an *awk* program explicitly in any desired  
 80748 locale by setting the environment in the shell.

80749 The undefined behavior resulting from NULs in extended regular expressions allows future

extensions for the GNU *gawk* program to process binary data.

The behavior in the case of invalid *awk* programs (including lexical, syntactic, and semantic errors) is undefined because it was considered overly limiting on implementations to specify. In most cases such errors can be expected to produce a diagnostic and a non-zero exit status. However, some implementations may choose to extend the language in ways that make use of certain invalid constructs. Other invalid constructs might be deemed worthy of a warning, but otherwise cause some reasonable behavior. Still other constructs may be very difficult to detect in some implementations. Also, different implementations might detect a given error during an initial parsing of the program (before reading any input files) while others might detect it when executing the program after reading some input. Implementors should be aware that diagnosing errors as early as possible and producing useful diagnostics can ease debugging of applications, and thus make an implementation more usable.

The unspecified behavior from using multi-character **RS** values is to allow possible future extensions based on extended regular expressions used for record separators. Historical implementations take the first character of the string and ignore the others.

Unspecified behavior when *split(string,array,<null>)* is used is to allow a proposed future extension that would split up a string into an array of individual characters.

In the context of the **getline** function, equally good arguments for different precedences of the **|** and **<** operators can be made. Historical practice has been that:

```
getline < "a" "b"
```

is parsed as:

```
( getline < "a" ) "b"
```

although many would argue that the intent was that the file **ab** should be read. However:

```
getline < "x" + 1
```

parses as:

```
getline < ( "x" + 1 )
```

Similar problems occur with the **|** version of **getline**, particularly in combination with **\$**. For example:

```
$"echo hi" | getline
```

(This situation is particularly problematic when used in a **print** statement, where the **|getline** part might be a redirection of the **print**.)

Since in most cases such constructs are not (or at least should not) be used (because they have a natural ambiguity for which there is no conventional parsing), the meaning of these constructs has been made explicitly unspecified. (The effect is that a conforming application that runs into the problem must parenthesize to resolve the ambiguity.) There appeared to be few if any actual uses of such constructs.

Grammars can be written that would cause an error under these circumstances. Where backwards-compatibility is not a large consideration, implementors may wish to use such grammars.

Some historical implementations have allowed some built-in functions to be called without an argument list, the result being a default argument list chosen in some “reasonable” way. Use of **length** as a synonym for **length(\$0)** is the only one of these forms that is thought to be widely known or widely used; this particular form is documented in various places (for example, most

historical *awk* reference pages, although not in the referenced *The AWK Programming Language*) as legitimate practice. With this exception, default argument lists have always been undocumented and vaguely defined, and it is not at all clear how (or if) they should be generalized to user-defined functions. They add no useful functionality and preclude possible future extensions that might need to name functions without calling them. Not standardizing them seems the simplest course. The standard developers considered that **length** merited special treatment, however, since it has been documented in the past and sees possibly substantial use in historical programs. Accordingly, this usage has been made legitimate, but Issue 5 removed the obsolescent marking for XSI-conforming implementations and many otherwise conforming applications depend on this feature.

In **sub** and **gsub**, if *repl* is a string literal (the lexical token **STRING**), then two consecutive <backslash> characters should be used in the string to ensure a single <backslash> will precede the <ampersand> when the resultant string is passed to the function. (For example, to specify one literal <ampersand> in the replacement string, use **gsub(ERE, "\\&")**.)

Historically, the only special character in the *repl* argument of **sub** and **gsub** string functions was the <ampersand> ('&') character and preceding it with the <backslash> character was used to turn off its special meaning.

The description in the ISO POSIX-2:1993 standard introduced behavior such that the <backslash> character was another special character and it was unspecified whether there were any other special characters. This description introduced several portability problems, some of which are described below, and so it has been replaced with the more historical description. Some of the problems include:

- Historically, to create the replacement string, a script could use **gsub(ERE, "\\&")**, but with the ISO POSIX-2:1993 standard wording, it was necessary to use **gsub(ERE, "\\&&")**. The <backslash> characters are doubled here because all string literals are subject to lexical analysis, which would reduce each pair of <backslash> characters to a single <backslash> before being passed to **gsub**.
- Since it was unspecified what the special characters were, for portable scripts to guarantee that characters are printed literally, each character had to be preceded with a <backslash>. (For example, a portable script had to use **gsub(ERE, "\\h\\i")** to produce a replacement string of "hi".)

The description for comparisons in the ISO POSIX-2:1993 standard did not properly describe historical practice because of the way numeric strings are compared as numbers. The current rules cause the following code:

```
if (0 == "000")
    print "strange, but true"
else
    print "not true"
```

to do a numeric comparison, causing the **if** to succeed. It should be intuitively obvious that this is incorrect behavior, and indeed, no historical implementation of *awk* actually behaves this way.

To fix this problem, the definition of *numeric string* was enhanced to include only those values obtained from specific circumstances (mostly external sources) where it is not possible to determine unambiguously whether the value is intended to be a string or a numeric.

Variables that are assigned to a numeric string shall also be treated as a numeric string. (For example, the notion of a numeric string can be propagated across assignments.) In comparisons, all variables having the uninitialized value are to be treated as a numeric operand evaluating to the numeric value zero.



Uninitialized variables include all types of variables including scalars, array elements, and fields. The definition of an uninitialized value in [Variables and Special Variables](#) (on page 2479) is necessary to describe the value placed on uninitialized variables and on fields that are valid (for example, `< $NF`) but have no characters in them and to describe how these variables are to be used in comparisons. A valid field, such as `$1`, that has no characters in it can be obtained from an input line of `"\t\t"` when `FS='\t'`. Historically, the comparison (`$1<10`) was done numerically after evaluating `$1` to the value zero.

The phrase "... also shall have the numeric value of the numeric string" was removed from several sections of the ISO POSIX-2:1993 standard because it specifies an unnecessary implementation detail. It is not necessary for POSIX.1-2008 to specify that these objects be assigned two different values. It is only necessary to specify that these objects may evaluate to two different values depending on context.

Historical implementations of *awk* did not parse hexadecimal integer or floating constants like `"0xa"` and `"0xap0"`. Due to an oversight, the 2001 through 2004 editions of this standard required support for hexadecimal floating constants. This was due to the reference to `atof()`. This version of the standard allows but does not require implementations to use `atof()` and includes a description of how floating-point numbers are recognized as an alternative to match historic behavior. The intent of this change is to allow implementations to recognize floating-point constants according to either the ISO/IEC 9899:1990 standard or ISO/IEC 9899:1999 standard, and to allow (but not require) implementations to recognize hexadecimal integer constants.

Historical implementations of *awk* did not support floating-point infinities and NaNs in *numeric strings*; e.g., `"-INF"` and `"NaN"`. However, implementations that use the `atof()` or `strtod()` functions to do the conversion picked up support for these values if they used a ISO/IEC 9899:1999 standard version of the function instead of a ISO/IEC 9899:1990 standard version. Due to an oversight, the 2001 through 2004 editions of this standard did not allow support for infinities and NaNs, but in this revision support is allowed (but not required). This is a silent change to the behavior of *awk* programs; for example, in the POSIX locale the expression:

```
("-INF" + 0 < 0)
```

formerly had the value 0 because `"-INF"` converted to 0, but now it may have the value 0 or 1.

## FUTURE DIRECTIONS

A future version of this standard may require the `!="` and `=="` operators to perform string comparisons by checking if the strings are identical (and not by checking if they collate equally).

## SEE ALSO

[Section 1.3](#) (on page 2325), [grep](#), [lex](#), [sed](#)

[XBD Chapter 5](#) (on page 121), [Section 6.1](#) (on page 125), [Chapter 8](#) (on page 173), [Chapter 9](#) (on page 181), [Section 12.2](#) (on page 216)

[XSH `atof\(\)`](#), [exec](#), [isspace\(\)](#), [popen\(\)](#), [setlocale\(\)](#), [strtod\(\)](#)

## CHANGE HISTORY

First released in Issue 2.

## Issue 5

The FUTURE DIRECTIONS section is added.

**Issue 6**

The *awk* utility is aligned with the IEEE P1003.2b draft standard.

The normative text is reworded to avoid use of the term “must” for application requirements.

IEEE PASC Interpretation 1003.2 #211 is applied, adding the sentence “An occurrence of two consecutive <backslash> characters shall be interpreted as just a single literal <backslash> character.” into the description of the **sub** string function.

**Issue 7**

PASC Interpretation 1003.2-1992 #107 (SD5-XCU-ERN-73) is applied, updating the description of the **OFS** variable.

Austin Group Interpretation 1003.1-2001 #189 is applied.

Austin Group Interpretation 1003.1-2001 #201 is applied, permitting implementations to support infinities and NaNs.

SD5-XCU-ERN-79 is applied, restoring the horizontal lines to [Table 4-1](#) (on page 2475), and SD5-XCU-ERN-80 is applied, changing the order of some table entries.

SD5-XCU-ERN-87 is applied, updating the descriptive text of the Grammar.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

The EXTENDED DESCRIPTION is changed to make the support of hexadecimal integer and floating constants optional.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0057 [224], XCU/TC1-2008/0058 [454], XCU/TC1-2008/0059 [224], XCU/TC1-2008/0060 [224], XCU/TC1-2008/0061 [254], XCU/TC1-2008/0062 [254], XCU/TC1-2008/0063 [224], and XCU/TC1-2008/0064 [454] are applied.

**NAME**

basename — return non-directory portion of a pathname

**SYNOPSIS**

basename *string* [*suffix*]

**DESCRIPTION**

The *string* operand shall be treated as a pathname, as defined in XBD [Section 3.271](#) (on page 76). The string *string* shall be converted to the filename corresponding to the last pathname component in *string* and then the suffix string *suffix*, if present, shall be removed. This shall be done by performing actions equivalent to the following steps in order:

1. If *string* is a null string, it is unspecified whether the resulting string is ' . ' or a null string. In either case, skip steps 2 through 6.
2. If *string* is " / ", it is implementation-defined whether steps 3 to 6 are skipped or processed.
3. If *string* consists entirely of <slash> characters, *string* shall be set to a single <slash> character. In this case, skip steps 4 to 6.
4. If there are any trailing <slash> characters in *string*, they shall be removed.
5. If there are any <slash> characters remaining in *string*, the prefix of *string* up to and including the last <slash> character in *string* shall be removed.
6. If the *suffix* operand is present, is not identical to the characters remaining in *string*, and is identical to a suffix of the characters remaining in *string*, the suffix *suffix* shall be removed from *string*. Otherwise, *string* is not modified by this step. It shall not be considered an error if *suffix* is not found in *string*.

The resulting string shall be written to standard output.

**OPTIONS**

None.

**OPERANDS**

The following operands shall be supported:

<i>string</i>	A string.
<i>suffix</i>	A string.

**STDIN**

Not used.

**INPUT FILES**

None.

**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *basename*:

<i>LANG</i>	Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)
<i>LC_ALL</i>	If set to a non-empty string value, override the values of all the other internationalization variables.
<i>LC_CTYPE</i>	Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).



80947 **LC\_MESSAGES**

80948 Determine the locale that should be used to affect the format and contents of  
 80949 diagnostic messages written to standard error.

80950 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

80951 **ASYNCHRONOUS EVENTS**

80952 Default.

80953 **STDOUT**

80954 The *basename* utility shall write a line to the standard output in the following format:

80955 "%s\n", *<resulting string>*

80956 **STDERR**

80957 The standard error shall be used only for diagnostic messages.

80958 **OUTPUT FILES**

80959 None.

80960 **EXTENDED DESCRIPTION**

80961 None.

80962 **EXIT STATUS**

80963 The following exit values shall be returned:

80964 0 Successful completion.

80965 >0 An error occurred.

80966 **CONSEQUENCES OF ERRORS**

80967 Default.

80968 **APPLICATION USAGE**

80969 The definition of *pathname* specifies implementation-defined behavior for pathnames starting  
 80970 with two <slash> characters. Therefore, applications shall not arbitrarily add <slash> characters  
 80971 to the beginning of a pathname unless they can ensure that there are more or less than two or are  
 80972 prepared to deal with the implementation-defined consequences.

80973 **EXAMPLES**

80974 If the string *string* is a valid pathname:

80975 `$(basename -- "string")`

80976 produces a filename that could be used to open the file named by *string* in the directory returned  
 80977 by:

80978 `$(dirname -- "string")`

80979 If the string *string* is not a valid pathname, the same algorithm is used, but the result need not be  
 80980 a valid filename. The *basename* utility is not expected to make any judgements about the validity  
 80981 of *string* as a pathname; it just follows the specified algorithm to produce a result string.

80982 The following shell script compiles `/usr/src/cmd/cat.c` and moves the output to a file named **cat**  
 80983 in the current directory when invoked with the argument `/usr/src/cmd/cat` or with the argument  
 80984 `/usr/src/cmd/cat.c`:

80985 `c99 -- "$(dirname -- "$1")/$(basename -- "$1" .c).c" &&  
 80986 mv a.out "$(basename -- "$1" .c)"`

80987 The EXAMPLES section of the *basename()* function (see XSH [basename\(\)](#)) includes a table +  
 80988 showing examples of the results of processing several sample pathnames by the *basename()* and +

80989 *dirname()* functions and by the *basename* and *dirname* utilities.

#### 80990 RATIONALE

80991 The behaviors of *basename* and *dirname* have been coordinated so that when *string* is a valid  
80992 pathname:

80993 `$(basename -- "string")`

80994 would be a valid filename for the file in the directory:

80995 `$(dirname -- "string")`

80996 This would not work for the early proposal versions of these utilities due to the way it specified  
80997 handling of trailing <slash> characters.

80998 Since the definition of *pathname* specifies implementation-defined behavior for pathnames  
80999 starting with two <slash> characters, this volume of POSIX.1-2008 specifies similar  
81000 implementation-defined behavior for the *basename* and *dirname* utilities.

#### 81001 FUTURE DIRECTIONS

81002 None.

#### 81003 SEE ALSO

81004 [Section 2.5](#) (on page 2339), *dirname*

81005 XBD [Section 3.271](#) (on page 76), [Chapter 8](#) (on page 173)

81006 XSH *basename()*, *dirname()* +

#### 81007 CHANGE HISTORY

81008 First released in Issue 2.

#### 81009 Issue 6

81010 IEEE PASC Interpretation 1003.2 #164 is applied.

81011 The normative text is reworded to avoid use of the term “must” for application requirements.

#### 81012 Issue 7

81013 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0065 [192,538], XCU/TC1-2008/0066  
81014 [192,538], and XCU/TC1-2008/0067 [192,430,538] are applied.

81015 **NAME**

81016 batch — schedule commands to be executed in a batch queue

81017 **SYNOPSIS**81018 *batch*81019 **DESCRIPTION**81020 The *batch* utility shall read commands from standard input and schedule them for execution in a  
81021 batch queue. It shall be the equivalent of the command:81022 `at -q b -m now`81023 where queue *b* is a special *at* queue, specifically for batch jobs. Batch jobs shall be submitted to  
81024 the batch queue with no time constraints and shall be run by the system using algorithms, based  
81025 on unspecified factors, that may vary with each invocation of *batch*.81026 XSI Users shall be permitted to use *batch* if their name appears in the file **at.allow** which is located in  
81027 an implementation-defined directory. If that file does not exist, the file **at.deny**, which is located  
81028 in an implementation-defined directory, shall be checked to determine whether the user shall be  
81029 denied access to *batch*. If neither file exists, only a process with appropriate privileges shall be  
81030 allowed to submit a job. If only **at.deny** exists and is empty, global usage shall be permitted. The  
81031 **at.allow** and **at.deny** files shall consist of one user name per line.81032 **OPTIONS**

81033 None.

81034 **OPERANDS**

81035 None.

81036 **STDIN**81037 The standard input shall be a text file consisting of commands acceptable to the shell command  
81038 language described in [Chapter 2](#) (on page 2335).81039 **INPUT FILES**81040 XSI The text files **at.allow** and **at.deny**, which are located in an implementation-defined directory,  
81041 shall contain zero or more user names, one per line, of users who are, respectively, authorized or  
81042 denied access to the *at* and *batch* utilities.81043 **ENVIRONMENT VARIABLES**81044 The following environment variables shall affect the execution of *batch*:81045 **LANG** Provide a default value for the internationalization variables that are unset or null.  
81046 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
81047 variables used to determine the values of locale categories.)81048 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
81049 internationalization variables.81050 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
81051 characters (for example, single-byte as opposed to multi-byte characters in  
81052 arguments and input files).81053 **LC\_MESSAGES**81054 Determine the locale that should be used to affect the format and contents of  
81055 diagnostic messages written to standard error and informative messages written to  
81056 standard output.81057 **LC\_TIME** Determine the format and contents for date and time strings written by *batch*.

81058	XSI	<b>NLSPATH</b>	Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .
81059		<b>SHELL</b>	Determine the name of a command interpreter to be used to invoke the at-job. If the variable is unset or null, <i>sh</i> shall be used. If it is set to a value other than a name for <i>sh</i> , the implementation shall do one of the following: use that shell; use <i>sh</i> ; use the login shell from the user database; any of the preceding accompanied by a warning diagnostic about which was chosen.
81060			
81061			
81062			
81063			
81064		<b>TZ</b>	Determine the timezone. The job shall be submitted for execution at the time specified by <i>timespec</i> or <i>-t time</i> relative to the timezone specified by the <i>TZ</i> variable. If <i>timespec</i> specifies a timezone, it overrides <i>TZ</i> . If <i>timespec</i> does not specify a timezone and <i>TZ</i> is unset or null, an unspecified default timezone shall be used.
81065			
81066			
81067			
81068			
81069		<b>ASYNCHRONOUS EVENTS</b>	
81070			Default.
81071		<b>STDOUT</b>	
81072			When standard input is a terminal, prompts of unspecified format for each line of the user input described in the STDIN section may be written to standard output.
81073			
81074		<b>STDERR</b>	
81075			The following shall be written to standard error when a job has been successfully submitted:
81076			"job %s at %s\n", <i>at_job_id</i> , < <i>date</i> >
81077			where <i>date</i> shall be equivalent in format to the output of:
81078			<i>date</i> + "%a %b %e %T %Y"
81079			The date and time written shall be adjusted so that they appear in the timezone of the user (as determined by the <i>TZ</i> variable).
81080			
81081			Neither this, nor warning messages concerning the selection of the command interpreter, are considered a diagnostic that changes the exit status.
81082			
81083			Diagnostic messages, if any, shall be written to standard error.
81084		<b>OUTPUT FILES</b>	
81085			None.
81086		<b>EXTENDED DESCRIPTION</b>	
81087			None.
81088		<b>EXIT STATUS</b>	
81089			The following exit values shall be returned:
81090			0 Successful completion.
81091			>0 An error occurred.
81092		<b>CONSEQUENCES OF ERRORS</b>	
81093			The job shall not be scheduled.

**81094 APPLICATION USAGE**

81095 It may be useful to redirect standard output within the specified commands.

**81096 EXAMPLES**

81097 1. This sequence can be used at a terminal:

```
81098 batch
81099 sort < file >outfile
81100 EOT
```

81101 2. This sequence, which demonstrates redirecting standard error to a pipe, is useful in a  
81102 command procedure (the sequence of output redirection specifications is significant):

```
81103 batch <<!  
81104 diff file1 file2 2>&1 >outfile | mailx mygroup  
81105 !
```

**81106 RATIONALE**

81107 Early proposals described *batch* in a manner totally separated from *at*, even though the historical  
81108 model treated it almost as a synonym for *at -qb*. A number of features were added to list and  
81109 control batch work separately from those in *at*. Upon further reflection, it was decided that the  
81110 benefit of this did not merit the change to the historical interface.

81111 The *-m* option was included on the equivalent *at* command because it is historical practice to  
81112 mail results to the submitter, even if all job-produced output is redirected. As explained in the  
81113 RATIONALE for *at*, the **now** keyword submits the job for immediate execution (after scheduling  
81114 delays), despite some historical systems where *at now* would have been considered an error.

**81115 FUTURE DIRECTIONS**

81116 None.

**81117 SEE ALSO**

81118 *at*

81119 XBD Chapter 8 (on page 173)

**81120 CHANGE HISTORY**

81121 First released in Issue 2.

**81122 Issue 6**

81123 This utility is marked as part of the User Portability Utilities option.

81124 The NAME is changed to align with the IEEE P1003.2b draft standard.

81125 The normative text is reworded to avoid use of the term “must” for application requirements.

**81126 Issue 7**

81127 The *batch* utility is moved from the User Portability Utilities option to the Base. User Portability  
81128 Utilities is now an option for interactive utilities.

81129 SD5-XCU-ERN-95 is applied, removing the references to fixed locations for the files referenced  
81130 by the *batch* utility.

81131 **NAME**

81132       bc — arbitrary-precision arithmetic language

81133 **SYNOPSIS**81134       bc [-l] [*file...*]81135 **DESCRIPTION**

81136       The *bc* utility shall implement an arbitrary precision calculator. It shall take input from any files  
 81137       given, then read from the standard input. If the standard input and standard output to *bc* are  
 81138       attached to a terminal, the invocation of *bc* shall be considered to be *interactive*, causing  
 81139       behavioral constraints described in the following sections.

81140 **OPTIONS**81141       The *bc* utility shall conform to XBD [Section 12.2](#) (on page 216).

81142       The following option shall be supported:

81143       -l           (The letter ell.) Define the math functions and initialize *scale* to 20, instead of the  
 81144       default zero; see the EXTENDED DESCRIPTION section.

81145 **OPERANDS**

81146       The following operand shall be supported:

81147       *file*        A pathname of a text file containing *bc* program statements. After all *files* have  
 81148       been read, *bc* shall read the standard input.

81149 **STDIN**

81150       See the INPUT FILES section.

81151 **INPUT FILES**

81152       Input files shall be text files containing a sequence of comments, statements, and function  
 81153       definitions that shall be executed as they are read.

81154 **ENVIRONMENT VARIABLES**81155       The following environment variables shall affect the execution of *bc*:

81156       LANG        Provide a default value for the internationalization variables that are unset or null.  
 81157                   (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 81158                   variables used to determine the values of locale categories.)

81159       LC\_ALL      If set to a non-empty string value, override the values of all the other  
 81160       internationalization variables.

81161       LC\_CTYPE    Determine the locale for the interpretation of sequences of bytes of text data as  
 81162       characters (for example, single-byte as opposed to multi-byte characters in  
 81163       arguments and input files).

81164       LC\_MESSAGES

81165                   Determine the locale that should be used to affect the format and contents of  
 81166       diagnostic messages written to standard error.

81167 xSI       NLSPATH   Determine the location of message catalogs for the processing of LC\_MESSAGES.

81168 **ASYNCHRONOUS EVENTS**

81169       Default.

81170 **STDOUT**

81171       The output of the *bc* utility shall be controlled by the program read, and consist of zero or more  
 81172       lines containing the value of all executed expressions without assignments. The radix and  
 81173       precision of the output shall be controlled by the values of the **obase** and **scale** variables; see the  
 81174       EXTENDED DESCRIPTION section.

81175 **STDERR**

81176 The standard error shall be used only for diagnostic messages.

81177 **OUTPUT FILES**

81178 None.

81179 **EXTENDED DESCRIPTION**81180 **Grammar**

81181 The grammar in this section and the lexical conventions in the following section shall together  
 81182 describe the syntax for *bc* programs. The general conventions for this style of grammar are  
 81183 described in [Section 1.3](#) (on page 2325). A valid program can be represented as the non-terminal  
 81184 symbol **program** in the grammar. This formal syntax shall take precedence over the text syntax  
 81185 description.

```

81186 %token      EOF NEWLINE STRING LETTER NUMBER
81187 %token      MUL_OP
81188 /*          '*', '/', '%'                               */
81189 %token      ASSIGN_OP
81190 /*          '=', '+=', '-=', '*=', '/=', '%=', '^=' */
81191 %token      REL_OP
81192 /*          '==', '<=', '>=', '!=', '<', '>'               */
81193 %token      INCR DECR
81194 /*          '++', '--'                                   */
81195 %token      Define Break Quit Length
81196 /*          'define', 'break', 'quit', 'length'         */
81197 %token      Return For If While Sqrt
81198 /*          'return', 'for', 'if', 'while', 'sqrt'      */
81199 %token      Scale Ibase Obase Auto
81200 /*          'scale', 'ibase', 'obase', 'auto'          */
81201 %start      program
81202 %%
81203 program      : EOF
81204               | input_item program
81205               ;
81206 input_item    : semicolon_list NEWLINE
81207               | function
81208               ;
81209 semicolon_list : /* empty */
81210               | statement
81211               | semicolon_list ';' statement
81212               | semicolon_list ';'
81213               ;
81214 statement_list : /* empty */
81215               | statement
81216               | statement_list NEWLINE

```

```

81217         | statement_list NEWLINE statement
81218         | statement_list ';'
81219         | statement_list ';' statement
81220         ;

81221     statement      : expression
81222         | STRING
81223         | Break
81224         | Quit
81225         | Return
81226         | Return '(' return_expression ')'
81227         | For '(' expression ';'
81228             relational_expression ';'
81229             expression ')' statement
81230         | If '(' relational_expression ')' statement
81231         | While '(' relational_expression ')' statement
81232         | '{' statement_list '}'
81233         ;

81234     function        : Define LETTER '(' opt_parameter_list ')'
81235                     | '{' NEWLINE opt_auto_define_list
81236                     | statement_list '}'
81237                     ;

81238     opt_parameter_list : /* empty */
81239         | parameter_list
81240         ;

81241     parameter_list    : LETTER
81242         | define_list ',' LETTER
81243         ;

81244     opt_auto_define_list : /* empty */
81245         | Auto define_list NEWLINE
81246         | Auto define_list ';'
81247         ;

81248     define_list        : LETTER
81249         | LETTER '[' ']'
81250         | define_list ',' LETTER
81251         | define_list ',' LETTER '[' ']'
81252         ;

81253     opt_argument_list  : /* empty */
81254         | argument_list
81255         ;

81256     argument_list      : expression
81257         | LETTER '[' ']' ',' argument_list
81258         ;

81259     relational_expression : expression
81260         | expression REL_OP expression
81261         ;

81262     return_expression   : /* empty */
81263         | expression

```



```

81264                                     ;
81265     expression                       : named_expression
81266                                     | NUMBER
81267                                     | '(' expression ')'
81268                                     | LETTER '(' opt_argument_list ')'
81269                                     | '-' expression
81270                                     | expression '+' expression
81271                                     | expression '-' expression
81272                                     | expression MUL_OP expression
81273                                     | expression '^' expression
81274                                     | INCR_DECR named_expression
81275                                     | named_expression INCR_DECR
81276                                     | named_expression ASSIGN_OP expression
81277                                     | Length '(' expression ')'
81278                                     | Sqrt '(' expression ')'
81279                                     | Scale '(' expression ')'
81280                                     ;
81281     named_expression                  : LETTER
81282                                     | LETTER '[' expression ']'
81283                                     | Scale
81284                                     | Ibase
81285                                     | Obase
81286                                     ;

```

### Lexical Conventions in bc

The lexical conventions for *bc* programs, with respect to the preceding grammar, shall be as follows:

1. Except as noted, *bc* shall recognize the longest possible token or delimiter beginning at a given point.
2. A comment shall consist of any characters beginning with the two adjacent characters `"/*"` and terminated by the next occurrence of the two adjacent characters `"*/"`. Comments shall have no effect except to delimit lexical tokens.
3. The `<newline>` shall be recognized as the token **NEWLINE**.
4. The token **STRING** shall represent a string constant; it shall consist of any characters beginning with the double-quote character (`'"`) and terminated by another occurrence of the double-quote character. The value of the string is the sequence of all characters between, but not including, the two double-quote characters. All characters shall be taken literally from the input, and there is no way to specify a string containing a double-quote character. The length of the value of each string shall be limited to `{BC_STRING_MAX}` bytes.
5. A `<blank>` shall have no effect except as an ordinary character if it appears within a **STRING** token, or to delimit a lexical token other than **STRING**.
6. The combination of a `<backslash>` character immediately followed by a `<newline>` shall have no effect other than to delimit lexical tokens with the following exceptions:

- 81307 • It shall be interpreted as the character sequence "\<newline>" in **STRING** tokens.
- 81308 • It shall be ignored as part of a multi-line **NUMBER** token.

81309 7. The token **NUMBER** shall represent a numeric constant. It shall be recognized by the  
81310 following grammar:

```
81311 NUMBER : integer
81312         | '.' integer
81313         | integer '.'
81314         | integer '.' integer
81315         ;

81316 integer : digit
81317         | integer digit
81318         ;

81319 digit   : 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7
81320         | 8 | 9 | A | B | C | D | E | F
81321         ;
```

81322 8. The value of a **NUMBER** token shall be interpreted as a numeral in the base specified by  
81323 the value of the internal register **ibase** (described below). Each of the **digit** characters  
81324 shall have the value from 0 to 15 in the order listed here, and the <period> character shall  
81325 represent the radix point. The behavior is undefined if digits greater than or equal to the  
81326 value of **ibase** appear in the token. However, note the exception for single-digit values  
81327 being assigned to **ibase** and **obase** themselves, in [Operations in bc](#) (on page 2519).

81328 9. The following keywords shall be recognized as tokens:

81329 <b>auto</b>	<b>ibase</b>	<b>length</b>	<b>return</b>	<b>while</b>
81330 <b>break</b>	<b>if</b>	<b>obase</b>	<b>scale</b>	
81331 <b>define</b>	<b>for</b>	<b>quit</b>	<b>sqrt</b>	

81332 10. Any of the following characters occurring anywhere except within a keyword shall be  
81333 recognized as the token **LETTER**:

81334 a b c d e f g h i j k l m n o p q r s t u v w x y z

81335 11. The following single-character and two-character sequences shall be recognized as the  
81336 token **ASSIGN\_OP**:

81337 = += -= \*= /= %= ^=

81338 12. If an '=' character, as the beginning of a token, is followed by a '-' character with no  
81339 intervening delimiter, the behavior is undefined.

81340 13. The following single-characters shall be recognized as the token **MUL\_OP**:

81341 \* / %

81342 14. The following single-character and two-character sequences shall be recognized as the  
81343 token **REL\_OP**:

81344 == <= >= != < >

81345 15. The following two-character sequences shall be recognized as the token **INCR\_DECR**:

81346 ++ --

16. The following single characters shall be recognized as tokens whose names are the character:

<newline> ( ) , + - ; [ ] ^ { }

17. The token **EOF** is returned when the end of input is reached.

### Operations in bc

There are three kinds of identifiers: ordinary identifiers, array identifiers, and function identifiers. All three types consist of single lowercase letters. Array identifiers shall be followed by square brackets ("[]"). An array subscript is required except in an argument or auto list. Arrays are singly dimensioned and can contain up to {BC\_DIM\_MAX} elements. Indexing shall begin at zero so an array is indexed from 0 to {BC\_DIM\_MAX}-1. Subscripts shall be truncated to integers. The application shall ensure that function identifiers are followed by parentheses, possibly enclosing arguments. The three types of identifiers do not conflict.

The following table summarizes the rules for precedence and associativity of all operators. Operators on the same line shall have the same precedence; rows are in order of decreasing precedence.

**Table 4-3** Operators in *bc*

Operator	Associativity
++, --	N/A
unary -	N/A
^	Right to left
*, /, %	Left to right
+, binary -	Left to right
=, +=, -=, *=, /=, %=, ^=	Right to left
==, <=, >=, !=, <, >	None

Each expression or named expression has a *scale*, which is the number of decimal digits that shall be maintained as the fractional portion of the expression.

*Named expressions* are places where values are stored. Named expressions shall be valid on the left side of an assignment. The value of a named expression shall be the value stored in the place named. Simple identifiers and array elements are named expressions; they have an initial value of zero and an initial scale of zero.

The internal registers **scale**, **ibase**, and **obase** are all named expressions. The scale of an expression consisting of the name of one of these registers shall be zero; values assigned to any of these registers are truncated to integers. The **scale** register shall contain a global value used in computing the scale of expressions (as described below). The value of the register **scale** is limited to  $0 \leq \text{scale} \leq \{\text{BC\_SCALE\_MAX}\}$  and shall have a default value of zero. The **ibase** and **obase** registers are the input and output number radix, respectively. The value of **ibase** shall be limited to:

$$2 \leq \text{ibase} \leq 16$$

The value of **obase** shall be limited to:

$$2 \leq \text{obase} \leq \{\text{BC\_BASE\_MAX}\}$$

When either **ibase** or **obase** is assigned a single **digit** value from the list in [Lexical Conventions in bc](#) (on page 2517), the value shall be assumed in hexadecimal. (For example, **ibase=A** sets to base ten, regardless of the current **ibase** value.) Otherwise, the behavior is undefined when

digits greater than or equal to the value of **ibase** appear in the input. Both **ibase** and **obase** shall have initial values of 10.

Internal computations shall be conducted as if in decimal, regardless of the input and output bases, to the specified number of decimal digits. When an exact result is not achieved (for example, **scale**=0; 3.2/1), the result shall be truncated.

For all values of **obase** specified by this volume of POSIX.1-2008, *bc* shall output numeric values by performing each of the following steps in order:

1. If the value is less than zero, a <hyphen-minus> ( ' - ' ) character shall be output.
2. One of the following is output, depending on the numerical value:
  - If the absolute value of the numerical value is greater than or equal to one, the integer portion of the value shall be output as a series of digits appropriate to **obase** (as described below), most significant digit first. The most significant non-zero digit shall be output next, followed by each successively less significant digit.
  - If the absolute value of the numerical value is less than one but greater than zero and the scale of the numerical value is greater than zero, it is unspecified whether the character 0 is output.
  - If the numerical value is zero, the character 0 shall be output.
3. If the scale of the value is greater than zero and the numeric value is not zero, a <period> character shall be output, followed by a series of digits appropriate to **obase** (as described below) representing the most significant portion of the fractional part of the value. If *s* represents the scale of the value being output, the number of digits output shall be *s* if **obase** is 10, less than or equal to *s* if **obase** is greater than 10, or greater than or equal to *s* if **obase** is less than 10. For **obase** values other than 10, this should be the number of digits needed to represent a precision of 10<sup>s</sup>.

For **obase** values from 2 to 16, valid digits are the first **obase** of the single characters:

0 1 2 3 4 5 6 7 8 9 A B C D E F

which represent the values zero to 15, inclusive, respectively.

For bases greater than 16, each digit shall be written as a separate multi-digit decimal number. Each digit except the most significant fractional digit shall be preceded by a single <space>. For bases from 17 to 100, *bc* shall write two-digit decimal numbers; for bases from 101 to 1 000, three-digit decimal strings, and so on. For example, the decimal number 1 024 in base 25 would be written as:

Δ01Δ15Δ24

and in base 125, as:

Δ008Δ024

Very large numbers shall be split across lines with 70 characters per line in the POSIX locale; other locales may split at different character boundaries. Lines that are continued shall end with a <backslash>.

A function call shall consist of a function name followed by parentheses containing a <comma>-separated list of expressions, which are the function arguments. A whole array passed as an argument shall be specified by the array name followed by empty square brackets. All function arguments shall be passed by value. As a result, changes made to the formal parameters shall have no effect on the actual arguments. If the function terminates by executing

81433 a **return** statement, the value of the function shall be the value of the expression in the  
 81434 parentheses of the **return** statement or shall be zero if no expression is provided or if there is no  
 81435 **return** statement.

81436 The result of **sqrt**(*expression*) shall be the square root of the expression. The result shall be  
 81437 truncated in the least significant decimal place. The scale of the result shall be the scale of the  
 81438 expression or the value of **scale**, whichever is larger.

81439 The result of **length**(*expression*) shall be the total number of significant decimal digits in the  
 81440 expression. The scale of the result shall be zero.

81441 The result of **scale**(*expression*) shall be the scale of the expression. The scale of the result shall be  
 81442 zero.

81443 A numeric constant shall be an expression. The scale shall be the number of digits that follow the  
 81444 radix point in the input representing the constant, or zero if no radix point appears.

81445 The sequence ( *expression* ) shall be an expression with the same value and scale as *expression*.  
 81446 The parentheses can be used to alter the normal precedence.

81447 The semantics of the unary and binary operators are as follows:

81448 *-expression*

81449 The result shall be the negative of the *expression*. The scale of the result shall be the scale of  
 81450 *expression*.

81451 The unary increment and decrement operators shall not modify the scale of the named  
 81452 expression upon which they operate. The scale of the result shall be the scale of that named  
 81453 expression.

81454 *++named-expression*

81455 The named expression shall be incremented by one. The result shall be the value of the  
 81456 named expression after incrementing.

81457 *--named-expression*

81458 The named expression shall be decremented by one. The result shall be the value of the  
 81459 named expression after decrementing.

81460 *named-expression++*

81461 The named expression shall be incremented by one. The result shall be the value of the  
 81462 named expression before incrementing.

81463 *named-expression--*

81464 The named expression shall be decremented by one. The result shall be the value of the  
 81465 named expression before decrementing.

81466 The exponentiation operator, <circumflex> ( ' ^ ' ), shall bind right to left.

81467 *expression^expression*

81468 The result shall be the first *expression* raised to the power of the second *expression*. If the  
 81469 second expression is not an integer, the behavior is undefined. If *a* is the scale of the left  
 81470 expression and *b* is the absolute value of the right expression, the scale of the result shall be:

81471 if *b* >= 0 min(*a* \* *b*, max(scale, *a*)) if *b* < 0 scale

81472 The multiplicative operators ( ' \* ', ' / ', ' % ' ) shall bind left to right.

81473 *expression\*expression*

81474 The result shall be the product of the two expressions. If *a* and *b* are the scales of the two  
 81475 expressions, then the scale of the result shall be:

81476  $\min(a+b, \max(\text{scale}, a, b))$

81477 *expression / expression*

81478 The result shall be the quotient of the two expressions. The scale of the result shall be the  
81479 value of **scale**.

81480 *expression % expression*

81481 For expressions *a* and *b*, *a % b* shall be evaluated equivalent to the steps:

81482 1. Compute *a / b* to current scale.

81483 2. Use the result to compute:

81484  $a - (a / b) * b$

81485 to scale:

81486  $\max(\text{scale} + \text{scale}(b), \text{scale}(a))$

81487 The scale of the result shall be:

81488  $\max(\text{scale} + \text{scale}(b), \text{scale}(a))$

81489 When **scale** is zero, the '**%**' operator is the mathematical remainder operator.

81490 The additive operators ('+', '-') shall bind left to right.

81491 *expression + expression*

81492 The result shall be the sum of the two expressions. The scale of the result shall be the  
81493 maximum of the scales of the expressions.

81494 *expression - expression*

81495 The result shall be the difference of the two expressions. The scale of the result shall be the  
81496 maximum of the scales of the expressions.

81497 The assignment operators ('=', '+=', '-=', '\*=', '/=', '%=', '^=') shall bind right to left.

81498 *named-expression = expression*

81499 This expression shall result in assigning the value of the expression on the right to the  
81500 named expression on the left. The scale of both the named expression and the result shall be  
81501 the scale of *expression*.

81502 The compound assignment forms:

81503 *named-expression <operator>= expression*

81504 shall be equivalent to:

81505 *named-expression = named-expression <operator> expression*

81506 except that the *named-expression* shall be evaluated only once.

81507 Unlike all other operators, the relational operators ('<', '>', '<=', '>=', '==', '!=') shall be  
81508 only valid as the object of an **if**, **while**, or inside a **for** statement.

81509 *expression1 < expression2*

81510 The relation shall be true if the value of *expression1* is strictly less than the value of  
81511 *expression2*.

81512 *expression1 > expression2*

81513 The relation shall be true if the value of *expression1* is strictly greater than the value of  
81514 *expression2*.

81515 *expression1* <=*expression2*  
 81516 The relation shall be true if the value of *expression1* is less than or equal to the value of  
 81517 *expression2*.

81518 *expression1* >=*expression2*  
 81519 The relation shall be true if the value of *expression1* is greater than or equal to the value of  
 81520 *expression2*.

81521 *expression1* ==*expression2*  
 81522 The relation shall be true if the values of *expression1* and *expression2* are equal.

81523 *expression1* !=*expression2*  
 81524 The relation shall be true if the values of *expression1* and *expression2* are unequal.

81525 There are only two storage classes in *bc*: global and automatic (local). Only identifiers that are  
 81526 local to a function need be declared with the **auto** command. The arguments to a function shall  
 81527 be local to the function. All other identifiers are assumed to be global and available to all  
 81528 functions. All identifiers, global and local, have initial values of zero. Identifiers declared as auto  
 81529 shall be allocated on entry to the function and released on returning from the function. They  
 81530 therefore do not retain values between function calls. Auto arrays shall be specified by the array  
 81531 name followed by empty square brackets. On entry to a function, the old values of the names  
 81532 that appear as parameters and as automatic variables shall be pushed onto a stack. Until the  
 81533 function returns, reference to these names shall refer only to the new values.

81534 References to any of these names from other functions that are called from this function also  
 81535 refer to the new value until one of those functions uses the same name for a local variable.

81536 When a statement is an expression, unless the main operator is an assignment, execution of the  
 81537 statement shall write the value of the expression followed by a <newline>.

81538 When a statement is a string, execution of the statement shall write the value of the string.

81539 Statements separated by <semicolon> or <newline> characters shall be executed sequentially. In  
 81540 an interactive invocation of *bc*, each time a <newline> is read that satisfies the grammatical  
 81541 production:

81542 `input_item : semicolon_list NEWLINE`

81543 the sequential list of statements making up the **semicolon\_list** shall be executed immediately  
 81544 and any output produced by that execution shall be written without any delay due to buffering.

81545 In an **if** statement (**if**(*relation*) *statement*), the *statement* shall be executed if the relation is true.

81546 The **while** statement (**while**(*relation*) *statement*) implements a loop in which the *relation* is tested;  
 81547 each time the *relation* is true, the *statement* shall be executed and the *relation* retested. When the  
 81548 *relation* is false, execution shall resume after *statement*.

81549 A **for** statement(**for**(*expression*; *relation*; *expression*) *statement*) shall be the same as:

81550 `first-expression`  
 81551 `while (relation) {`  
 81552 `statement`  
 81553 `last-expression`  
 81554 `}`

81555 The application shall ensure that all three expressions are present.

81556 The **break** statement shall cause termination of a **for** or **while** statement.

81557 The **auto** statement (**auto** *identifier* [*identifier*] ...) shall cause the values of the identifiers to be



pushed down. The identifiers can be ordinary identifiers or array identifiers. Array identifiers shall be specified by following the array name by empty square brackets. The application shall ensure that the **auto** statement is the first statement in a function definition.

A **define** statement:

```
define LETTER ( opt_parameter_list ) {
    opt_auto_define_list
    statement_list
}
```

defines a function named **LETTER**. If a function named **LETTER** was previously defined, the **define** statement shall replace the previous definition. The expression:

```
LETTER ( opt_argument_list )
```

shall invoke the function named **LETTER**. The behavior is undefined if the number of arguments in the invocation does not match the number of parameters in the definition. Functions shall be defined before they are invoked. A function shall be considered to be defined within its own body, so recursive calls are valid. The values of numeric constants within a function shall be interpreted in the base specified by the value of the **ibase** register when the function is invoked.

The **return** statements (**return** and **return(expression)**) shall cause termination of a function, popping of its auto variables, and specification of the result of the function. The first form shall be equivalent to **return(0)**. The value and scale of the result returned by the function shall be the value and scale of the expression returned.

The **quit** statement (**quit**) shall stop execution of a *bc* program at the point where the statement occurs in the input, even if it occurs in a function definition, or in an **if**, **for**, or **while** statement.

The following functions shall be defined when the **-l** option is specified:

**s(expression)**

Sine of argument in radians.

**c(expression)**

Cosine of argument in radians.

**a(expression)**

Arctangent of argument.

**l(expression)**

Natural logarithm of argument.

**e(expression)**

Exponential function of argument.

**j(expression1, expression2)**

Bessel function of *expression2* of the first kind of integer order *expression1*.

The scale of the result returned by these functions shall be the value of the **scale** register at the time the function is invoked. The value of the **scale** register after these functions have completed their execution shall be the same value it had upon invocation. The behavior is undefined if any of these functions is invoked with an argument outside the domain of the mathematical function.



81599 **EXIT STATUS**

81600 The following exit values shall be returned:

81601 0 All input files were processed successfully.

81602 *unspecified* An error occurred.81603 **CONSEQUENCES OF ERRORS**81604 If any *file* operand is specified and the named file cannot be accessed, *bc* shall write a diagnostic message to standard error and terminate without any further action.81606 In an interactive invocation of *bc*, the utility should print an error message and recover following any error in the input. In a non-interactive invocation of *bc*, invalid input causes undefined behavior.81609 **APPLICATION USAGE**81610 Automatic variables in *bc* do not work in exactly the same way as in either C or PL/1.81611 For historical reasons, the exit status from *bc* cannot be relied upon to indicate that an error has occurred. Returning zero after an error is possible. Therefore, *bc* should be used primarily by interactive users (who can react to error messages) or by application programs that can somehow validate the answers returned as not including error messages.81615 The *bc* utility always uses the <period> ( ' . ' ) character to represent a radix point, regardless of any decimal-point character specified as part of the current locale. In languages like C or *awk*, the <period> character is used in program source, so it can be portable and unambiguous, while the locale-specific character is used in input and output. Because there is no distinction between source and input in *bc*, this arrangement would not be possible. Using the locale-specific character in *bc*'s input would introduce ambiguities into the language; consider the following example in a locale with a <comma> as the decimal-point character:

```
81622 define f(a,b) {
81623     ...
81624 }
81625 ...
81626 f(1,2,3)
```

81627 Because of such ambiguities, the &lt;period&gt; character is used in input. Having input follow different conventions from output would be confusing in either pipeline usage or interactive usage, so the &lt;period&gt; is also used in output.

81630 **EXAMPLES**81631 In the shell, the following assigns an approximation of the first ten digits of ' $\pi$ ' to the variable *x*:81632 `x=$(printf "%s\n" 'scale = 10; 104348/33215' | bc)`81633 The following *bc* program prints the same approximation of ' $\pi$ ', with a label, to standard output:

```
81635 scale = 10
81636 "pi equals "
81637 104348 / 33215
```

81638 The following defines a function to compute an approximate value of the exponential function (note that such a function is predefined if the `-l` option is specified):

```
81640 scale = 20
81641 define e(x){
81642     auto a, b, c, i, s
```

```

81643     a = 1
81644     b = 1
81645     s = 1
81646     for (i = 1; 1 == 1; i++){
81647         a = a*x
81648         b = b*i
81649         c = a/b
81650         if (c == 0) {
81651             return(s)
81652         }
81653         s = s+c
81654     }
81655 }

```

The following prints approximate values of the exponential function of the first ten integers:

```

81657 for (i = 1; i <= 10; ++i) {
81658     e(i)
81659 }

```

## RATIONALE

The *bc* utility is implemented historically as a front-end processor for *dc*; *dc* was not selected to be part of this volume of POSIX.1-2008 because *bc* was thought to have a more intuitive programmatic interface. Current implementations that implement *bc* using *dc* are expected to be compliant.

The exit status for error conditions has been left unspecified for several reasons:

- The *bc* utility is used in both interactive and non-interactive situations. Different exit codes may be appropriate for the two uses.
- It is unclear when a non-zero exit should be given; divide-by-zero, undefined functions, and syntax errors are all possibilities.
- It is not clear what utility the exit status has.
- In the 4.3 BSD, System V, and Ninth Edition implementations, *bc* works in conjunction with *dc*. The *dc* utility is the parent, *bc* is the child. This was done to cleanly terminate *bc* if *dc* aborted.

The decision to have *bc* exit upon encountering an inaccessible input file is based on the belief that *bc file1 file2* is used most often when at least *file1* contains data/function declarations/initializations. Having *bc* continue with prerequisite files missing is probably not useful. There is no implication in the CONSEQUENCES OF ERRORS section that *bc* must check all its files for accessibility before opening any of them.

There was considerable debate on the appropriateness of the language accepted by *bc*. Several reviewers preferred to see either a pure subset of the C language or some changes to make the language more compatible with C. While the *bc* language has some obvious similarities to C, it has never claimed to be compatible with any version of C. An interpreter for a subset of C might be a very worthwhile utility, and it could potentially make *bc* obsolete. However, no such utility is known in historical practice, and it was not within the scope of this volume of POSIX.1-2008 to define such a language and utility. If and when they are defined, it may be appropriate to include them in a future version of this standard. This left the following alternatives:

1. Exclude any calculator language from this volume of POSIX.1-2008.

The consensus of the standard developers was that a simple programmatic calculator language is very useful for both applications and interactive users. The only arguments for excluding any calculator were that it would become obsolete if and when a C-compatible one emerged, or that the absence would encourage the development of such a C-compatible one. These arguments did not sufficiently address the needs of current application developers.

2. Standardize the historical *dc*, possibly with minor modifications.

The consensus of the standard developers was that *dc* is a fundamentally less usable language and that that would be far too severe a penalty for avoiding the issue of being similar to but incompatible with C.

3. Standardize the historical *bc*, possibly with minor modifications.

This was the approach taken. Most of the proponents of changing the language would not have been satisfied until most or all of the incompatibilities with C were resolved. Since most of the changes considered most desirable would break historical applications and require significant modification to historical implementations, almost no modifications were made. The one significant modification that was made was the replacement of the historical *bc* assignment operators "*=*", and so on, with the more modern "*+=*", and so on. The older versions are considered to be fundamentally flawed because of the lexical ambiguity in uses like *a=-1*.

In order to permit implementations to deal with backwards-compatibility as they see fit, the behavior of this one ambiguous construct was made undefined. (At least three implementations have been known to support this change already, so the degree of change involved should not be great.)

The '*%*' operator is the mathematical remainder operator when **scale** is zero. The behavior of this operator for other values of **scale** is from historical implementations of *bc*, and has been maintained for the sake of historical applications despite its non-intuitive nature.

Historical implementations permit setting **ibase** and **obase** to a broader range of values. This includes values less than 2, which were not seen as sufficiently useful to standardize. These implementations do not interpret input properly for values of **ibase** that are greater than 16. This is because numeric constants are recognized syntactically, rather than lexically, as described in this volume of POSIX.1-2008. They are built from lexical tokens of single hexadecimal digits and <period> characters. Since <blank> characters between tokens are not visible at the syntactic level, it is not possible to recognize the multi-digit "digits" used in the higher bases properly. The ability to recognize input in these bases was not considered useful enough to require modifying these implementations. Note that the recognition of numeric constants at the syntactic level is not a problem with conformance to this volume of POSIX.1-2008, as it does not impact the behavior of conforming applications (and correct *bc* programs). Historical implementations also accept input with all of the digits '*0*'–'*9*' and '*A*'–'*F*' regardless of the value of **ibase**; since digits with value greater than or equal to **ibase** are not really appropriate, the behavior when they appear is undefined, except for the common case of:

```
ibase=8;
/* Process in octal base. */
...
ibase=A
/* Restore decimal base. */
```

In some historical implementations, if the expression to be written is an uninitialized array

81734 element, a leading <space> and/or up to four leading 0 characters may be output before the  
81735 character zero. This behavior is considered a bug; it is unlikely that any currently conforming  
81736 application relies on:

81737 `echo 'b[3]' | bc`

81738 returning 00000 rather than 0.

81739 Exact calculation of the number of fractional digits to output for a given value in a base other  
81740 than 10 can be computationally expensive. Historical implementations use a faster  
81741 approximation, and this is permitted. Note that the requirements apply only to values of **obase**  
81742 that this volume of POSIX.1-2008 requires implementations to support (in particular, not to 1, 0,  
81743 or negative bases, if an implementation supports them as an extension).

81744 Historical implementations of *bc* did not allow array parameters to be passed as the last  
81745 parameter to a function. New implementations are encouraged to remove this restriction even  
81746 though it is not required by the grammar.

#### 81747 **FUTURE DIRECTIONS**

81748 None.

#### 81749 **SEE ALSO**

81750 [Section 1.3](#) (on page 2325), *awk*

81751 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

#### 81752 **CHANGE HISTORY**

81753 First released in Issue 4.

##### 81754 **Issue 5**

81755 The FUTURE DIRECTIONS section is added.

##### 81756 **Issue 6**

81757 Updated to align with the IEEE P1003.2b draft standard, which included resolution of several  
81758 interpretations of the ISO POSIX-2: 1993 standard.

81759 The normative text is reworded to avoid use of the term “must” for application requirements.

##### 81760 **Issue 7**

81761 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

81762 **NAME**81763 `bg` — run jobs in the background81764 **SYNOPSIS**81765 UP `bg [job_id...]`81766 **DESCRIPTION**

81767 If job control is enabled (see the description of `set -m`), the `bg` utility shall resume suspended jobs  
 81768 from the current environment (see [Section 2.12](#), on page 2371) by running them as background  
 81769 jobs. If the job specified by `job_id` is already a running background job, the `bg` utility shall have  
 81770 no effect and shall exit successfully.

81771 Using `bg` to place a job into the background shall cause its process ID to become “known in the  
 81772 current shell execution environment”, as if it had been started as an asynchronous list; see  
 81773 [Section 2.9.3.1](#) (on page 2360).

81774 **OPTIONS**

81775 None.

81776 **OPERANDS**

81777 The following operand shall be supported:

81778 `job_id` Specify the job to be resumed as a background job. If no `job_id` operand is given,  
 81779 the most recently suspended job shall be used. The format of `job_id` is described in  
 81780 XBD [Section 3.204](#) (on page 66).

81781 **STDIN**

81782 Not used.

81783 **INPUT FILES**

81784 None.

81785 **ENVIRONMENT VARIABLES**81786 The following environment variables shall affect the execution of `bg`:

81787 `LANG` Provide a default value for the internationalization variables that are unset or null.  
 81788 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 81789 variables used to determine the values of locale categories.)

81790 `LC_ALL` If set to a non-empty string value, override the values of all the other  
 81791 internationalization variables.

81792 `LC_CTYPE` Determine the locale for the interpretation of sequences of bytes of text data as  
 81793 characters (for example, single-byte as opposed to multi-byte characters in  
 81794 arguments).

81795 `LC_MESSAGES`

81796 Determine the locale that should be used to affect the format and contents of  
 81797 diagnostic messages written to standard error.

81798 XSI `NLSPATH` Determine the location of message catalogs for the processing of `LC_MESSAGES`.

81799 **ASYNCHRONOUS EVENTS**

81800 Default.

81801 **STDOUT**81802 The output of `bg` shall consist of a line in the format:81803 `"[%d] %s\n", <job-number>, <command>`

81804 where the fields are as follows:

81805 <job-number> A number that can be used to identify the job to the *wait*, *fg*, and *kill* utilities. Using  
81806 these utilities, the job can be identified by prefixing the job number with ' % '.

81807 <command> The associated command that was given to the shell.

#### 81808 **STDERR**

81809 The standard error shall be used only for diagnostic messages.

#### 81810 **OUTPUT FILES**

81811 None.

#### 81812 **EXTENDED DESCRIPTION**

81813 None.

#### 81814 **EXIT STATUS**

81815 The following exit values shall be returned:

81816 0 Successful completion.

81817 >0 An error occurred.

#### 81818 **CONSEQUENCES OF ERRORS**

81819 If job control is disabled, the *bg* utility shall exit with an error and no job shall be placed in the  
81820 background.

#### 81821 **APPLICATION USAGE**

81822 A job is generally suspended by typing the SUSP character (<control>-Z on most systems); see  
81823 XBD [Chapter 11](#) (on page 199). At that point, *bg* can put the job into the background. This is  
81824 most effective when the job is expecting no terminal input and its output has been redirected to  
81825 non-terminal files. A background job can be forced to stop when it has terminal output by  
81826 issuing the command:

81827 `stty tostop`

81828 A background job can be stopped with the command:

81829 `kill -s stop job ID`

81830 The *bg* utility does not work as expected when it is operating in its own utility execution  
81831 environment because that environment has no suspended jobs. In the following examples:

81832 ... | xargs bg  
81833 (bg)

81834 each *bg* operates in a different environment and does not share its parent shell's understanding  
81835 of jobs. For this reason, *bg* is generally implemented as a shell regular built-in.

#### 81836 **EXAMPLES**

81837 None.

#### 81838 **RATIONALE**

81839 The extensions to the shell specified in this volume of POSIX.1-2008 have mostly been based on  
81840 features provided by the KornShell. The job control features provided by *bg*, *fg*, and *jobs* are also  
81841 based on the KornShell. The standard developers examined the characteristics of the C shell  
81842 versions of these utilities and found that differences exist. Despite widespread use of the C shell,  
81843 the KornShell versions were selected for this volume of POSIX.1-2008 to maintain a degree of  
81844 uniformity with the rest of the KornShell features selected (such as the very popular command  
81845 line editing features).

81846 The *bg* utility is expected to wrap its output if the output exceeds the number of display  
81847 columns.

81848 **FUTURE DIRECTIONS**

81849 None.

81850 **SEE ALSO**

81851 [Section 2.9.3.1](#) (on page 2360), *fg*, *kill*, *jobs*, *wait*

81852 XBD [Section 3.204](#) (on page 66), [Chapter 8](#) (on page 173), [Chapter 11](#) (on page 199)

81853 **CHANGE HISTORY**

81854 First released in Issue 4.

81855 **Issue 6**

81856 This utility is marked as part of the User Portability Utilities option.

81857 The JC margin marker on the SYNOPSIS is removed since support for Job Control is mandatory  
81858 in this version. This is a FIPS requirement.

81859 **Issue 7**

81860 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



**NAME**

c99 — compile standard C programs

**SYNOPSIS**

```
c99 [options...] pathname [[pathname] [-I directory]
[-L directory] [-l library]]...
```

**DESCRIPTION**

The *c99* utility is an interface to the standard C compilation system; it shall accept source code conforming to the ISO C standard. The system conceptually consists of a compiler and link editor. The input files referenced by *pathname* operands and *-I* option-arguments shall be compiled and linked to produce an executable file. (It is unspecified whether the linking occurs entirely within the operation of *c99*; some implementations may produce objects that are not fully resolved until the file is executed.)

If the *-c* option is specified, for all *pathname* operands of the form *file.c*, the files:

```
$(basename pathname .c).o
```

shall be created as the result of successful compilation. If the *-c* option is not specified, it is unspecified whether such *.o* files are created or deleted for the *file.c* operands.

If there are no options that prevent link editing (such as *-c* or *-E*), and all input files compile and link without error, the resulting executable file shall be written according to the *-o outfile* option (if present) or to the file *a.out*.

The executable file shall be created as specified in [Section 1.1.1.4](#) (on page 2318), except that the file permission bits shall be set to:

```
S_IRWXO | S_IRWXG | S_IRWXU
```

and the bits specified by the *umask* of the process shall be cleared.

**OPTIONS**

The *c99* utility shall conform to XBD [Section 12.2](#) (on page 216), except that:

- Options can be interspersed with operands.
- The order of specifying the *-L* and *-I* options, and the order of specifying *-l* options with respect to *pathname* operands is significant.
- Conforming applications shall specify each option separately; that is, grouping option letters (for example, *-cO*) need not be recognized by all implementations.

The following options shall be supported:

*-c* Suppress the link-edit phase of the compilation, and do not remove any object files that are produced.

*-D name[=value]*

Define *name* as if by a C-language **#define** directive. If no *=value* is given, a value of 1 shall be used. The *-D* option has lower precedence than the *-U* option. That is, if *name* is used in both a *-U* and a *-D* option, *name* shall be undefined regardless of the order of the options. Additional implementation-defined *names* may be provided by the compiler. Implementations shall support at least 2 048 bytes of *-D* definitions and 256 *names*.



81901	–E	Copy C-language source files to standard output, executing all preprocessor directives; no compilation shall be performed. If any operand is not a text file, the effects are unspecified.	
81902			
81903			
81904	–g	Produce symbolic information in the object or executable files; the nature of this information is unspecified, and may be modified by implementation-defined interactions with other options.	
81905			
81906			
81907	–I <i>directory</i>	Change the algorithm for searching for headers whose names are not absolute pathnames to look in the directory named by the <i>directory</i> pathname before looking in the usual places. Thus, headers whose names are enclosed in double-quotes (" ") shall be searched for first in the directory of the file with the <b>#include</b> line, then in directories named in –I options, and last in the usual places. For headers whose names are enclosed in angle brackets ("< >"), the header shall be searched for only in directories named in –I options and then in the usual places. Directories named in –I options shall be searched in the order specified. If the –I option is used to specify a directory that is one of the usual places searched by default, the results are unspecified. Implementations shall support at least ten instances of this option in a single <i>c99</i> command invocation.	
81908			
81909			
81910			
81911			
81912			
81913			
81914			
81915			
81916			
81917			
81918	–L <i>directory</i>	Change the algorithm of searching for the libraries named in the –I objects to look in the directory named by the <i>directory</i> pathname before looking in the usual places. Directories named in –L options shall be searched in the order specified. If the –L option is used to specify a directory that is one of the usual places searched by default, the results are unspecified. Implementations shall support at least ten instances of this option in a single <i>c99</i> command invocation. If a directory specified by a –L option contains files with names starting with any of the strings "libc.", "libl.", "libpthread.", "libm.", "librt.", "libtrace.", "libxnet.", or "liby.", the results are unspecified.	
81919			
81920			
81921			
81922			
81923			
81924			
81925			
81926			
81927	–l <i>library</i>	Search the library named <b>liblibrary.a</b> . A library shall be searched when its name is encountered, so the placement of a –l option is significant. Several standard libraries can be specified in this manner, as described in the EXTENDED DESCRIPTION section. Implementations may recognize implementation-defined suffixes other than <b>.a</b> as denoting libraries.	
81928			
81929			
81930			
81931			
81932	–O <i>optlevel</i>	Specify the level of code optimization. If the <i>optlevel</i> option-argument is the digit '0', all special code optimizations shall be disabled. If it is the digit '1', the nature of the optimization is unspecified. If the –O option is omitted, the nature of the system's default optimization is unspecified. It is unspecified whether code generated in the presence of the –O 0 option is the same as that generated when –O is omitted. Other <i>optlevel</i> values may be supported.	
81933			
81934			
81935			
81936			
81937			
81938	–o <i>outfile</i>	Use the pathname <i>outfile</i> , instead of the default <b>a.out</b> , for the executable file produced. If the –o option is present with –c or –E, the result is unspecified.	
81939			
81940	–s	Produce object or executable files, or both, from which symbolic and other information not required for proper execution using the <i>exec</i> family defined in the System Interfaces volume of POSIX.1-2008 has been removed (stripped). If both –g and –s options are present, the action taken is unspecified.	
81941			
81942			
81943			
81944	–U <i>name</i>	Remove any initial definition of <i>name</i> .	
81945		Multiple instances of the –D, –I, –L, –l, and –U options can be specified.	

**OPERANDS**

The application shall ensure that at least one *pathname* operand is specified. The following forms for *pathname* operands shall be supported:

*file.c* A C-language source file to be compiled and optionally linked. The application shall ensure that the operand is of this form if the `-c` option is used.

*file.a* A library of object files typically produced by the *ar* utility, and passed directly to the link editor. Implementations may recognize implementation-defined suffixes other than *.a* as denoting object file libraries.

*file.o* An object file produced by *c99 -c* and passed directly to the link editor. Implementations may recognize implementation-defined suffixes other than *.o* as denoting object files.

The processing of other files is implementation-defined.

**STDIN**

Not used.

**INPUT FILES**

Each input file shall be one of the following: a text file containing a C-language source program, an object file in the format produced by *c99 -c*, or a library of object files, in the format produced by archiving zero or more object files, using *ar*. Implementations may supply additional utilities that produce files in these formats. Additional input file formats are implementation-defined.

**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *c99*:

*LANG* Provide a default value for the internationalization variables that are unset or null. (See XBD Section 8.2 (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

*LC\_ALL* If set to a non-empty string value, override the values of all the other internationalization variables.

*LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files).

*LC\_MESSAGES*

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

*NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

*TMPDIR* Provide a pathname that should override the default directory for temporary files, if any. On XSI-conforming systems, provide a pathname that shall override the default directory for temporary files, if any.

**ASYNCHRONOUS EVENTS**

Default.

**STDOUT**

If more than one *pathname* operand ending in *.c* (or possibly other unspecified suffixes) is given, for each such file:

"%s:\n", *<pathname>*

may be written. These messages, if written, shall precede the processing of each input file; they

81989 shall not be written to the standard output if they are written to the standard error, as described  
81990 in the STDERR section.

81991 If the **-E** option is specified, the standard output shall be a text file that represents the results of  
81992 the preprocessing stage of the language; it may contain extra information appropriate for  
81993 subsequent compilation passes.

#### 81994 **STDERR**

81995 The standard error shall be used only for diagnostic messages. If more than one *pathname*  
81996 operand ending in **.c** (or possibly other unspecified suffixes) is given, for each such file:

81997 `"%s:\n", <pathname>`

81998 may be written to allow identification of the diagnostic and warning messages with the  
81999 appropriate input file. These messages, if written, shall precede the processing of each input file;  
82000 they shall not be written to the standard error if they are written to the standard output, as  
82001 described in the STDOUT section.

82002 This utility may produce warning messages about certain conditions that do not warrant  
82003 returning an error (non-zero) exit value.

#### 82004 **OUTPUT FILES**

82005 Object files or executable files or both are produced in unspecified formats. If the pathname of  
82006 an object file or executable file to be created by *c99* resolves to an existing directory entry for a  
82007 file that is not a regular file, it is unspecified whether *c99* shall attempt to create the file or shall  
82008 issue a diagnostic and exit with a non-zero exit status.

#### 82009 **EXTENDED DESCRIPTION**

##### 82010 **Standard Libraries**

82011 The *c99* utility shall recognize the following **-l** options for standard libraries:

82012 **-l c** This option shall make available all interfaces referenced in the System Interfaces  
82013 volume of POSIX.1-2008, with the possible exception of those interfaces listed as  
82014 residing in **<aio.h>**, **<arpa/inet.h>**, **<complex.h>**, **<fenv.h>**, **<math.h>**,  
82015 **<mqueue.h>**, **<netdb.h>**, **<net/if.h>**, **<netinet/in.h>**, **<pthread.h>**, **<sched.h>**,  
82016 **<semaphore.h>**, **<spawn.h>**, **<sys/socket.h>**, *pthread\_kill()*, and *pthread\_sigmask()*  
82017 in **<signal.h>**, **<trace.h>**, interfaces marked as optional in **<sys/mman.h>**,  
82018 interfaces marked as ADV (Advisory Information) in **<fcntl.h>**, and interfaces  
82019 beginning with the prefix *clock\_* or *timer\_* in **<time.h>**. This option shall not be  
82020 required to be present to cause a search of this library.

82021 **-l l** This option shall make available all interfaces required by the C-language output  
82022 of *lex* that are not made available through the **-l c** option.

82023 **-l pthread** This option shall make available all interfaces referenced in **<pthread.h>** and  
82024 *pthread\_kill()* and *pthread\_sigmask()* referenced in **<signal.h>**. An implementation  
82025 may search this library in the absence of this option.

82026 **-l m** This option shall make available all interfaces referenced in **<math.h>**,  
82027 **<complex.h>**, and **<fenv.h>**. An implementation may search this library in the  
82028 absence of this option.

82029 **-l rt** This option shall make available all interfaces referenced in **<aio.h>**, **<mqueue.h>**,  
82030 **<sched.h>**, **<semaphore.h>**, and **<spawn.h>**, interfaces marked as optional in  
82031 **<sys/mman.h>**, interfaces marked as ADV (Advisory Information) in **<fcntl.h>**,  
82032 and interfaces beginning with the prefix *clock\_* and *timer\_* in **<time.h>**. An

- 82033 implementation may search this library in the absence of this option.
- 82034 OB **-l trace** This option shall make available all interfaces referenced in `<trace.h>`. An  
82035 implementation may search this library in the absence of this option.
- 82036 **-l xnet** This option shall make available all interfaces referenced in `<arpa/inet.h>`,  
82037 `<netdb.h>`, `<net/if.h>`, `<netinet/in.h>`, and `<sys/socket.h>`. An implementation  
82038 may search this library in the absence of this option.
- 82039 **-l y** This option shall make available all interfaces required by the C-language output  
82040 of *yacc* that are not made available through the **-l c** option.
- 82041 In the absence of options that inhibit invocation of the link editor, such as **-c** or **-E**, the *c99* utility  
82042 shall cause the equivalent of a **-l c** option to be passed to the link editor after the last *pathname*  
82043 operand or **-l** option, causing it to be searched after all other object files and libraries are loaded.
- 82044 OB It is unspecified whether the libraries **libc.a**, **libl.a**, **libm.a**, **libpthread.a**, **librt.a**, **libtrace.a**,  
82045 **libxnet.a**, or **liby.a** exist as regular files. The implementation may accept as **-l** option-arguments  
82046 names of objects that do not exist as regular files.

## 82047 External Symbols

82048 The C compiler and link editor shall support the significance of external symbols up to a length  
82049 of at least 31 bytes; the action taken upon encountering symbols exceeding the implementation-  
82050 defined maximum symbol length is unspecified.

82051 The compiler and link editor shall support a minimum of 511 external symbols per source or  
82052 object file, and a minimum of 4095 external symbols in total. A diagnostic message shall be  
82053 written to the standard output if the implementation-defined limit is exceeded; other actions are  
82054 unspecified.

## 82055 Header Search

82056 If a file with the same name as one of the standard headers defined in XBD Chapter 13 (on page  
82057 219), not provided as part of the implementation, is placed in any of the usual places that are  
82058 searched by default for headers, the results are unspecified.

## 82059 Programming Environments

82060 All implementations shall support one of the following programming environments as a default.  
82061 Implementations may support more than one of the following programming environments.  
82062 Applications can use *sysconf()* or *getconf* to determine which programming environments are  
82063 supported.

Table 4-4 Programming Environments: Type Sizes

Programming Environment <i>getconf</i> Name	Bits in int	Bits in long	Bits in pointer	Bits in off_t
_POSIX_V7_ILP32_OFF32	32	32	32	32
_POSIX_V7_ILP32_OFFBIG	32	32	32	≥64
_POSIX_V7_LP64_OFF64	32	64	64	64
_POSIX_V7_LPBIG_OFFBIG	≥32	≥64	≥64	≥64

All implementations shall support one or more environments where the widths of the following types are no greater than the width of type **long**:

<b>blksize_t</b>	<b>ptrdiff_t</b>	<b>tcflag_t</b>
<b>cc_t</b>	<b>size_t</b>	<b>wchar_t</b>
<b>mode_t</b>	<b>speed_t</b>	<b>wint_t</b>
<b>nfds_t</b>	<b>ssize_t</b>	
<b>pid_t</b>	<b>suseconds_t</b>	

The executable files created when these environments are selected shall be in a proper format for execution by the *exec* family of functions. Each environment may be one of the ones in Table 4-4, or it may be another environment. The names for the environments that meet this requirement shall be output by a *getconf* command using the `_POSIX_V7_WIDTH_RESTRICTED_ENVS` argument, as a <newline>-separated list of names suitable for use with the *getconf* `-v` option. If more than one environment meets the requirement, the names of all such environments shall be output on separate lines. Any of these names can then be used in a subsequent *getconf* command to obtain the flags specific to that environment with the following suffixes added as appropriate:

`_CFLAGS` To get the C compiler flags.

`_LDFLAGS` To get the linker/loader flags.

`_LIBS` To get the libraries.

This requirement may be removed in a future version.

When this utility processes a file containing a function called *main()*, it shall be defined with a return type equivalent to **int**. Using return from the initial call to *main()* shall be equivalent (other than with respect to language scope issues) to calling *exit()* with the returned value. Reaching the end of the initial call to *main()* shall be equivalent to calling *exit(0)*. The implementation shall not declare a prototype for this function.

Implementations provide configuration strings for C compiler flags, linker/loader flags, and libraries for each supported environment. When an application needs to use a specific programming environment rather than the implementation default programming environment while compiling, the application shall first verify that the implementation supports the desired environment. If the desired programming environment is supported, the application shall then invoke *c99* with the appropriate C compiler flags as the first options for the compile, the appropriate linker/loader flags after any other options except `-l` but before any operands or `-l` options, and the appropriate libraries at the end of the operands and `-l` options.

Conforming applications shall not attempt to link together object files compiled for different programming models. Applications shall also be aware that binary data placed in shared memory or in files might not be recognized by applications built for other programming models.

82106

**Table 4-5** Programming Environments: c99 Arguments

Programming Environment <i>getconf</i> Name	Use	c99 Arguments <i>getconf</i> Name
_POSIX_V7_ILP32_OFF32	C Compiler Flags Linker/Loader Flags Libraries	POSIX_V7_ILP32_OFF32_CFLAGS POSIX_V7_ILP32_OFF32_LDFLAGS POSIX_V7_ILP32_OFF32_LIBS
_POSIX_V7_ILP32_OFFBIG	C Compiler Flags Linker/Loader Flags Libraries	POSIX_V7_ILP32_OFFBIG_CFLAGS POSIX_V7_ILP32_OFFBIG_LDFLAGS POSIX_V7_ILP32_OFFBIG_LIBS
_POSIX_V7_LP64_OFF64	C Compiler Flags Linker/Loader Flags Libraries	POSIX_V7_LP64_OFF64_CFLAGS POSIX_V7_LP64_OFF64_LDFLAGS POSIX_V7_LP64_OFF64_LIBS
_POSIX_V7_LPBIG_OFFBIG	C Compiler Flags Linker/Loader Flags Libraries	POSIX_V7_LPBIG_OFFBIG_CFLAGS POSIX_V7_LPBIG_OFFBIG_LDFLAGS POSIX_V7_LPBIG_OFFBIG_LIBS

In addition to the type size programming environments above, all implementations also support a multi-threaded programming environment that is orthogonal to all of the programming environments listed above. The *getconf* utility can be used to get flags for the threaded programming environment, as indicated in Table 4-6.

82125

**Table 4-6** Threaded Programming Environment: c99 Arguments

Programming Environment <i>getconf</i> Name	Use	c99 Arguments <i>getconf</i> Name
_POSIX_THREADS	C Compiler Flags Linker/Loader Flags	POSIX_V7_THREADS_CFLAGS POSIX_V7_THREADS_LDFLAGS

These programming environment flags may be used in conjunction with any of the type size programming environments supported by the implementation.

**EXIT STATUS**

The following exit values shall be returned:

0 Successful compilation or link edit.

>0 An error occurred.

**CONSEQUENCES OF ERRORS**

When c99 encounters a compilation error that causes an object file not to be created, it shall write a diagnostic to standard error and continue to compile other source code operands, but it shall not perform the link phase and it shall return a non-zero exit status. If the link edit is unsuccessful, a diagnostic message shall be written to standard error and c99 exits with a non-zero status. A conforming application shall rely on the exit status of c99, rather than on the existence or mode of the executable file.



## APPLICATION USAGE

Since the `c99` utility usually creates files in the current directory during the compilation process, it is typically necessary to run the `c99` utility in a directory in which a file can be created.

On systems providing POSIX Conformance (see XBD Chapter 2, on page 15), `c99` is required only with the C-Language Development option; XSI-conformant systems always provide `c99`.

Some historical implementations have created `.o` files when `-c` is not specified and more than one source file is given. Since this area is left unspecified, the application cannot rely on `.o` files being created, but it also must be prepared for any related `.o` files that already exist being deleted at the completion of the link edit.

There is the possible implication that if a user supplies versions of the standard functions (before they would be encountered by an implicit `-l c` or explicit `-l m`), that those versions would be used in place of the standard versions. There are various reasons this might not be true (functions defined as macros, manipulations for clean name space, and so on), so the existence of files named in the same manner as the standard libraries within the `-L` directories is explicitly stated to produce unspecified behavior.

All of the functions specified in the System Interfaces volume of POSIX.1-2008 may be made visible by implementations when the Standard C Library is searched. Conforming applications must explicitly request searching the other standard libraries when functions made visible by those libraries are used.

In the ISO C standard the mapping from physical source characters to the C source character set is implementation-defined. Implementations may strip white-space characters before the terminating `<newline>` of a (physical) line as part of this mapping and, as a consequence of this, one or more white-space characters (and no other characters) between a `<backslash>` character and the `<newline>` character that terminates the line produces implementation-defined results. Portable applications should not use such constructs.

Some `c99` compilers not conforming to POSIX.1-2008 do not support trigraphs by default.

## EXAMPLES

1. The following usage example compiles `foo.c` and creates the executable file `foo`:

```
c99 -o foo foo.c
```

The following usage example compiles `foo.c` and creates the object file `foo.o`:

```
c99 -c foo.c
```

The following usage example compiles `foo.c` and creates the executable file `a.out`:

```
c99 foo.c
```

The following usage example compiles `foo.c`, links it with `bar.o`, and creates the executable file `a.out`. It may also create and leave `foo.o`:

```
c99 foo.c bar.o
```

2. The following example shows how an application using threads interfaces can test for support of and use a programming environment supporting 32-bit `int`, `long`, and `pointer` types and an `off_t` type using at least 64 bits:

```
offbig_env=$(getconf _POSIX_V7_ILP32_OFFBIG)
if [ $offbig_env != "-1" ] && [ $offbig_env != "undefined" ]
then
    c99 $(getconf POSIX_V7_ILP32_OFFBIG_CFLAGS) \
```

```

82186      $(getconf POSIX_V7_THREADS_CFLAGS) -D_XOPEN_SOURCE=700 \
82187      $(getconf POSIX_V7_ILP32_OFFBIG_LDFLAGS) \
82188      $(getconf POSIX_V7_THREADS_LDFLAGS) foo.c -o foo \
82189      $(getconf POSIX_V7_ILP32_OFFBIG_LIBS) \
82190      -l pthread
82191 else
82192     echo ILP32_OFFBIG programming environment not supported
82193     exit 1
82194 fi

```

3. The following examples clarify the use and interactions of `-L` and `-l` options.

Consider the case in which module **a.c** calls function *f()* in library **libQ.a**, and module **b.c** calls function *g()* in library **libp.a**. Assume that both libraries reside in **/a/b/c**. The command line to compile and link in the desired way is:

```
c99 -L /a/b/c main.o a.c -l Q b.c -l p
```

In this case the `-L` option need only precede the first `-l` option, since both **libQ.a** and **libp.a** reside in the same directory.

Multiple `-L` options can be used when library name collisions occur. Building on the previous example, suppose that the user wants to use a new **libp.a**, in **/a/a/a**, but still wants *f()* from **/a/b/c/libQ.a**:

```
c99 -L /a/a/a -L /a/b/c main.o a.c -l Q b.c -l p
```

In this example, the linker searches the `-L` options in the order specified, and finds **/a/a/a/libp.a** before **/a/b/c/libp.a** when resolving references for **b.c**. The order of the `-l` options is still important, however.

4. The following example shows how an application can use a programming environment where the widths of the following types:

**blksize\_t, cc\_t, mode\_t, nfsd\_t, pid\_t, ptrdiff\_t, size\_t, speed\_t, ssize\_t, suseconds\_t, tflag\_t, wchar\_t, wint\_t**

are no greater than the width of type **long**:

```

82213 # First choose one of the listed environments ...
82214
82215 # ... if there are no additional constraints, the first one will do:
82216 CENV=$(getconf POSIX_V7_WIDTH_RESTRICTED_ENVS | head -n 1)
82217
82218 # ... or, if an environment that supports large files is preferred,
82219 # look for names that contain "OFF64" or "OFFBIG". (This chooses
82220 # the last one in the list if none match.)
82221 for CENV in $(getconf POSIX_V7_WIDTH_RESTRICTED_ENVS)
82222 do
82223     case $CENV in
82224         *OFF64*|*OFFBIG*) break ;;
82225     esac
82226 done
82227
82228 # The chosen environment name can now be used like this:
82229
82230 c99 $(getconf ${CENV}_CFLAGS) -D _POSIX_C_SOURCE=200809L \
82231     $(getconf ${CENV}_LDFLAGS) foo.c -o foo \
82232     $(getconf ${CENV}_LIBS)

```



## RATIONALE

The *c99* utility is based on the *c89* utility originally introduced in the ISO POSIX-2:1993 standard.

Some of the changes from *c89* include the ability to intersperse options and operands (which many *c89* implementations allowed despite it not being specified), the description of `-l` as an option instead of an operand, and the modification to the contents of the Standard Libraries section to account for new headers and options; for example, `<spawn.h>` added to the description of `-l rt`, and `-l trace` added for the Tracing option.

POSIX.1-2008 specifies that the *c99* utility must be able to use regular files for `*.o` files and for `a.out` files. Implementations are free to overwrite existing files of other types when attempting to create object files and executable files, but are not required to do so. If something other than a regular file is specified and using it fails for any reason, *c99* is required to issue a diagnostic message and exit with a non-zero exit status. But for some file types, the problem may not be noticed for a long time. For example, if a FIFO named `a.out` exists in the current directory, *c99* may attempt to open `a.out` and will hang in the `open()` call until another process opens the FIFO for reading. Then *c99* may write most of the `a.out` to the FIFO and fail when it tries to seek back close to the start of the file to insert a timestamp (FIFOs are not seekable files). The *c99* utility is also allowed to issue a diagnostic immediately if it encounters an `a.out` or `*.o` file that is not a regular file. For portable use, applications should ensure that any `a.out`, `-o` option-argument, or `*.o` files corresponding to any `*.c` files do not conflict with names already in use that are not regular files or symbolic links that point to regular files.

On many systems, multi-threaded applications run in a programming environment that is distinct from that used by single-threaded applications. This multi-threaded programming environment (in addition to needing to specify `-l pthread` at link time) may require additional flags to be set when headers are processed at compile time (`-D_REENTRANT` being common). This programming environment is orthogonal to the type size programming environments discussed above and listed in Table 4-4 (on page 2537). This version of the standard adds *getconf* utility calls to provide the C compiler flags and linker/loader flags needed to support multi-threaded applications. Note that on a system where single-threaded applications are a special case of a multi-threaded application, both of these *getconf* calls may return NULL strings; on other implementations both of these strings may be non-NULL strings.

The C standardization committee invented trigraphs (e.g., "??!" to represent ' | ') to address character portability problems in development environments based on national variants of the 7-bit ISO/IEC 646:1991 standard character set. However, these environments were already obsolete by the time the first ISO C standard was published, and in practice trigraphs have not been used for their intended purpose, and usually are intended to have their original meaning in K&R C. For example, in practice a C-language source string like "What??!" is usually intended to end in two <question-mark> characters and an <exclamation-mark>, not in ' | '.

When the `-E` option is used, execution of some `#pragma` preprocessor directives may simply result in a copy of the directive being included in the output as part of the allowed extra information used by subsequent compilation passes (see STDOUT).

## FUTURE DIRECTIONS

Unlike all of the other non-OB-shaded utilities in this standard, a utility by this name probably will not appear in the next version of this standard. This utility's name is tied to the current revision of the ISO C standard at the time this standard is approved. Since the ISO C standard and this standard are maintained by different organizations on different schedules, we cannot predict what the compiler will be named in the next version of the standard.

**SEE ALSO**

Section 1.1.1.4 (on page 2318), *ar*, *getconf*, *make*, *nm*, *strip*, *umask*

XBD Chapter 8 (on page 173), Section 12.2 (on page 216), Chapter 13 (on page 219)

XSH *exec*, *sysconf*()

**CHANGE HISTORY**

First released in Issue 6. Included for alignment with the ISO/IEC 9899:1999 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/12 is applied, correcting the EXTENDED DESCRIPTION of **-l c** and **-l m**. Previously, the text did not take into account the presence of the *c99* math headers.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/13 is applied, changing the reference to the **libxnet** library to **libxnet.a**.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/5 is applied, updating the OPTIONS section, so that the names of files contained in the directory specified by the **-L** option are not assumed to end in the **.a** suffix. The set of library prefixes is also updated.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/6 is applied, removing the lead underscore from the `POSIX_V6_WIDTH_RESTRICTED_ENVS` variable in the EXTENDED DESCRIPTION and the EXAMPLES sections.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #020 (SD5-XCU-ERN-10) is applied, adding to the OUTPUT FILES section and also adding associated RATIONALE.

Austin Group Interpretation 1003.1-2001 #095 is applied, clarifying the **-l library** operand.

Austin Group Interpretation 1003.1-2001 #166 is applied.

Austin Group Interpretation 1003.1-2001 #190 is applied, clarifying the handling of trailing white-space characters.

Austin Group Interpretation 1003.1-2001 #191 is applied, adding APPLICATION USAGE and RATIONALE regarding C-language trigraphs.

SD5-XCU-ERN-6 is applied, clarifying that Guideline 9 of the Utility Syntax Guidelines does not apply (options can be interspersed with operands).

SD5-XCU-ERN-11 is applied, adding the **<net/if.h>** header to the descriptions of **-l c** and **-l xnet**.

SD5-XCU-ERN-65 is applied, updating the EXAMPLES section.

SD5-XCU-ERN-67 and SD5-XCU-ERN-97 are applied, updating the SYNOPSIS.

SD5-XCU-ERN-133 is applied, updating the EXTENDED DESCRIPTION.

The *getconf* variables for the supported programming environments are updated to be V7.

The **-l trace** operand is marked obsolescent.

The *c99* reference page is rewritten to describe **-l** as an option rather than an operand.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0068 [129], XCU/TC1-2008/0069 [187], XCU/TC1-2008/0070 [187], XCU/TC1-2008/0071 [131], XCU/TC1-2008/0072 [187], and XCU/TC1-2008/0073 [364,430] are applied.

82316 **NAME**

82317 cal — print a calendar

82318 **SYNOPSIS**82319 XSI cal **[[month] year]**82320 **DESCRIPTION**

82321 The *cal* utility shall write a calendar to standard output using the Julian calendar for dates from  
 82322 January 1, 1 through September 2, 1752 and the Gregorian calendar for dates from September 14,  
 82323 1752 through December 31, 9999 as though the Gregorian calendar had been adopted on  
 82324 September 14, 1752.

82325 If no operands are given, *cal* shall produce a one-month calendar for the current month in the  
 82326 current year. If only the *year* operand is given, *cal* shall produce a calendar for all twelve months  
 82327 in the given calendar year. If both *month* and *year* operands are given, *cal* shall produce a one-  
 82328 month calendar for the given month in the given year.

82329 **OPTIONS**

82330 None.

82331 **OPERANDS**

82332 The following operands shall be supported:

82333 *month* Specify the month to be displayed, represented as a decimal integer from 1  
 82334 (January) to 12 (December).

82335 *year* Specify the year for which the calendar is displayed, represented as a decimal  
 82336 integer from 1 to 9999.

82337 **STDIN**

82338 Not used.

82339 **INPUT FILES**

82340 None.

82341 **ENVIRONMENT VARIABLES**82342 The following environment variables shall affect the execution of *cal*:

82343 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 82344 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 82345 variables used to determine the values of locale categories.)

82346 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 82347 internationalization variables.

82348 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 82349 characters (for example, single-byte as opposed to multi-byte characters in  
 82350 arguments).

82351 *LC\_MESSAGES*

82352 Determine the locale that should be used to affect the format and contents of  
 82353 diagnostic messages written to standard error, and informative messages written  
 82354 to standard output.

82355 *LC\_TIME* Determine the format and contents of the calendar.

82356 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

82357           TZ           Determine the timezone used to calculate the value of the current month.

82358 **ASYNCHRONOUS EVENTS**

82359           Default.

82360 **STDOUT**

82361           The standard output shall be used to display the calendar, in an unspecified format.

82362 **STDERR**

82363           The standard error shall be used only for diagnostic messages.

82364 **OUTPUT FILES**

82365           None.

82366 **EXTENDED DESCRIPTION**

82367           None.

82368 **EXIT STATUS**

82369           The following exit values shall be returned:

82370           0   Successful completion.

82371           >0  An error occurred.

82372 **CONSEQUENCES OF ERRORS**

82373           Default.

82374 **APPLICATION USAGE**

82375           Note that:

82376           cal 83

82377           refers to A.D. 83, not 1983.

82378 **EXAMPLES**

82379           None.

82380 **RATIONALE**

82381           Earlier versions of this standard incorrectly required that the command:

82382           cal 2000

82383           write a one-month calendar for the current calendar month (no matter what the current year is)  
82384           in the year 2000 to standard output. This did not match historic practice in any known version of  
82385           the *cal* utility. The description has been updated to match historic practice. When only the *year*  
82386           operand is given, *cal* writes a twelve-month calendar for the specified year.

82387 **FUTURE DIRECTIONS**

82388           A future version of this standard may support locale-specific recognition of the date of adoption  
82389           of the Gregorian calendar.

82390 **SEE ALSO**

82391           XBD [Chapter 8](#) (on page 173)

82392 **CHANGE HISTORY**

82393           First released in Issue 2.

82394 **Issue 6**

82395           The DESCRIPTION is updated to allow for traditional behavior for years before the adoption of  
82396           the Gregorian calendar.

82397 **Issue 7**

82398 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

82399 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0074 [56] and XCU/TC1-2008/0075  
82400 [56] are applied.

82401 **NAME**

82402       cat — concatenate and print files

82403 **SYNOPSIS**82404       cat [-u] [*file*...]82405 **DESCRIPTION**82406       The *cat* utility shall read files in sequence and shall write their contents to the standard output in  
82407       the same sequence.82408 **OPTIONS**82409       The *cat* utility shall conform to XBD [Section 12.2](#) (on page 216).

82410       The following option shall be supported:

82411       -u           Write bytes from the input file to the standard output without delay as each is  
82412       read.82413 **OPERANDS**

82414       The following operand shall be supported:

82415       *file*        A pathname of an input file. If no *file* operands are specified, the standard input  
82416       shall be used. If a *file* is '-', the *cat* utility shall read from the standard input at  
82417       that point in the sequence. The *cat* utility shall not close and reopen standard input  
82418       when it is referenced in this way, but shall accept multiple occurrences of '-' as a  
82419       *file* operand.82420 **STDIN**82421       The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '-'.  
82422       See the INPUT FILES section.82423 **INPUT FILES**

82424       The input files can be any file type.

82425 **ENVIRONMENT VARIABLES**82426       The following environment variables shall affect the execution of *cat*:82427       LANG        Provide a default value for the internationalization variables that are unset or null.  
82428       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
82429       variables used to determine the values of locale categories.)82430       LC\_ALL       If set to a non-empty string value, override the values of all the other  
82431       internationalization variables.82432       LC\_CTYPE     Determine the locale for the interpretation of sequences of bytes of text data as  
82433       characters (for example, single-byte as opposed to multi-byte characters in  
82434       arguments).

82435       LC\_MESSAGES

82436       Determine the locale that should be used to affect the format and contents of  
82437       diagnostic messages written to standard error.

82438 XSI       NLSPATH   Determine the location of message catalogs for the processing of LC\_MESSAGES.

82439 **ASYNCHRONOUS EVENTS**

82440       Default.

82441 **STDOUT**

82442 The standard output shall contain the sequence of bytes read from the input files. Nothing else  
 82443 shall be written to the standard output. If the standard output is a regular file, and is the same +  
 82444 file as any of the input file operands, the implementation may treat this as an error.

82445 **STDERR**

82446 The standard error shall be used only for diagnostic messages.

82447 **OUTPUT FILES**

82448 None.

82449 **EXTENDED DESCRIPTION**

82450 None.

82451 **EXIT STATUS**

82452 The following exit values shall be returned:

82453 0 All input files were output successfully.

82454 >0 An error occurred.

82455 **CONSEQUENCES OF ERRORS**

82456 Default.

82457 **APPLICATION USAGE**

82458 The **-u** option has value in prototyping non-blocking reads from FIFOs. The intent is to support  
 82459 the following sequence:

```
82460 mkfifo foo
82461 cat -u foo > /dev/ttyl3 &
82462 cat -u > foo
```

82463 It is unspecified whether standard output is or is not buffered in the default case. This is  
 82464 sometimes of interest when standard output is associated with a terminal, since buffering may  
 82465 delay the output. The presence of the **-u** option guarantees that unbuffered I/O is available. It is  
 82466 implementation-defined whether the *cat* utility buffers output if the **-u** option is not specified.  
 82467 Traditionally, the **-u** option is implemented using the equivalent of the *setvbuf()* function  
 82468 defined in the System Interfaces volume of POSIX.1-2008.

82469 **EXAMPLES**

82470 The following command:

```
82471 cat myfile
```

82472 writes the contents of the file **myfile** to standard output.

82473 The following command:

```
82474 cat doc1 doc2 > doc.all
```

82475 concatenates the files **doc1** and **doc2** and writes the result to **doc.all**.

82476 Because of the shell language mechanism used to perform output redirection, a command such  
 82477 as this:

```
82478 cat doc doc.end > doc
```

82479 causes the original data in **doc** to be lost before *cat* even begins execution. This is true whether |  
 82480 the *cat* command fails with an error or silently succeeds (the specification allows both |  
 82481 behaviors). In order to append the contents of **doc.end** without losing the original contents of |  
 82482 **doc**, this command should be used instead: |

82483 `cat doc.end >> doc`

82484 The command:

82485 `cat start - middle - end > file`

82486 when standard input is a terminal, gets two arbitrary pieces of input from the terminal with a  
82487 single invocation of *cat*. Note, however, that if standard input is a regular file, this would be  
82488 equivalent to the command:

82489 `cat start - middle /dev/null end > file`

82490 because the entire contents of the file would be consumed by *cat* the first time '-' was used as a  
82491 *file* operand and an end-of-file condition would be detected immediately when '-' was  
82492 referenced the second time.

#### 82493 RATIONALE

82494 Historical versions of the *cat* utility include the *-e*, *-t*, and *-v*, options which permit the ends of  
82495 lines, <tab> characters, and invisible characters, respectively, to be rendered visible in the  
82496 output. The standard developers omitted these options because they provide too fine a degree of  
82497 control over what is made visible, and similar output can be obtained using a command such as:

82498 `sed -n l pathname`

82499 The latter also has the advantage that its output is unambiguous, whereas the output of  
82500 historical *cat -etv* is not.

82501 The *-s* option was omitted because it corresponds to different functions in BSD and System  
82502 V-based systems. The BSD *-s* option to squeeze blank lines can be accomplished by the shell  
82503 script shown in the following example:

```
82504 sed -n '  
82505 # Write non-empty lines.  
82506 ./ {  
82507     p  
82508     d  
82509 }  
82510 # Write a single empty line, then look for more empty lines.  
82511 /^$/ p  
82512 # Get next line, discard the held <newline> (empty line),  
82513 # and look for more empty lines.  
82514 :Empty  
82515 /^$/ {  
82516     N  
82517     s/./.  
82518     b Empty  
82519 }  
82520 # Write the non-empty line before going back to search  
82521 # for the first in a set of empty lines.  
82522 p  
82523 '
```

82524 The System V *-s* option to silence error messages can be accomplished by redirecting the  
82525 standard error. Note that the BSD documentation for *cat* uses the term "blank line" to mean the  
82526 same as the POSIX "empty line": a line consisting only of a <newline>.

82527 The BSD *-n* option was omitted because similar functionality can be obtained from the *-n*  
82528 option of the *pr* utility.



82529 **FUTURE DIRECTIONS**

82530 None.

82531 **SEE ALSO**82532 *more*82533 XBD *Chapter 8* (on page 173), *Section 12.2* (on page 216)82534 XSH *setvbuf()*82535 **CHANGE HISTORY**

82536 First released in Issue 2.

82537 **Issue 7**

82538 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

82539 SD5-XCU-ERN-174 is applied, changing the RATIONALE.

**NAME**

cd — change the working directory

**SYNOPSIS**

cd [-L|-P] [*directory*]

cd -

**DESCRIPTION**

The *cd* utility shall change the working directory of the current shell execution environment (see [Section 2.12](#), on page 2371) by executing the following steps in sequence. (In the following steps, the symbol **curpath** represents an intermediate value used to simplify the description of the algorithm used by *cd*. There is no requirement that **curpath** be made visible to the application.)

1. If no *directory* operand is given and the *HOME* environment variable is empty or undefined, the default behavior is implementation-defined and no further steps shall be taken.
2. If no *directory* operand is given and the *HOME* environment variable is set to a non-empty value, the *cd* utility shall behave as if the *directory* named in the *HOME* environment variable was specified as the *directory* operand.
3. If the *directory* operand begins with a <slash> character, set **curpath** to the operand and proceed to step 7.
4. If the first component of the *directory* operand is dot or dot-dot, proceed to step 6.
5. Starting with the first pathname in the <colon>-separated pathnames of *CDPATH* (see the ENVIRONMENT VARIABLES section) if the pathname is non-null, test if the concatenation of that pathname, a <slash> character if that pathname did not end with a <slash> character, and the *directory* operand names a directory. If the pathname is null, test if the concatenation of dot, a <slash> character, and the operand names a directory. In either case, if the resulting string names an existing directory, set **curpath** to that string and proceed to step 7. Otherwise, repeat this step with the next pathname in *CDPATH* until all pathnames have been tested.
6. Set **curpath** to the *directory* operand.
7. If the *-P* option is in effect, proceed to step 10. If **curpath** does not begin with a <slash> character, set **curpath** to the string formed by the concatenation of the value of *PWD*, a <slash> character if the value of *PWD* did not end with a <slash> character, and **curpath**.
8. The **curpath** value shall then be converted to canonical form as follows, considering each component from beginning to end, in sequence:
  - a. Dot components and any <slash> characters that separate them from the next component shall be deleted.
  - b. For each dot-dot component, if there is a preceding component and it is neither root nor dot-dot, then:
    - i. If the preceding component does not refer (in the context of pathname resolution with symbolic links followed) to a directory, then the *cd* utility shall display an appropriate error message and no further steps shall be taken.
    - ii. The preceding component, all <slash> characters separating the preceding component from dot-dot, dot-dot, and all <slash> characters separating dot-dot from the following component (if any) shall be deleted.

- c. An implementation may further simplify **curpath** by removing any trailing <slash> characters that are not also leading <slash> characters, replacing multiple non-leading consecutive <slash> characters with a single <slash>, and replacing three or more leading <slash> characters with a single <slash>. If, as a result of this canonicalization, the **curpath** variable is null, no further steps shall be taken.
9. If **curpath** is longer than {PATH\_MAX} bytes (including the terminating null) and the *directory* operand was not longer than {PATH\_MAX} bytes (including the terminating null), then **curpath** shall be converted from an absolute pathname to an equivalent relative pathname if possible. This conversion shall always be considered possible if the value of *PWD*, with a trailing <slash> added if it does not already have one, is an initial substring of **curpath**. Whether or not it is considered possible under other circumstances is unspecified. Implementations may also apply this conversion if **curpath** is not longer than {PATH\_MAX} bytes or the *directory* operand was longer than {PATH\_MAX} bytes.
10. The *cd* utility shall then perform actions equivalent to the *chdir()* function called with **curpath** as the *path* argument. If these actions fail for any reason, the *cd* utility shall display an appropriate error message and the remainder of this step shall not be executed. If the **-P** option is not in effect, the *PWD* environment variable shall be set to the value that **curpath** had on entry to step 9 (i.e., before conversion to a relative pathname). If the **-P** option is in effect, the *PWD* environment variable shall be set to the string that would be output by *pwd -P*. If there is insufficient permission on the new directory, or on any parent of that directory, to determine the current working directory, the value of the *PWD* environment variable is unspecified.

If, during the execution of the above steps, the *PWD* environment variable is set, the *OLDPWD* environment variable shall also be set to the value of the old working directory (that is the current working directory immediately prior to the call to *cd*).

## OPTIONS

The *cd* utility shall conform to XBD [Section 12.2](#) (on page 216).

The following options shall be supported by the implementation:

- L** Handle the operand dot-dot logically; symbolic link components shall not be resolved before dot-dot components are processed (see steps 8. and 9. in the DESCRIPTION).
- P** Handle the operand dot-dot physically; symbolic link components shall be resolved before dot-dot components are processed (see step 7. in the DESCRIPTION).

If both **-L** and **-P** options are specified, the last of these options shall be used and all others ignored. If neither **-L** nor **-P** is specified, the operand shall be handled dot-dot logically; see the DESCRIPTION.

## OPERANDS

The following operands shall be supported:

- directory* An absolute or relative pathname of the directory that shall become the new working directory. The interpretation of a relative pathname by *cd* depends on the **-L** option and the *CDPATH* and *PWD* environment variables. If *directory* is an empty string, the results are unspecified.
- When a <hyphen-minus> is used as the operand, this shall be equivalent to the command:
- ```
cd "$OLDPWD" && pwd
```

82630 which changes to the previous working directory and then writes its name.

#### 82631 **STDIN**

82632 Not used.

#### 82633 **INPUT FILES**

82634 None.

#### 82635 **ENVIRONMENT VARIABLES**

82636 The following environment variables shall affect the execution of *cd*:

82637 *CDPATH* A <colon>-separated list of pathnames that refer to directories. The *cd* utility shall  
82638 use this list in its attempt to change the directory, as described in the  
82639 DESCRIPTION. An empty string in place of a directory pathname represents the  
82640 current directory. If *CDPATH* is not set, it shall be treated as if it were an empty  
82641 string.

82642 *HOME* The name of the directory, used when no *directory* operand is specified.

82643 *LANG* Provide a default value for the internationalization variables that are unset or null.  
82644 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
82645 variables used to determine the values of locale categories.)

82646 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
82647 internationalization variables.

82648 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
82649 characters (for example, single-byte as opposed to multi-byte characters in  
82650 arguments).

82651 *LC\_MESSAGES*

82652 Determine the locale that should be used to affect the format and contents of  
82653 diagnostic messages written to standard error.

82654 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

82655 *OLDPWD* A pathname of the previous working directory, used by *cd -*.

82656 *PWD* This variable shall be set as specified in the DESCRIPTION. If an application sets  
82657 or unsets the value of *PWD*, the behavior of *cd* is unspecified.

#### 82658 **ASYNCHRONOUS EVENTS**

82659 Default.

#### 82660 **STDOUT**

82661 If a non-empty directory name from *CDPATH* is used, or if *cd -* is used, an absolute pathname of  
82662 the new working directory shall be written to the standard output as follows:

82663 "%s\n", <new directory>

82664 Otherwise, there shall be no output.

#### 82665 **STDERR**

82666 The standard error shall be used only for diagnostic messages.

#### 82667 **OUTPUT FILES**

82668 None.

82669 **EXTENDED DESCRIPTION**

82670 None.

82671 **EXIT STATUS**

82672 The following exit values shall be returned:

82673 0 The directory was successfully changed.

82674 &gt;0 An error occurred.

82675 **CONSEQUENCES OF ERRORS**

82676 The working directory shall remain unchanged.

82677 **APPLICATION USAGE**82678 Since *cd* affects the current shell execution environment, it is always provided as a shell regular  
82679 built-in. If it is called in a subshell or separate utility execution environment, such as one of the  
82680 following:82681 (cd /tmp)  
82682 nohup cd  
82683 find . -exec cd {} \;

82684 it does not affect the working directory of the caller's environment.

82685 The user must have execute (search) permission in *directory* in order to change to it.82686 **EXAMPLES**82687 The following template can be used to perform processing in the directory specified by *location*  
82688 and end up in the current working directory in use before the first *cd* command was issued:82689 cd *location*  
82690 if [ \$? -ne 0 ]  
82691 then  
82692 print error message  
82693 exit 1  
82694 fi  
82695 ... do whatever is desired as long as the OLDPWD environment variable  
82696 is not modified  
82697 cd -82698 **RATIONALE**82699 The use of the *CDPATH* was introduced in the System V shell. Its use is analogous to the use of  
82700 the *PATH* variable in the shell. The BSD C shell used a shell parameter *cdpath* for this purpose.82701 A common extension when *HOME* is undefined is to get the login directory from the user  
82702 database for the invoking user. This does not occur on System V implementations.82703 Some historical shells, such as the KornShell, took special actions when the directory name  
82704 contained a dot-dot component, selecting the logical parent of the directory, rather than the  
82705 actual parent directory; that is, it moved up one level toward the '/' in the pathname,  
82706 remembering what the user typed, rather than performing the equivalent of:

82707 chdir("../");

82708 In such a shell, the following commands would not necessarily produce equivalent output for all  
82709 directories:

82710 cd .. &amp;&amp; ls ls ..

82711 This behavior is now the default. It is not consistent with the definition of dot-dot in most

82712 historical practice; that is, while this behavior has been optionally available in the KornShell,  
82713 other shells have historically not supported this functionality. The logical pathname is stored in  
82714 the *PWD* environment variable when the *cd* utility completes and this value is used to construct  
82715 the next directory name if *cd* is invoked with the *-L* option.

#### 82716 FUTURE DIRECTIONS

82717 None.

#### 82718 SEE ALSO

82719 [Section 2.12](#) (on page 2371), *pwd*

82720 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

82721 XSH *chdir()*

#### 82722 CHANGE HISTORY

82723 First released in Issue 2.

#### 82724 Issue 6

82725 The following new requirements on POSIX implementations derive from alignment with the  
82726 Single UNIX Specification:

- 82727 • The *cd* *-* operand, *PWD*, and *OLDPWD* are added.

82728 The *-L* and *-P* options are added to align with the IEEE P1003.2b draft standard. This also  
82729 includes the introduction of a new description to include the effect of these options.

82730 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/14 is applied, changing the SYNOPSIS to  
82731 make it clear that the *-L* and *-P* options are mutually-exclusive.

#### 82732 Issue 7

82733 Austin Group Interpretation 1003.1-2001 #037 is applied.

82734 Austin Group Interpretation 1003.1-2001 #199 is applied, clarifying how the *cd* utility handles  
82735 concatenation of two pathnames when the first pathname ends in a <slash> character.

82736 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

82737 Step 7 of the processing performed by *cd* is revised to refer to **curpath** instead of “the operand”.

82738 Changes to the *pwd* utility and *PWD* environment variable have been made to match the  
82739 changes to the *getcwd()* function made for Austin Group Interpretation 1003.1-2001 #140.

82740 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0076 [230], XCU/TC1-2008/0077  
82741 [240], XCU/TC1-2008/0078 [240], and XCU/TC1-2008/0079 [123] are applied.

82742 **NAME**

82743 cflow — generate a C-language flowgraph (DEVELOPMENT)

82744 **SYNOPSIS**

```
82745 XSI cflow [-r] [-d num] [-D name[=def]]... [-i incl] [-I dir]...
82746 [-U dir]... file...
```

82747 **DESCRIPTION**

82748 The *cflow* utility shall analyze a collection of object files or assembler, C-language, *lex*, or *yacc*  
 82749 source files, and attempt to build a graph, written to standard output, charting the external  
 82750 references.

82751 **OPTIONS**

82752 The *cflow* utility shall conform to XBD [Section 12.2](#) (on page 216), except that the order of the **-D**,  
 82753 **-I**, and **-U** options (which are identical to their interpretation by *c99*) is significant.

82754 The following options shall be supported:

- 82755 **-d num** Indicate the depth at which the flowgraph is cut off. The application shall ensure  
 82756 that the argument *num* is a decimal integer. By default this is a very large number  
 82757 (typically greater than 32 000). Attempts to set the cut-off depth to a non-positive  
 82758 integer shall be ignored.
- 82759 **-i incl** Increase the number of included symbols. The *incl* option-argument is one of the  
 82760 following characters:
- 82761 *x* Include external and static data symbols. The default shall be to include only  
 82762 functions in the flowgraph.
- 82763 *\_* (Underscore) Include names that begin with an <underscore>. The default  
 82764 shall be to exclude these functions (and data if **-i x** is used).
- 82765 **-r** Reverse the caller: callee relationship, producing an inverted listing showing the  
 82766 callers of each function. The listing shall also be sorted in lexicographical order by  
 82767 callee.

82768 **OPERANDS**

82769 The following operand is supported:

- 82770 *file* The pathname of a file for which a graph is to be generated. Filenames suffixed by  
 82771 **.l** shall be taken to be *lex* input, **.y** as *yacc* input, **.c** as *c99* input, and **.i** as the  
 82772 output of *c99* **-E**. Such files shall be processed as appropriate, determined by their  
 82773 suffix.
- 82774 Files suffixed by **.s** (conventionally assembler source) may have more limited  
 82775 information extracted from them.

82776 **STDIN**

82777 Not used.

82778 **INPUT FILES**

82779 The input files shall be object files or assembler, C-language, *lex*, or *yacc* source files.

82780 **ENVIRONMENT VARIABLES**

82781 The following environment variables shall affect the execution of *cflow*:

- 82782 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 82783 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 82784 variables used to determine the values of locale categories.)



82785 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
82786 internationalization variables.

82787 *LC\_COLLATE*  
82788 Determine the locale for the ordering of the output when the *-r* option is used.

82789 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
82790 characters (for example, single-byte as opposed to multi-byte characters in  
82791 arguments and input files).

82792 *LC\_MESSAGES*  
82793 Determine the locale that should be used to affect the format and contents of  
82794 diagnostic messages written to standard error.

82795 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 82796 **ASYNCHRONOUS EVENTS**

82797 Default.

## 82798 **STDOUT**

82799 The flowgraph written to standard output shall be formatted as follows:

82800 "%d %s:%s\n", <reference number>, <global>, <definition>

82801 Each line of output begins with a reference (that is, line) number, followed by indentation of at  
82802 least one column position per level. This is followed by the name of the global, a <colon>, and  
82803 its definition. Normally globals are only functions not defined as an external or beginning with  
82804 an <underscore>; see the *OPTIONS* section for the *-i* inclusion option. For information extracted  
82805 from C-language source, the definition consists of an abstract type declaration (for example, **char**  
82806 \*) and, delimited by angle brackets, the name of the source file and the line number where the  
82807 definition was found. Definitions extracted from object files indicate the filename and location  
82808 counter under which the symbol appeared (for example, *text*).

82809 Once a definition of a name has been written, subsequent references to that name contain only  
82810 the reference number of the line where the definition can be found. For undefined references,  
82811 only "< >" shall be written.

## 82812 **STDERR**

82813 The standard error shall be used only for diagnostic messages.

## 82814 **OUTPUT FILES**

82815 None.

## 82816 **EXTENDED DESCRIPTION**

82817 None.

## 82818 **EXIT STATUS**

82819 The following exit values shall be returned:

82820 0 Successful completion.

82821 >0 An error occurred.

## 82822 **CONSEQUENCES OF ERRORS**

82823 Default.



**APPLICATION USAGE**

Files produced by *lex* and *yacc* cause the reordering of line number declarations, and this can confuse *cflow*. To obtain proper results, the input of *yacc* or *lex* must be directed to *cflow*.

**EXAMPLES**

Given the following in **file.c**:

```
int i;
int f();
int g();
int h();
int
main()
{
    f();
    g();
    f();
}
int
f()
{
    i = h();
}
```

The command:

```
cflow -i x file.c
```

produces the output:

```
1 main: int(), <file.c 6>
2   f: int(), <file.c 13>
3     h: <>
4     i: int, <file.c 1>
5   g: <>
```

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*c99*, *lex*, *yacc*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

**CHANGE HISTORY**

First released in Issue 2.

**Issue 6**

The normative text is reworded to avoid use of the term “must” for application requirements.

**Issue 7**

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

**NAME**

`chgrp` — change the file group ownership

**SYNOPSIS**

`chgrp [-h] group file...`

`chgrp -R [-H|-L|-P] group file...`

**DESCRIPTION**

The `chgrp` utility shall set the group ID of the file named by each *file* operand to the group ID specified by the *group* operand.

For each *file* operand, or, if the `-R` option is used, each file encountered while walking the directory trees specified by the *file* operands, the `chgrp` utility shall perform actions equivalent to the `chown()` function defined in the System Interfaces volume of POSIX.1-2008, called with the following arguments:

- The *file* operand shall be used as the *path* argument.
- The user ID of the file shall be used as the *owner* argument.
- The specified group ID shall be used as the *group* argument.

Unless `chgrp` is invoked by a process with appropriate privileges, the set-user-ID and set-group-ID bits of a regular file shall be cleared upon successful completion; the set-user-ID and set-group-ID bits of other file types may be cleared.

**OPTIONS**

The `chgrp` utility shall conform to XBD [Section 12.2](#) (on page 216).

The following options shall be supported by the implementation:

- h** For each *file* operand that names a file of type symbolic link, `chgrp` shall attempt to set the group ID of the symbolic link instead of the file referenced by the symbolic link.
- H** If the `-R` option is specified and a symbolic link referencing a file of type directory is specified on the command line, `chgrp` shall change the group of the directory referenced by the symbolic link and all files in the file hierarchy below it.
- L** If the `-R` option is specified and a symbolic link referencing a file of type directory is specified on the command line or encountered during the traversal of a file hierarchy, `chgrp` shall change the group of the directory referenced by the symbolic link and all files in the file hierarchy below it.
- P** If the `-R` option is specified and a symbolic link is specified on the command line or encountered during the traversal of a file hierarchy, `chgrp` shall change the group ID of the symbolic link. The `chgrp` utility shall not follow the symbolic link to any other part of the file hierarchy.
- R** Recursively change file group IDs. For each *file* operand that names a directory, `chgrp` shall change the group of the directory and all files in the file hierarchy below it. Unless a `-H`, `-L`, or `-P` option is specified, it is unspecified which of these options will be used as the default.

Specifying more than one of the mutually-exclusive options `-H`, `-L`, and `-P` shall not be considered an error. The last option specified shall determine the behavior of the utility.

**82907 OPERANDS**

82908 The following operands shall be supported:

82909 *group* A group name from the group database or a numeric group ID. Either specifies a  
 82910 group ID to be given to each file named by one of the *file* operands. If a numeric  
 82911 *group* operand exists in the group database as a group name, the group ID number  
 82912 associated with that group name is used as the group ID.

82913 *file* A pathname of a file whose group ID is to be modified.

**82914 STDIN**

82915 Not used.

**82916 INPUT FILES**

82917 None.

**82918 ENVIRONMENT VARIABLES**

82919 The following environment variables shall affect the execution of *chgrp*:

82920 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 82921 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 82922 variables used to determine the values of locale categories.)

82923 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 82924 internationalization variables.

82925 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 82926 characters (for example, single-byte as opposed to multi-byte characters in  
 82927 arguments).

82928 *LC\_MESSAGES*

82929 Determine the locale that should be used to affect the format and contents of  
 82930 diagnostic messages written to standard error.

82931 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

**82932 ASYNCHRONOUS EVENTS**

82933 Default.

**82934 STDOUT**

82935 Not used.

**82936 STDERR**

82937 The standard error shall be used only for diagnostic messages.

**82938 OUTPUT FILES**

82939 None.

**82940 EXTENDED DESCRIPTION**

82941 None.

**82942 EXIT STATUS**

82943 The following exit values shall be returned:

82944 0 The utility executed successfully and all requested changes were made.

82945 >0 An error occurred.

**CONSEQUENCES OF ERRORS**

Default.

**APPLICATION USAGE**

Only the owner of a file or the user with appropriate privileges may change the owner or group of a file.

Some implementations restrict the use of *chgrp* to a user with appropriate privileges when the *group* specified is not the effective group ID or one of the supplementary group IDs of the calling process.

**EXAMPLES**

None.

**RATIONALE**

The System V and BSD versions use different exit status codes. Some implementations used the exit status as a count of the number of errors that occurred; this practice is unworkable since it can overflow the range of valid exit status values. The standard developers chose to mask these by specifying only 0 and >0 as exit values.

The functionality of *chgrp* is described substantially through references to *chown()*. In this way, there is no duplication of effort required for describing the interactions of permissions, multiple groups, and so on.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*chmod*, *chown*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

XSH *chown()*

**CHANGE HISTORY**

First released in Issue 2.

**Issue 6**

New options **-H**, **-L**, and **-P** are added to align with the IEEE P1003.2b draft standard. These options affect the processing of symbolic links.

IEEE PASC Interpretation 1003.2 #172 is applied, changing the CONSEQUENCES OF ERRORS section to "Default."

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/15 is applied, changing the SYNOPSIS to make it clear that **-h** and **-R** are optional.

**Issue 7**

SD5-XCU-ERN-8 is applied, removing the **-R** from the first line of the SYNOPSIS.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0080 [237,341] is applied.

82983 **NAME**

82984       chmod — change the file modes

82985 **SYNOPSIS**82986       chmod [-R] *mode file...*82987 **DESCRIPTION**82988       The *chmod* utility shall change any or all of the file mode bits of the file named by each *file*  
82989       operand in the way specified by the *mode* operand.82990       It is implementation-defined whether and how the *chmod* utility affects any alternate or  
82991       additional file access control mechanism (see XBD [Section 4.5](#), on page 108) being used for the  
82992       specified file.82993       Only a process whose effective user ID matches the user ID of the file, or a process with  
82994       appropriate privileges, shall be permitted to change the file mode bits of a file.82995       Upon successfully changing the file mode bits of a file, the *chmod* utility shall mark for update  
82996       the last file status change timestamp of the file.82997 **OPTIONS**82998       The *chmod* utility shall conform to XBD [Section 12.2](#) (on page 216).

82999       The following option shall be supported:

83000       **-R**       Recursively change file mode bits. For each *file* operand that names a directory,  
83001       *chmod* shall change the file mode bits of the directory and all files in the file  
83002       hierarchy below it.83003 **OPERANDS**

83004       The following operands shall be supported:

83005       *mode*       Represents the change to be made to the file mode bits of each file named by one of  
83006       the *file* operands; see the EXTENDED DESCRIPTION section.83007       *file*       A pathname of a file whose file mode bits shall be modified.83008 **STDIN**

83009       Not used.

83010 **INPUT FILES**

83011       None.

83012 **ENVIRONMENT VARIABLES**83013       The following environment variables shall affect the execution of *chmod*:83014       **LANG**       Provide a default value for the internationalization variables that are unset or null.  
83015       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
83016       variables used to determine the values of locale categories.)83017       **LC\_ALL**       If set to a non-empty string value, override the values of all the other  
83018       internationalization variables.83019       **LC\_CTYPE**   Determine the locale for the interpretation of sequences of bytes of text data as  
83020       characters (for example, single-byte as opposed to multi-byte characters in  
83021       arguments).83022       **LC\_MESSAGES**83023       Determine the locale that should be used to affect the format and contents of  
83024       diagnostic messages written to standard error.

|       |                             |                |                                                                                                                                                        |
|-------|-----------------------------|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| 83025 | XSI                         | <b>NLSPATH</b> | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                  |
| 83026 | <b>ASYNCHRONOUS EVENTS</b>  |                |                                                                                                                                                        |
| 83027 |                             |                | Default.                                                                                                                                               |
| 83028 | <b>STDOUT</b>               |                |                                                                                                                                                        |
| 83029 |                             |                | Not used.                                                                                                                                              |
| 83030 | <b>STDERR</b>               |                |                                                                                                                                                        |
| 83031 |                             |                | The standard error shall be used only for diagnostic messages.                                                                                         |
| 83032 | <b>OUTPUT FILES</b>         |                |                                                                                                                                                        |
| 83033 |                             |                | None.                                                                                                                                                  |
| 83034 | <b>EXTENDED DESCRIPTION</b> |                |                                                                                                                                                        |
| 83035 |                             |                | The <i>mode</i> operand shall be either a <i>symbolic_mode</i> expression or a non-negative octal integer. The                                         |
| 83036 |                             |                | <i>symbolic_mode</i> form is described by the grammar later in this section.                                                                           |
| 83037 |                             |                | Each <b>clause</b> shall specify an operation to be performed on the current file mode bits of each <i>file</i> .                                      |
| 83038 |                             |                | The operations shall be performed on each <i>file</i> in the order in which the <b>clauses</b> are specified.                                          |
| 83039 |                             |                | The <b>who</b> symbols <b>u</b> , <b>g</b> , and <b>o</b> shall specify the <i>user</i> , <i>group</i> , and <i>other</i> parts of the file mode bits, |
| 83040 |                             |                | respectively. A <b>who</b> consisting of the symbol <b>a</b> shall be equivalent to <b>ugo</b> .                                                       |
| 83041 |                             |                | The <b>perm</b> symbols <b>r</b> , <b>w</b> , and <b>x</b> represent the <i>read</i> , <i>write</i> , and <i>execute/search</i> portions of file mode  |
| 83042 |                             |                | bits, respectively. The <b>perm</b> symbol <b>s</b> shall represent the <i>set-user-ID-on-execution</i> (when <b>who</b>                               |
| 83043 |                             |                | contains or implies <b>u</b> ) and <i>set-group-ID-on-execution</i> (when <b>who</b> contains or implies <b>g</b> ) bits.                              |
| 83044 |                             |                | The <b>perm</b> symbol <b>X</b> shall represent the execute/search portion of the file mode bits if the file is a                                      |
| 83045 |                             |                | directory or if the current (unmodified) file mode bits have at least one of the execute bits                                                          |
| 83046 |                             |                | ( <i>S_IXUSR</i> , <i>S_IXGRP</i> , or <i>S_IXOTH</i> ) set. It shall be ignored if the file is not a directory and none of                            |
| 83047 |                             |                | the execute bits are set in the current file mode bits.                                                                                                |
| 83048 |                             |                | The <b>permcop</b> symbols <b>u</b> , <b>g</b> , and <b>o</b> shall represent the current permissions associated with the                              |
| 83049 |                             |                | user, group, and other parts of the file mode bits, respectively. For the remainder of this section,                                                   |
| 83050 |                             |                | <b>perm</b> refers to the non-terminals <b>perm</b> and <b>permcop</b> in the grammar.                                                                 |
| 83051 |                             |                | If multiple <b>actionlists</b> are grouped with a single <b>wholist</b> in the grammar, each <b>actionlist</b> shall be                                |
| 83052 |                             |                | applied in the order specified with that <b>wholist</b> . The <i>op</i> symbols shall represent the operation                                          |
| 83053 |                             |                | performed, as follows:                                                                                                                                 |
| 83054 |                             | +              | If <b>perm</b> is not specified, the '+' operation shall not change the file mode bits.                                                                |
| 83055 |                             |                | If <b>who</b> is not specified, the file mode bits represented by <b>perm</b> for the owner, group, and                                                |
| 83056 |                             |                | other permissions, except for those with corresponding bits in the file mode creation mask                                                             |
| 83057 |                             |                | of the invoking process, shall be set.                                                                                                                 |
| 83058 |                             |                | Otherwise, the file mode bits represented by the specified <b>who</b> and <b>perm</b> values shall be set.                                             |
| 83059 |                             | –              | If <b>perm</b> is not specified, the '–' operation shall not change the file mode bits.                                                                |
| 83060 |                             |                | If <b>who</b> is not specified, the file mode bits represented by <b>perm</b> for the owner, group, and                                                |
| 83061 |                             |                | other permissions, except for those with corresponding bits in the file mode creation mask                                                             |
| 83062 |                             |                | of the invoking process, shall be cleared.                                                                                                             |
| 83063 |                             |                | Otherwise, the file mode bits represented by the specified <b>who</b> and <b>perm</b> values shall be                                                  |
| 83064 |                             |                | cleared.                                                                                                                                               |
| 83065 |                             | =              | Clear the file mode bits specified by the <b>who</b> value, or, if no <b>who</b> value is specified, all of the                                        |
| 83066 |                             |                | file mode bits specified in this volume of POSIX.1-2008.                                                                                               |

If **perm** is not specified, the '=' operation shall make no further modifications to the file mode bits.

If **who** is not specified, the file mode bits represented by **perm** for the owner, group, and other permissions, except for those with corresponding bits in the file mode creation mask of the invoking process, shall be set.

Otherwise, the file mode bits represented by the specified **who** and **perm** values shall be set.

When using the symbolic mode form on a regular file, it is implementation-defined whether or not:

- Requests to set the set-user-ID-on-execution or set-group-ID-on-execution bit when all execute bits are currently clear and none are being set are ignored.
- Requests to clear all execute bits also clear the set-user-ID-on-execution and set-group-ID-on-execution bits.
- Requests to clear the set-user-ID-on-execution or set-group-ID-on-execution bits when all execute bits are currently clear are ignored. However, if the command *ls -l file* writes an *s* in the position indicating that the set-user-ID-on-execution or set-group-ID-on-execution is set, the commands *chmod u-s file* or *chmod g-s file*, respectively, shall not be ignored.

When using the symbolic mode form on other file types, it is implementation-defined whether or not requests to set or clear the set-user-ID-on-execution or set-group-ID-on-execution bits are honored.

If the **who** symbol **o** is used in conjunction with the **perm** symbol **s** with no other **who** symbols being specified, the set-user-ID-on-execution and set-group-ID-on-execution bits shall not be modified. It shall not be an error to specify the **who** symbol **o** in conjunction with the **perm** symbol **s**.

The **perm** symbol **t** shall specify the S\_ISVTX bit. When used with a file of type directory, it can be used with the **who** symbol **a**, or with no **who** symbol. It shall not be an error to specify a **who** symbol of **u**, **g**, or **o** in conjunction with the **perm** symbol **t**, but the meaning of these combinations is unspecified. The effect when using the **perm** symbol **t** with any file type other than directory is unspecified.

For an octal integer *mode* operand, the file mode bits shall be set absolutely.

For each bit set in the octal number, the corresponding file permission bit shown in the following table shall be set; all other file permission bits shall be cleared. For regular files, for each bit set in the octal number corresponding to the set-user-ID-on-execution or the set-group-ID-on-execution, bits shown in the following table shall be set; if these bits are not set in the octal number, they are cleared. For other file types, it is implementation-defined whether or not requests to set or clear the set-user-ID-on-execution or set-group-ID-on-execution bits are honored.

| Octal | Mode Bit | Octal | Mode Bit | Octal | Mode Bit | Octal | Mode Bit |
|-------|----------|-------|----------|-------|----------|-------|----------|
| 4000  | S_ISUID  | 0400  | S_IRUSR  | 0040  | S_IRGRP  | 0004  | S_IROTH  |
| 2000  | S_ISGID  | 0200  | S_IWUSR  | 0020  | S_IWGRP  | 0002  | S_IWOTH  |
| 1000  | S_ISVTX  | 0100  | S_IXUSR  | 0010  | S_IXGRP  | 0001  | S_IXOTH  |

When bits are set in the octal number other than those listed in the table above, the behavior is unspecified.



**Grammar for chmod**

The grammar and lexical conventions in this section describe the syntax for the *symbolic\_mode* operand. The general conventions for this style of grammar are described in [Section 1.3](#) (on page 2325). A valid *symbolic\_mode* can be represented as the non-terminal symbol *symbolic\_mode* in the grammar. This formal syntax shall take precedence over the preceding text syntax description.

The lexical processing is based entirely on single characters. Implementations need not allow <blank> characters within the single argument being processed.

```
%start      symbolic_mode
%%

symbolic_mode      : clause
                    | symbolic_mode ',' clause
                    ;

clause              : actionlist
                    | wholist actionlist
                    ;

wholist              : who
                    | wholist who
                    ;

who                  : 'u' | 'g' | 'o' | 'a'
                    ;

actionlist           : action
                    | actionlist action
                    ;

action               : op
                    | op permlist
                    | op permcopy
                    ;

permcopy             : 'u' | 'g' | 'o'
                    ;

op                   : '+' | '-' | '='
                    ;

permlist             : perm
                    | perm permlist
                    ;

XSI perm             : 'r' | 'w' | 'x' | 'X' | 's' | 't'
                    ;
```

**EXIT STATUS**

The following exit values shall be returned:

- 0 The utility executed successfully and all requested changes were made.
- >0 An error occurred.



## CONSEQUENCES OF ERRORS

Default.

## APPLICATION USAGE

Some implementations of the *chmod* utility change the mode of a directory before the files in the directory when performing a recursive (**-R** option) change; others change the directory mode after the files in the directory. If an application tries to remove read or search permission for a file hierarchy, the removal attempt fails if the directory is changed first; on the other hand, trying to re-enable permissions to a restricted hierarchy fails if directories are changed last. Users should not try to make a hierarchy inaccessible to themselves.

Some implementations of *chmod* never used the *umask* of the process when changing modes; systems conformant with this volume of POSIX.1-2008 do so when **who** is not specified. Note the difference between:

```
chmod a-w file
```

which removes all write permissions, and:

```
chmod -- -w file
```

which removes write permissions that would be allowed if **file** was created with the same *umask*.

Conforming applications should never assume that they know how the set-user-ID and set-group-ID bits on directories are interpreted.

## EXAMPLES

| Mode         | Results                                                                                            |
|--------------|----------------------------------------------------------------------------------------------------|
| <i>a+=</i>   | Equivalent to <i>a+,a=</i> ; clears all file mode bits.                                            |
| <i>go+-w</i> | Equivalent to <i>go+,go-w</i> ; clears group and other write bits.                                 |
| <i>g=o-w</i> | Equivalent to <i>g=o,g-w</i> ; sets group bit to match other bits and then clears group write bit. |
| <i>g-r+w</i> | Equivalent to <i>g-r,g+w</i> ; clears group read bit and sets group write bit.                     |
| <i>uo=g</i>  | Sets owner bits to match group bits and sets other bits to match group bits.                       |

## RATIONALE

The functionality of *chmod* is described substantially through references to concepts defined in the System Interfaces volume of POSIX.1-2008. In this way, there is less duplication of effort required for describing the interactions of permissions. However, the behavior of this utility is not described in terms of the *chmod()* function from the System Interfaces volume of POSIX.1-2008 because that specification requires certain side-effects upon alternate file access control mechanisms that might not be appropriate, depending on the implementation.

Implementations that support mandatory file and record locking as specified by the 1984 /usr/group standard historically used the combination of set-group-ID bit set and group execute bit clear to indicate mandatory locking. This condition is usually set or cleared with the symbolic mode **perm** symbol **l** instead of the **perm** symbols **s** and **x** so that the mandatory locking mode is not changed without explicit indication that that was what the user intended. Therefore, the details on how the implementation treats these conditions must be defined in the documentation. This volume of POSIX.1-2008 does not require mandatory locking (nor does the System Interfaces volume of POSIX.1-2008), but does allow it as an extension. However, this volume of POSIX.1-2008 does require that the *ls* and *chmod* utilities work consistently in this

area. If *ls -l file* indicates that the set-group-ID bit is set, *chmod g-s file* must clear it (assuming appropriate privileges exist to change modes).

The System V and BSD versions use different exit status codes. Some implementations used the exit status as a count of the number of errors that occurred; this practice is unworkable since it can overflow the range of valid exit status values. This problem is avoided here by specifying only 0 and >0 as exit values.

The System Interfaces volume of POSIX.1-2008 indicates that implementation-defined restrictions may cause the S\_ISUID and S\_ISGID bits to be ignored. This volume of POSIX.1-2008 allows the *chmod* utility to choose to modify these bits before calling *chmod()* (or some function providing equivalent capabilities) for non-regular files. Among other things, this allows implementations that use the set-user-ID and set-group-ID bits on directories to enable extended features to handle these extensions in an intelligent manner.

The **X perm** symbol was adopted from BSD-based systems because it provides commonly desired functionality when doing recursive (**-R** option) modifications. Similar functionality is not provided by the *find* utility. Historical BSD versions of *chmod*, however, only supported **X** with *op+*; it has been extended in this volume of POSIX.1-2008 because it is also useful with *op=*. (It has also been added for *op-* even though it duplicates **x**, in this case, because it is intuitive and easier to explain.)

The grammar was extended with the *permcopy* non-terminal to allow historical-practice forms of symbolic modes like **o=u -g** (that is, set the “other” permissions to the permissions of “owner” minus the permissions of “group”).

## FUTURE DIRECTIONS

None.

## SEE ALSO

*ls*, *umask*

XBD [Section 4.5](#) (on page 108), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

XSH *chmod()*

## CHANGE HISTORY

First released in Issue 2.

## Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- Octal modes have been kept and made mandatory despite being marked obsolescent in the ISO POSIX-2: 1993 standard.

IEEE PASC Interpretation 1003.2 #172 is applied, changing the CONSEQUENCES OF ERRORS section to “Default.”.

The Open Group Base Resolution bwg2001-010 is applied, adding the description of the S\_ISVTX bit and the **t perm** symbol as part of the XSI option.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/16 is applied, changing the XSI shaded text in the EXTENDED DESCRIPTION from:

“The **perm** symbol **t** shall specify the S\_ISVTX bit and shall apply to directories only. The effect when using it with any other file type is unspecified. It can be used with the **who** symbols **o**, **a**, or with no **who** symbol. It shall not be an error to specify a **who** symbol of **u** or **g** in conjunction with the **perm** symbol **t**; it shall be ignored for **u** and **g**.”

83240 to:

83241 “The **perm** symbol **t** shall specify the S\_ISVTX bit. When used with a file of type directory,  
83242 it can be used with the **who** symbol **a**, or with no **who** symbol. It shall not be an error to  
83243 specify a **who** symbol of **u**, **g**, or **o** in conjunction with the **perm** symbol **t**, but the meaning  
83244 of these combinations is unspecified. The effect when using the **perm** symbol **t** with any  
83245 file type other than directory is unspecified.”

83246 This change is to permit historical behavior.

83247 **Issue 7**

83248 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

83249 Austin Group Interpretation 1003.1-2001 #130 is applied, adding text to the DESCRIPTION  
83250 about about marking for update the last file status change timestamp of the file.

**NAME**

chown — change the file ownership

**SYNOPSIS**

chown [-h] owner[:group] file...

chown -R [-H|-L|-P] owner[:group] file...

**DESCRIPTION**

The *chown* utility shall set the user ID of the file named by each *file* operand to the user ID specified by the *owner* operand.

For each *file* operand, or, if the **-R** option is used, each file encountered while walking the directory trees specified by the *file* operands, the *chown* utility shall perform actions equivalent to the *chown()* function defined in the System Interfaces volume of POSIX.1-2008, called with the following arguments:

1. The *file* operand shall be used as the *path* argument.
2. The user ID indicated by the *owner* portion of the first operand shall be used as the *owner* argument.
3. If the *group* portion of the first operand is given, the group ID indicated by it shall be used as the *group* argument; otherwise, the group ownership shall not be changed.

Unless *chown* is invoked by a process with appropriate privileges, the set-user-ID and set-group-ID bits of a regular file shall be cleared upon successful completion; the set-user-ID and set-group-ID bits of other file types may be cleared.

**OPTIONS**

The *chown* utility shall conform to XBD [Section 12.2](#) (on page 216).

The following options shall be supported by the implementation:

- h** For each file operand that names a file of type symbolic link, *chown* shall attempt to set the user ID of the symbolic link. If a group ID was specified, for each file operand that names a file of type symbolic link, *chown* shall attempt to set the group ID of the symbolic link.
- H** If the **-R** option is specified and a symbolic link referencing a file of type directory is specified on the command line, *chown* shall change the user ID (and group ID, if specified) of the directory referenced by the symbolic link and all files in the file hierarchy below it.
- L** If the **-R** option is specified and a symbolic link referencing a file of type directory is specified on the command line or encountered during the traversal of a file hierarchy, *chown* shall change the user ID (and group ID, if specified) of the directory referenced by the symbolic link and all files in the file hierarchy below it.
- P** If the **-R** option is specified and a symbolic link is specified on the command line or encountered during the traversal of a file hierarchy, *chown* shall change the owner ID (and group ID, if specified) of the symbolic link. The *chown* utility shall not follow the symbolic link to any other part of the file hierarchy.
- R** Recursively change file user and group IDs. For each *file* operand that names a directory, *chown* shall change the user ID (and group ID, if specified) of the directory and all files in the file hierarchy below it. Unless a **-H**, **-L**, or **-P** option is specified, it is unspecified which of these options will be used as the default.

Specifying more than one of the mutually-exclusive options **-H**, **-L**, and **-P** shall not be

83295 considered an error. The last option specified shall determine the behavior of the utility.

#### 83296 OPERANDS

83297 The following operands shall be supported:

83298 *owner[:group]* A user ID and optional group ID to be assigned to *file*. The *owner* portion of this  
 83299 operand shall be a user name from the user database or a numeric user ID. Either  
 83300 specifies a user ID which shall be given to each file named by one of the *file*  
 83301 operands. If a numeric *owner* operand exists in the user database as a user name,  
 83302 the user ID number associated with that user name shall be used as the user ID.  
 83303 Similarly, if the *group* portion of this operand is present, it shall be a group name  
 83304 from the group database or a numeric group ID. Either specifies a group ID which  
 83305 shall be given to each file. If a numeric group operand exists in the group database  
 83306 as a group name, the group ID number associated with that group name shall be  
 83307 used as the group ID.

83308 *file* A pathname of a file whose user ID is to be modified.

#### 83309 STDIN

83310 Not used.

#### 83311 INPUT FILES

83312 None.

#### 83313 ENVIRONMENT VARIABLES

83314 The following environment variables shall affect the execution of *chown*:

83315 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 83316 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 83317 variables used to determine the values of locale categories.)

83318 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 83319 internationalization variables.

83320 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 83321 characters (for example, single-byte as opposed to multi-byte characters in  
 83322 arguments).

83323 *LC\_MESSAGES*

83324 Determine the locale that should be used to affect the format and contents of  
 83325 diagnostic messages written to standard error.

83326 *XSI* *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

#### 83327 ASYNCHRONOUS EVENTS

83328 Default.

#### 83329 STDOUT

83330 Not used.

#### 83331 STDERR

83332 The standard error shall be used only for diagnostic messages.

#### 83333 OUTPUT FILES

83334 None.

83335 **EXTENDED DESCRIPTION**

83336 None.

83337 **EXIT STATUS**

83338 The following exit values shall be returned:

83339 0 The utility executed successfully and all requested changes were made.

83340 &gt;0 An error occurred.

83341 **CONSEQUENCES OF ERRORS**

83342 Default.

83343 **APPLICATION USAGE**83344 Only the owner of a file or the user with appropriate privileges may change the owner or group  
83345 of a file.83346 Some implementations restrict the use of *chown* to a user with appropriate privileges.83347 **EXAMPLES**

83348 None.

83349 **RATIONALE**83350 The System V and BSD versions use different exit status codes. Some implementations used the  
83351 exit status as a count of the number of errors that occurred; this practice is unworkable since it  
83352 can overflow the range of valid exit status values. These are masked by specifying only 0 and >0  
83353 as exit values.83354 The functionality of *chown* is described substantially through references to functions in the  
83355 System Interfaces volume of POSIX.1-2008. In this way, there is no duplication of effort required  
83356 for describing the interactions of permissions, multiple groups, and so on.83357 The 4.3 BSD method of specifying both owner and group was included in this volume of  
83358 POSIX.1-2008 because:

- 83359 • There are cases where the desired end condition could not be achieved using the *chgrp* and  
83360 *chown* (that only changed the user ID) utilities. (If the current owner is not a member of the  
83361 desired group and the desired owner is not a member of the current group, the *chown()*  
83362 function could fail unless both owner and group are changed at the same time.)
- 83363 • Even if they could be changed independently, in cases where both are being changed, there  
83364 is a 100% performance penalty caused by being forced to invoke both utilities.

83365 The BSD syntax *user[.group]* was changed to *user[:group]* in this volume of POSIX.1-2008 because  
83366 the <period> is a valid character in login names (as specified by the Base Definitions volume of  
83367 POSIX.1-2008, login names consist of characters in the portable filename character set). The  
83368 <colon> character was chosen as the replacement for the <period> character because it would  
83369 never be allowed as a character in a user name or group name on historical implementations.83370 The **-R** option is considered by some observers as an undesirable departure from the historical  
83371 UNIX system tools approach; since a tool, *find*, already exists to recurse over directories, there  
83372 seemed to be no good reason to require other tools to have to duplicate that functionality.  
83373 However, the **-R** option was deemed an important user convenience, is far more efficient than  
83374 forking a separate process for each element of the directory hierarchy, and is in widespread  
83375 historical use.

83376 **FUTURE DIRECTIONS**

83377 None.

83378 **SEE ALSO**83379 *chgrp, chmod*83380 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)83381 XSH *chown()*83382 **CHANGE HISTORY**

83383 First released in Issue 2.

83384 **Issue 6**83385 New options **-h**, **-H**, **-L**, and **-P** are added to align with the IEEE P1003.2b draft standard. These  
83386 options affect the processing of symbolic links.

83387 The normative text is reworded to avoid use of the term “must” for application requirements.

83388 IEEE PASC Interpretation 1003.2 #172 is applied, changing the CONSEQUENCES OF ERRORS  
83389 section to “Default.”.83390 The “otherwise, ...” text in item 3. of the DESCRIPTION is changed to “otherwise, the group  
83391 ownership shall not be changed”.83392 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/17 is applied, changing the SYNOPSIS to  
83393 make it clear that **-h** and **-R** are optional.83394 **Issue 7**83395 SD5-XCU-ERN-9 is applied, removing the **-R** from the first line of the SYNOPSIS.

83396 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

83397 The description of the **-h** and **-P** options is revised.



83398 **NAME**

83399 cksum — write file checksums and sizes

83400 **SYNOPSIS**83401 cksum [*file*...]83402 **DESCRIPTION**

83403 The *cksum* utility shall calculate and write to standard output a cyclic redundancy check (CRC)  
 83404 for each input file, and also write to standard output the number of octets in each file. The CRC  
 83405 used is based on the polynomial used for CRC error checking in the ISO/IEC 8802-3:1996  
 83406 standard (Ethernet).

83407 The encoding for the CRC checksum is defined by the generating polynomial:

$$83408 G(x) = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$$

83409 Mathematically, the CRC value corresponding to a given file shall be defined by the following  
 83410 procedure:

- 83411 1. The *n* bits to be evaluated are considered to be the coefficients of a mod 2 polynomial  
 83412 *M*(*x*) of degree *n*−1. These *n* bits are the bits from the file, with the most significant bit  
 83413 being the most significant bit of the first octet of the file and the last bit being the least  
 83414 significant bit of the last octet, padded with zero bits (if necessary) to achieve an integral  
 83415 number of octets, followed by one or more octets representing the length of the file as a  
 83416 binary value, least significant octet first. The smallest number of octets capable of  
 83417 representing this integer shall be used.
- 83418 2. *M*(*x*) is multiplied by  $x^{32}$  (that is, shifted left 32 bits) and divided by *G*(*x*) using mod 2  
 83419 division, producing a remainder *R*(*x*) of degree ≤ 31.
- 83420 3. The coefficients of *R*(*x*) are considered to be a 32-bit sequence.
- 83421 4. The bit sequence is complemented and the result is the CRC.

83422 **OPTIONS**

83423 None.

83424 **OPERANDS**

83425 The following operand shall be supported:

83426 *file* A pathname of a file to be checked. If no *file* operands are specified, the standard  
 83427 input shall be used.

83428 **STDIN**

83429 The standard input shall be used if no *file* operands are specified, and shall be used if a *file*  
 83430 operand is '−' and the implementation treats the '−' as meaning standard input. Otherwise,  
 83431 the standard input shall not be used. See the INPUT FILES section.

83432 **INPUT FILES**

83433 The input files can be any file type.

83434 **ENVIRONMENT VARIABLES**83435 The following environment variables shall affect the execution of *cksum*:

- 83436 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 83437 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 83438 variables used to determine the values of locale categories.)
- 83439 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 83440 internationalization variables.



83441 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 83442 characters (for example, single-byte as opposed to multi-byte characters in  
 83443 arguments).

83444 **LC\_MESSAGES**  
 83445 Determine the locale that should be used to affect the format and contents of  
 83446 diagnostic messages written to standard error.

83447 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

## 83448 **ASYNCHRONOUS EVENTS**

83449 Default.

## 83450 **STDOUT**

83451 For each file processed successfully, the *cksum* utility shall write in the following format:

83452 "%u %d %s\n", <checksum>, <# of octets>, <pathname>

83453 If no *file* operand was specified, the pathname and its leading <space> shall be omitted.

## 83454 **STDERR**

83455 The standard error shall be used only for diagnostic messages.

## 83456 **OUTPUT FILES**

83457 None.

## 83458 **EXTENDED DESCRIPTION**

83459 None.

## 83460 **EXIT STATUS**

83461 The following exit values shall be returned:

83462 0 All files were processed successfully.

83463 >0 An error occurred.

## 83464 **CONSEQUENCES OF ERRORS**

83465 Default.

## 83466 **APPLICATION USAGE**

83467 The *cksum* utility is typically used to quickly compare a suspect file against a trusted version of  
 83468 the same, such as to ensure that files transmitted over noisy media arrive intact. However, this  
 83469 comparison cannot be considered cryptographically secure. The chances of a damaged file  
 83470 producing the same CRC as the original are small; deliberate deception is difficult, but probably  
 83471 not impossible.

83472 Although input files to *cksum* can be any type, the results need not be what would be expected  
 83473 on character special device files or on file types not described by the System Interfaces volume of  
 83474 POSIX.1-2008. Since this volume of POSIX.1-2008 does not specify the block size used when  
 83475 doing input, checksums of character special files need not process all of the data in those files.

83476 The algorithm is expressed in terms of a bitstream divided into octets. If a file is transmitted  
 83477 between two systems and undergoes any data transformation (such as changing little-endian  
 83478 byte ordering to big-endian), identical CRC values cannot be expected. Implementations  
 83479 performing such transformations may extend *cksum* to handle such situations.

## 83480 **EXAMPLES**

83481 None.

## RATIONALE

The following C-language program can be used as a model to describe the algorithm. It assumes that a **char** is one octet. It also assumes that the entire file is available for one pass through the function. This was done for simplicity in demonstrating the algorithm, rather than as an implementation model.

```
static unsigned long crctab[] = {
0x00000000,
0x04c11db7, 0x09823b6e, 0x0d4326d9, 0x130476dc, 0x17c56b6b,
0x1a864db2, 0x1e475005, 0x2608edb8, 0x22c9f00f, 0x2f8ad6d6,
0x2b4bcb61, 0x350c9b64, 0x31cd86d3, 0x3c8ea00a, 0x384fbd8d,
0x4c11db70, 0x48d0c6c7, 0x4593e01e, 0x4152fda9, 0x5f15adac,
0x5bd4b01b, 0x569796c2, 0x52568b75, 0x6a1936c8, 0x6ed82b7f,
0x639b0da6, 0x675a1011, 0x791d4014, 0x7ddc5da3, 0x709f7b7a,
0x745e66cd, 0x9823b6e0, 0x9ce2ab57, 0x91a18d8e, 0x95609039,
0x8b27c03c, 0x8fe6dd8b, 0x82a5fb52, 0x8664e6e5, 0xbe2b5b58,
0xbaea46ef, 0xb7a96036, 0xb3687d81, 0xad2f2d84, 0xa9ee3033,
0xa4ad16ea, 0xa06c0b5d, 0xd4326d90, 0xd0f37027, 0xddb056fe,
0xd9714b49, 0xc7361b4c, 0xc3f706fb, 0xceb42022, 0xca753d95,
0xf23a8028, 0xf6fb9d9f, 0xfbb8bb46, 0xff79a6f1, 0xe13ef6f4,
0xe5ffeb43, 0xe8bccd9a, 0xec7dd02d, 0x34867077, 0x30476dc0,
0x3d044b19, 0x39c556ae, 0x278206ab, 0x23431b1c, 0x2e003dc5,
0x2ac12072, 0x128e9dcf, 0x164f8078, 0x1b0ca6a1, 0x1fcd8bb16,
0x018aeb13, 0x054bf6a4, 0x0808d07d, 0x0cc9cdca, 0x7897ab07,
0x7c56b6b0, 0x71159069, 0x75d48dde, 0x6b93dddb, 0x6f52c06c,
0x6211e6b5, 0x66d0fb02, 0x5e9f46bf, 0x5a5e5b08, 0x571d7ddl,
0x53dc6066, 0x4d9b3063, 0x495a2dd4, 0x44190b0d, 0x40d816ba,
0xaca5c697, 0xa864db20, 0xa527fdf9, 0xale6e04e, 0xbfa1b04b,
0xbbb60adfc, 0xb6238b25, 0xb2e29692, 0x8aad2b2f, 0x8e6c3698,
0x832f1041, 0x87ee0df6, 0x99a95df3, 0x9d684044, 0x902b669d,
0x94ea7b2a, 0xe0b41de7, 0xe4750050, 0xe9362689, 0xedf73b3e,
0xf3b06b3b, 0xf771768c, 0xfa325055, 0xfef34de2, 0xc6bcf05f,
0xc27dede8, 0xcf3ecb31, 0xcbbffd686, 0xd5b88683, 0xd1799b34,
0xdc3abded, 0xd8fba05a, 0x690ce0ee, 0x6dcd5f59, 0x608edb80,
0x644fc637, 0x7a089632, 0x7ec98b85, 0x738aad5c, 0x774bb0eb,
0x4f040d56, 0x4bc510e1, 0x46863638, 0x42472b8f, 0x5c007b8a,
0x58c1663d, 0x558240e4, 0x51435d53, 0x251d3b9e, 0x21dc2629,
0x2c9f00f0, 0x285e1d47, 0x36194d42, 0x32d850f5, 0x3f9b762c,
0x3b5a6b9b, 0x0315d626, 0x07d4cb91, 0x0a97ed48, 0x0e56f0ff,
0x1011a0fa, 0x14d0bd4d, 0x19939b94, 0x1d528623, 0xf12f560e,
0xf5ee4bb9, 0xf8ad6d60, 0xfc6c70d7, 0xe22b20d2, 0xe6ea3d65,
0xeba91bbc, 0xef68060b, 0xd727bbb6, 0xd3e6a601, 0xdea580d8,
0xda649d6f, 0xc423cd6a, 0xc0e2d0dd, 0xcda1f604, 0xc960ebb3,
0xbd3e8d7e, 0xb9ff90c9, 0xb4bcb610, 0xb07daba7, 0xae3afba2,
0xaafbe615, 0xa7b8c0cc, 0xa379dd7b, 0x9b3660c6, 0x9ff77d71,
0x92b45ba8, 0x9675461f, 0x8832161a, 0x8cf30bad, 0x81b02d74,
0x857130c3, 0x5d8a9099, 0x594b8d2e, 0x5408abf7, 0x50c9b640,
0x4e8ee645, 0x4a4ffb2, 0x470cdd2b, 0x43cdc09c, 0x7b827d21,
0x7f436096, 0x7200464f, 0x76c15bf8, 0x68860bfd, 0x6c47164a,
0x61043093, 0x65c52d24, 0x119b4be9, 0x155a565e, 0x18197087,
0x1cd86d30, 0x029f3d35, 0x065e2082, 0x0b1d065b, 0x0fdc1bec,
0x3793a651, 0x3352bbe6, 0x3e119d3f, 0x3ad08088, 0x2497d08d,
```

```

83533     0x2056cd3a, 0x2d15ebe3, 0x29d4f654, 0xc5a92679, 0xc1683bce,
83534     0xcc2b1d17, 0xc8ea00a0, 0xd6ad50a5, 0xd26c4d12, 0xdf2f6bcb,
83535     0xdbee767c, 0xe3alcba1, 0xe760d676, 0xea23f0af, 0xee2ed18,
83536     0xf0a5bd1d, 0xf464a0aa, 0xf9278673, 0xfde69bc4, 0x89b8fd09,
83537     0x8d79e0be, 0x803ac667, 0x84fbd0, 0x9abc8bd5, 0x9e7d9662,
83538     0x933eb0bb, 0x97ffad0c, 0xafb010b1, 0xab710d06, 0xa6322bdf,
83539     0xa2f33668, 0xbcb4666d, 0xb8757bda, 0xb5365d03, 0xb1f740b4
83540     };

83541     unsigned long memcrc(const unsigned char *b, size_t n)
83542     {
83543     /*   Input arguments:
83544        *   const unsigned char*   b == byte sequence to checksum
83545        *   size_t                 n == length of sequence
83546        */

83547         register size_t i;
83548         register unsigned c, s = 0;

83549         for (i = n; i > 0; --i) {
83550             c = *b++;
83551             s = (s << 8) ^ crctab[(s >> 24) ^ c];
83552         }

83553         /* Extend with the length of the string. */
83554         while (n != 0) {
83555             c = n & 0377;
83556             n >>= 8;
83557             s = (s << 8) ^ crctab[(s >> 24) ^ c];
83558         }

83559         return ~s;
83560     }

```

The historical practice of writing the number of “blocks” has been changed to writing the number of octets, since the latter is not only more useful, but also since historical implementations have not been consistent in defining what a “block” meant.

The algorithm used was selected to increase the operational robustness of *cksum*. Neither the System V nor BSD *sum* algorithm was selected. Since each of these was different and each was the default behavior on those systems, no realistic compromise was available if either were selected—some set of historical applications would break. Therefore, the name was changed to *cksum*. Although the historical *sum* commands will probably continue to be provided for many years, programs designed for portability across systems should use the new name.

The algorithm selected is based on that used by the ISO/IEC 8802-3:1996 standard (Ethernet) for the frame check sequence field. The algorithm used does not match the technical definition of a *checksum*; the term is used for historical reasons. The length of the file is included in the CRC calculation because this parallels inclusion of a length field by Ethernet in its CRC, but also because it guards against inadvertent collisions between files that begin with different series of zero octets. The chance that two different files produce identical CRCs is much greater when their lengths are not considered. Keeping the length and the checksum of the file itself separate would yield a slightly more robust algorithm, but historical usage has always been that a single number (the checksum as printed) represents the signature of the file. It was decided that historical usage was the more important consideration.

Early proposals contained modifications to the Ethernet algorithm that involved extracting table values whenever an intermediate result became zero. This was demonstrated to be less robust than the current method and mathematically difficult to describe or justify.

The calculation used is identical to that given in pseudo-code in the referenced Sarwate article. The pseudo-code rendition is:

```
X <- 0; Y <- 0;
for i <- m -1 step -1 until 0 do
  begin
    T <- X(1) ^ A[i];
    X(1) <- X(0); X(0) <- Y(1); Y(1) <- Y(0); Y(0) <- 0;
    comment: f[T] and f'[T] denote the T-th words in the
             table f and f' ;
    X <- X ^ f[T]; Y <- Y ^ f'[T];
  end
```

The pseudo-code is reproduced exactly as given; however, note that in the case of *cksum*, **A[i]** represents a byte of the file, the words **X** and **Y** are treated as a single 32-bit value, and the tables **f** and **f'** are a single table containing 32-bit values.

The referenced Sarwate article also discusses generating the table.

#### **FUTURE DIRECTIONS**

None.

#### **SEE ALSO**

XBD [Chapter 8](#) (on page 173)

#### **CHANGE HISTORY**

First released in Issue 4.

#### **Issue 7**

Austin Group Interpretation 1003.1-2001 #092 is applied.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0081 [446] is applied.

83608 **NAME**83609 `cmp` — compare two files83610 **SYNOPSIS**83611 `cmp [-l|-s] file1 file2`83612 **DESCRIPTION**

83613 The *cmp* utility shall compare two files. The *cmp* utility shall write no output if the files are the  
 83614 same. Under default options, if they differ, it shall write to standard output the byte and line  
 83615 number at which the first difference occurred. Bytes and lines shall be numbered beginning with  
 83616 1.

83617 **OPTIONS**83618 The *cmp* utility shall conform to XBD [Section 12.2](#) (on page 216).

83619 The following options shall be supported:

83620 **-l** (Lowercase ell.) Write the byte number (decimal) and the differing bytes (octal) for  
 83621 each difference.

83622 **-s** Write nothing to standard output or standard error when files differ; indicate |  
 83623 differing files through exit status only. It is unspecified whether a diagnostic |  
 83624 message is written to standard error when an error is encountered; if a message is |  
 83625 not written, the error is indicated through exit status only.

83626 **OPERANDS**

83627 The following operands shall be supported:

83628 *file1* A pathname of the first file to be compared. If *file1* is '-', the standard input shall  
 83629 be used.

83630 *file2* A pathname of the second file to be compared. If *file2* is '-', the standard input  
 83631 shall be used.

83632 If both *file1* and *file2* refer to standard input or refer to the same FIFO special, block special, or  
 83633 character special file, the results are undefined.

83634 **STDIN**

83635 The standard input shall be used only if the *file1* or *file2* operand refers to standard input. See the  
 83636 INPUT FILES section.

83637 **INPUT FILES**

83638 The input files can be any file type.

83639 **ENVIRONMENT VARIABLES**83640 The following environment variables shall affect the execution of *cmp*:

83641 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 83642 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 83643 variables used to determine the values of locale categories.)

83644 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 83645 internationalization variables.

83646 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 83647 characters (for example, single-byte as opposed to multi-byte characters in  
 83648 arguments).

83649 **LC\_MESSAGES**

83650 Determine the locale that should be used to affect the format and contents of  
 83651 diagnostic messages written to standard error and informative messages written to

83652 standard output.

83653 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 83654 ASYNCHRONOUS EVENTS

83655 Default.

## 83656 STDOUT

83657 In the POSIX locale, results of the comparison shall be written to standard output. When no  
83658 options are used, the format shall be:

83659 "%s %s differ: char %d, line %d\n", *file1*, *file2*,  
83660 <byte number>, <line number>

83661 When the *-l* option is used, the format shall be:

83662 "%d %o %o\n", <byte number>, <differing byte>,  
83663 <differing byte>

83664 for each byte that differs. The first <differing byte> number is from *file1* while the second is from  
83665 *file2*. In both cases, <byte number> shall be relative to the beginning of the file, beginning with 1.

83666 No output shall be written to standard output when the *-s* option is used.

## 83667 STDERR

83668 The standard error shall be used only for diagnostic messages. If the *-l* option is used and *file1*  
83669 and *file2* differ in length, or if the *-s* option is not used and *file1* and *file2* are identical for the  
83670 entire length of the shorter file, in the POSIX locale the following diagnostic message shall be  
83671 written:

83672 "cmp: EOF on %s%s\n", <name of shorter file>, <additional info>

83673 The <additional info> field shall either be null or a string that starts with a <blank> and contains  
83674 no <newline> characters. Some implementations report on the number of lines in this case.

83675 If the *-s* option is used and an error occurs, it is unspecified whether a diagnostic message is +  
83676 written to standard error.

## 83677 OUTPUT FILES

83678 None.

## 83679 EXTENDED DESCRIPTION

83680 None.

## 83681 EXIT STATUS

83682 The following exit values shall be returned:

83683 0 The files are identical.

83684 1 The files are different; this includes the case where one file is identical to the first part of the  
83685 other.

83686 >1 An error occurred.

## 83687 CONSEQUENCES OF ERRORS

83688 Default.

## APPLICATION USAGE

Although input files to *cmp* can be any type, the results might not be what would be expected on character special device files or on file types not described by the System Interfaces volume of POSIX.1-2008. Since this volume of POSIX.1-2008 does not specify the block size used when doing input, comparisons of character special files need not compare all of the data in those files.

For files which are not text files, line numbers simply reflect the presence of a <newline>, without any implication that the file is organized into lines.

Since the behavior of *-s* differs between implementations as to whether error messages are written, the only way to ensure consistent behavior of *cmp* when *-s* is used is to redirect standard error to */dev/null*.

If error messages are wanted, instead of using *-s* standard output should be redirected to */dev/null*, and anything written to standard error should be discarded if the exit status is 1. For example:

```
silent_cmp() {
    # compare files with no output except error messages
    message=$(cmp "$@" 2>&1 >/dev/null)
    status=$?
    case $status in
        0|1) ;;
        *) printf '%s\n' "$message" ;;
    esac
    return $status
}
```

## EXAMPLES

None.

## RATIONALE

The global language in [Section 1.4](#) (on page 2326) indicates that using two mutually-exclusive options together produces unspecified results. Some System V implementations consider the option usage:

```
cmp -l -s ...
```

to be an error. They also treat:

```
cmp -s -l ...
```

as if no options were specified. Both of these behaviors are considered bugs, but are allowed.

The word **char** in the standard output format comes from historical usage, even though it is actually a byte number. When *cmp* is supported in other locales, implementations are encouraged to use the word *byte* or its equivalent in another language. Users should not interpret this difference to indicate that the functionality of the utility changed between locales.

Some implementations report on the number of lines in the identical-but-shorter file case. This is allowed by the inclusion of the <additional info> fields in the output format. The restriction on having a leading <blank> and no <newline> characters is to make parsing for the filename easier. It is recognized that some filenames containing white-space characters make parsing difficult anyway, but the restriction does aid programs used on systems where the names are predominantly well behaved.

83732 **FUTURE DIRECTIONS**

83733 Future versions of this standard may require that diagnostic messages are written to standard |  
83734 error when the `-s` option is specified.

83735 **SEE ALSO**

83736 *comm*, *diff*

83737 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

83738 **CHANGE HISTORY**

83739 First released in Issue 2.

83740 **Issue 7**

83741 SD5-XCU-ERN-96 is applied, updating the STDERR section.

83742 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



83743 **NAME**83744 `comm` — select or reject lines common to two files83745 **SYNOPSIS**83746 `comm [-123] file1 file2`83747 **DESCRIPTION**

83748 The *comm* utility shall read *file1* and *file2*, which should be ordered in the current collating  
 83749 sequence, and produce three text columns as output: lines only in *file1*, lines only in *file2*, and  
 83750 lines in both files.

83751 If the lines in both files are not ordered according to the collating sequence of the current locale,  
 83752 the results are unspecified.

83753 If the collating sequence of the current locale does not have a total ordering of all characters (see +  
 83754 XBD Section 7.3.2, on page 147) and any lines from the input files collate equally but are not +  
 83755 identical, *comm* should treat them as different lines but may treat them as being the same. If it +  
 83756 treats them as different, *comm* should expect them to be ordered according to a further byte-by- +  
 83757 byte comparison using the collating sequence for the POSIX locale and if they are not ordered in +  
 83758 this way, the output of *comm* can identify such lines as being both unique to *file1* and unique to +  
 83759 *file2* instead of being in both files.

83760 **OPTIONS**83761 The *comm* utility shall conform to XBD Section 12.2 (on page 216).

83762 The following options shall be supported:

- 83763 **-1** Suppress the output column of lines unique to *file1*.
- 83764 **-2** Suppress the output column of lines unique to *file2*.
- 83765 **-3** Suppress the output column of lines duplicated in *file1* and *file2*.

83766 **OPERANDS**

83767 The following operands shall be supported:

- 83768 *file1* A pathname of the first file to be compared. If *file1* is '-', the standard input shall  
 83769 be used.
- 83770 *file2* A pathname of the second file to be compared. If *file2* is '-', the standard input  
 83771 shall be used.

83772 If both *file1* and *file2* refer to standard input or to the same FIFO special, block special, or  
 83773 character special file, the results are undefined.

83774 **STDIN**

83775 The standard input shall be used only if one of the *file1* or *file2* operands refers to standard input.  
 83776 See the INPUT FILES section.

83777 **INPUT FILES**

83778 The input files shall be text files.

83779 **ENVIRONMENT VARIABLES**83780 The following environment variables shall affect the execution of *comm*:

- 83781 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 83782 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 83783 variables used to determine the values of locale categories.)

83784 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
83785 internationalization variables.

83786 *LC\_COLLATE*  
83787 Determine the locale for the collating sequence *comm* expects to have been used  
83788 when the input files were sorted.

83789 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
83790 characters (for example, single-byte as opposed to multi-byte characters in  
83791 arguments and input files).

83792 *LC\_MESSAGES*  
83793 Determine the locale that should be used to affect the format and contents of  
83794 diagnostic messages written to standard error.

83795 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 83796 ASYNCHRONOUS EVENTS

83797 Default.

## 83798 STDOUT

83799 The *comm* utility shall produce output depending on the options selected. If the *-1*, *-2*, and *-3*  
83800 options are all selected, *comm* shall write nothing to standard output.

83801 If the *-1* option is not selected, lines contained only in *file1* shall be written using the format:

83802 "%s\n", <line in file1>

83803 If the *-2* option is not selected, lines contained only in *file2* are written using the format:

83804 "%s%s\n", <lead>, <line in file2>

83805 where the string <lead> is as follows:

83806 <tab> The *-1* option is not selected.

83807 null string The *-1* option is selected.

83808 If the *-3* option is not selected, lines contained in both files shall be written using the format:

83809 "%s%s\n", <lead>, <line in both>

83810 where the string <lead> is as follows:

83811 <tab><tab> Neither the *-1* nor the *-2* option is selected.

83812 <tab> Exactly one of the *-1* and *-2* options is selected.

83813 null string Both the *-1* and *-2* options are selected.

83814 If the input files were ordered according to the collating sequence of the current locale, the lines  
83815 written shall be in the collating sequence of the current locale. If the input files contained any  
83816 lines that collated equally but were not identical and within each file those lines were ordered  
83817 according to a further byte-by-byte comparison using the collating sequence for the POSIX  
83818 locale, and *comm* treated them as different lines, then lines written that collate equally but are not  
83819 identical should be ordered according to a further byte-by-byte comparison using the collating  
83820 sequence for the POSIX locale.

## 83821 STDERR

83822 The standard error shall be used only for diagnostic messages.

83823 **OUTPUT FILES**

83824 None.

83825 **EXTENDED DESCRIPTION**

83826 None.

83827 **EXIT STATUS**

83828 The following exit values shall be returned:

83829 0 All input files were successfully output as specified.

83830 &gt;0 An error occurred.

83831 **CONSEQUENCES OF ERRORS**

83832 Default.

83833 **APPLICATION USAGE**83834 If the input files are not properly presorted, the output of *comm* might not be useful.

83835 When using *comm* to process pathnames, it is recommended that LC\_ALL, or at least LC\_CTYPE +  
 83836 and LC\_COLLATE, are set to POSIX or C in the environment, since pathnames can contain byte +  
 83837 sequences that do not form valid characters in some locales, in which case the utility's behavior +  
 83838 would be undefined. In the POSIX locale each byte is a valid single-byte character, and therefore +  
 83839 this problem is avoided. +

83840 If the collating sequence of the current locale does not have a total ordering of all characters, this +  
 83841 can affect the behavior of *comm* in the following ways: +

- 83842 • If *comm* treats lines as being the same only if they are identical, some lines can be +  
 83843 misleadingly identified as being both unique to *file1* and unique to *file2*. +
- 83844 • If *comm* treats lines as being the same if they collate equally and a line from *file1* collates +  
 83845 equally with a line from *file2* but is not identical to it, one of the lines is misleadingly +  
 83846 identified as being in both files and the other is not written to the output at all. +

83847 Such problems can be avoided by forcing the use of the POSIX locale; for example, the following +  
 83848 identifies lines in both *file1* and *file2*: +

```
83849 LC_ALL=POSIX sort file1 > file1.posix +
83850 LC_ALL=POSIX sort file2 > file2.posix +
83851 LC_ALL=POSIX comm -12 file1.posix file2.posix | sort +
```

83852 The final *sort* re-sorts the output of *comm* according to the collating sequence of the original +  
 83853 locale. Doing this might be difficult if more than one column is output and leading <blank>s +  
 83854 cannot be ignored. +

83855 **EXAMPLES**

83856 If a file named **xcu** contains a sorted list of the utilities in this volume of POSIX.1-2008, a file  
 83857 named **xpg3** contains a sorted list of the utilities specified in the X/Open Portability Guide, Issue  
 83858 3, and a file named **svid89** contains a sorted list of the utilities in the System V Interface  
 83859 Definition Third Edition:

```
83860 comm -23 xcu xpg3 | comm -23 - svid89
```

83861 would print a list of utilities in this volume of POSIX.1-2008 not specified by either of the other  
 83862 documents:

```
83863 comm -12 xcu xpg3 | comm -12 - svid89
```

83864 would print a list of utilities specified by all three documents, and:

83865           comm -l2 xpg3 svid89 | comm -23 - xcu

83866           would print a list of utilities specified by both XPG3 and the SVID, but not specified in this  
83867           volume of POSIX.1-2008.

83868 **RATIONALE**

83869           None.

83870 **FUTURE DIRECTIONS**

83871           A future version of this standard may require that if any lines from the input files collate equally |  
83872           but are not identical, then *comm* treats them as different lines and expects them to be ordered |  
83873           according to a further byte-by-byte comparison using the collating sequence for the POSIX |  
83874           locale. |

83875           A future version of this standard may require that if the input files contained any lines that |  
83876           collated equally but were not identical and within each file those lines were ordered according to |  
83877           a further byte-by-byte comparison using the collating sequence for the POSIX locale, then lines |  
83878           written that collate equally but are not identical are ordered according to a further byte-by-byte |  
83879           comparison using the collating sequence for the POSIX locale.

83880 **SEE ALSO**

83881           *cmp, diff, sort, uniq*

83882           XBD [Section 7.3.2](#) (on page 147), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

83883 **CHANGE HISTORY**

83884           First released in Issue 2.

83885 **Issue 6**

83886           The normative text is reworded to avoid use of the term “must” for application requirements.

83887 **NAME**83888        `command` — execute a simple command83889 **SYNOPSIS**83890        `command` [-p] *command\_name* [*argument...*]83891        `command` [-p][-v|-V] *command\_name*83892 **DESCRIPTION**83893        The *command* utility shall cause the shell to treat the arguments as a simple command,  
83894        suppressing the shell function lookup that is described in [Section 2.9.1.1](#) (on page 2357), item 1b.83895        If the *command\_name* is the same as the name of one of the special built-in utilities, the special  
83896        properties in the enumerated list at the beginning of [Section 2.14](#) (on page 2374) shall not occur.  
83897        In every other respect, if *command\_name* is not the name of a function, the effect of *command*  
83898        (with no options) shall be the same as omitting *command*.83899        When the `-v` or `-V` option is used, the *command* utility shall provide information concerning  
83900        how a command name is interpreted by the shell.83901 **OPTIONS**83902        The *command* utility shall conform to XBD [Section 12.2](#) (on page 216).

83903        The following options shall be supported:

83904        **-p**        Perform the command search using a default value for *PATH* that is guaranteed to  
83905        find all of the standard utilities.83906        **-v**        Write a string to standard output that indicates the pathname or command that  
83907        will be used by the shell, in the current shell execution environment (see [Section](#)  
83908        [2.12](#), on page 2371), to invoke *command\_name*, but do not invoke *command\_name*.83909                • Utilities, regular built-in utilities, *command\_names* including a `<slash>`  
83910                character, and any implementation-defined functions that are found using  
83911                the *PATH* variable (as described in [Section 2.9.1.1](#), on page 2357), shall be  
83912                written as absolute pathnames.83913                • Shell functions, special built-in utilities, regular built-in utilities not  
83914                associated with a *PATH* search, and shell reserved words shall be written as  
83915                just their names.83916                • An alias shall be written as a command line that represents its alias  
83917                definition.83918                • Otherwise, no output shall be written and the exit status shall reflect that the  
83919                name was not found.83920        **-V**        Write a string to standard output that indicates how the name given in the  
83921        *command\_name* operand will be interpreted by the shell, in the current shell  
83922        execution environment (see [Section 2.12](#), on page 2371), but do not invoke  
83923        *command\_name*. Although the format of this string is unspecified, it shall indicate  
83924        in which of the following categories *command\_name* falls and shall include the  
83925        information stated:83926                • Utilities, regular built-in utilities, and any implementation-defined functions  
83927                that are found using the *PATH* variable (as described in [Section 2.9.1.1](#), on  
83928                page 2357), shall be identified as such and include the absolute pathname in  
83929                the string.

- 83930 • Other shell functions shall be identified as functions.
- 83931 • Aliases shall be identified as aliases and their definitions included in the
- 83932 string.
- 83933 • Special built-in utilities shall be identified as special built-in utilities.
- 83934 • Regular built-in utilities not associated with a *PATH* search shall be identified
- 83935 as regular built-in utilities. (The term “regular” need not be used.)
- 83936 • Shell reserved words shall be identified as reserved words.

#### 83937 OPERANDS

83938 The following operands shall be supported:

83939 *argument* One of the strings treated as an argument to *command\_name*.

83940 *command\_name*

83941 The name of a utility or a special built-in utility.

#### 83942 STDIN

83943 Not used.

#### 83944 INPUT FILES

83945 None.

#### 83946 ENVIRONMENT VARIABLES

83947 The following environment variables shall affect the execution of *command*:

83948 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 83949 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 83950 variables used to determine the values of locale categories.)

83951 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 83952 internationalization variables.

83953 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 83954 characters (for example, single-byte as opposed to multi-byte characters in  
 83955 arguments).

83956 *LC\_MESSAGES*

83957 Determine the locale that should be used to affect the format and contents of  
 83958 diagnostic messages written to standard error and informative messages written to  
 83959 standard output.

83960 *XS* *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

83961 *PATH* Determine the search path used during the command search described in [Section](#)  
 83962 [2.9.1.1](#) (on page 2357), except as described under the *-p* option.

#### 83963 ASYNCHRONOUS EVENTS

83964 Default.

#### 83965 STDOUT

83966 When the *-v* option is specified, standard output shall be formatted as:

83967 "%s\n", <pathname or command>

83968 When the *-V* option is specified, standard output shall be formatted as:

83969 "%s\n", <unspecified>

**83970 STDERR**

83971 The standard error shall be used only for diagnostic messages.

**83972 OUTPUT FILES**

83973 None.

**83974 EXTENDED DESCRIPTION**

83975 None.

**83976 EXIT STATUS**

83977 When the `-v` or `-V` options are specified, the following exit values shall be returned:

83978 0 Successful completion.

83979 >0 The *command\_name* could not be found or an error occurred.

83980 Otherwise, the following exit values shall be returned:

83981 126 The utility specified by *command\_name* was found but could not be invoked.

83982 127 An error occurred in the *command* utility or the utility specified by *command\_name* could not  
83983 be found.

83984 Otherwise, the exit status of *command* shall be that of the simple command specified by the  
83985 arguments to *command*.

**83986 CONSEQUENCES OF ERRORS**

83987 Default.

**83988 APPLICATION USAGE**

83989 The order for command search allows functions to override regular built-ins and path searches.  
83990 This utility is necessary to allow functions that have the same name as a utility to call the utility  
83991 (instead of a recursive call to the function).

83992 The system default path is available using *getconf*; however, since *getconf* may need to have the  
83993 *PATH* set up before it can be called itself, the following can be used:

83994 `command -p getconf PATH`

83995 There are some advantages to suppressing the special characteristics of special built-ins on  
83996 occasion. For example:

83997 `command exec > unwritable-file`

83998 does not cause a non-interactive script to abort, so that the output status can be checked by the  
83999 script.

84000 The *command*, *env*, *nohup*, *time*, and *xargs* utilities have been specified to use exit code 127 if an  
84001 error occurs so that applications can distinguish “failure to find a utility” from “invoked utility  
84002 exited with an error indication”. The value 127 was chosen because it is not commonly used for  
84003 other meanings; most utilities use small values for “normal error conditions” and the values  
84004 above 128 can be confused with termination due to receipt of a signal. The value 126 was chosen  
84005 in a similar manner to indicate that the utility could be found, but not invoked. Some scripts  
84006 produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
84007 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
84008 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
84009 any other reason.

84010 Since the `-v` and `-V` options of *command* produce output in relation to the current shell execution  
84011 environment, *command* is generally provided as a shell regular built-in. If it is called in a subshell  
84012 or separate utility execution environment, such as one of the following:



```
84013 (PATH=foo command -v)
84014 nohup command -v
```

84015 it does not necessarily produce correct results. For example, when called with *nohup* or an *exec*  
 84016 function, in a separate utility execution environment, most implementations are not able to  
 84017 identify aliases, functions, or special built-ins.

84018 Two types of regular built-ins could be encountered on a system and these are described  
 84019 separately by *command*. The description of command search in [Section 2.9.1.1](#) (on page 2357)  
 84020 allows for a standard utility to be implemented as a regular built-in as long as it is found in the  
 84021 appropriate place in a *PATH* search. So, for example, *command -v true* might yield */bin/true* or  
 84022 some similar pathname. Other implementation-defined utilities that are not defined by this  
 84023 volume of POSIX.1-2008 might exist only as built-ins and have no pathname associated with  
 84024 them. These produce output identified as (regular) built-ins. Applications encountering these are  
 84025 not able to count on *execing* them, using them with *nohup*, overriding them with a different  
 84026 *PATH*, and so on.

## 84027 EXAMPLES

- 84028 1. Make a version of *cd* that always prints out the new working directory exactly once:

```
84029 cd() {
84030     command cd "$@" >/dev/null
84031     pwd
84032 }
```

- 84033 2. Start off a “secure shell script” in which the script avoids being spoofed by its parent:

```
84034 IFS='
84035 '
84036 # The preceding value should be <space><tab><newline>.
84037 # Set IFS to its default value.
84038 \unalias -a
84039 # Unset all possible aliases.
84040 # Note that unalias is escaped to prevent an alias
84041 # being used for unalias.
84042 unset -f command
84043 # Ensure command is not a user function.
84044 PATH="$(command -p getconf PATH):$PATH"
84045 # Put on a reliable PATH prefix.
84046 # ...
```

84047 At this point, given correct permissions on the directories called by *PATH*, the script has  
 84048 the ability to ensure that any utility it calls is the intended one. It is being very cautious  
 84049 because it assumes that implementation extensions may be present that would allow user  
 84050 functions to exist when it is invoked; this capability is not specified by this volume of  
 84051 POSIX.1-2008, but it is not prohibited as an extension. For example, the *ENV* variable  
 84052 precedes the invocation of the script with a user start-up script. Such a script could define  
 84053 functions to spoof the application.

## 84054 RATIONALE

84055 Since *command* is a regular built-in utility it is always found prior to the *PATH* search.

84056 There is nothing in the description of *command* that implies the command line is parsed any  
 84057 differently from that of any other simple command. For example:



command a | b ; c

is not parsed in any special way that causes ' | ' or ' ; ' to be treated other than a pipe operator or <semicolon> or that prevents function lookup on **b** or **c**.

The *command* utility is somewhat similar to the Eighth Edition shell *builtin* command, but since *command* also goes to the file system to search for utilities, the name *builtin* would not be intuitive.

The *command* utility is most likely to be provided as a regular built-in. It is not listed as a special built-in for the following reasons:

- The removal of exportable functions made the special precedence of a special built-in unnecessary.
- A special built-in has special properties (see [Section 2.14](#), on page 2374) that were inappropriate for invoking other utilities. For example, two commands such as:

```
date > unwritable-file
```

```
command date > unwritable-file
```

would have entirely different results; in a non-interactive script, the former would continue to execute the next command, the latter would abort. Introducing this semantic difference along with suppressing functions was seen to be non-intuitive.

The **-p** option is present because it is useful to be able to ensure a safe path search that finds all the standard utilities. This search might not be identical to the one that occurs through one of the *exec* functions (as defined in the System Interfaces volume of POSIX.1-2008) when *PATH* is unset. At the very least, this feature is required to allow the script to access the correct version of *getconf* so that the value of the default path can be accurately retrieved.

The *command* **-v** and **-V** options were added to satisfy requirements from users that are currently accomplished by three different historical utilities: *type* in the System V shell, *whence* in the KornShell, and *which* in the C shell. Since there is no historical agreement on how and what to accomplish here, the POSIX *command* utility was enhanced and the historical utilities were left unmodified. The C shell *which* merely conducts a path search. The KornShell *whence* is more elaborate—in addition to the categories required by POSIX, it also reports on tracked aliases, exported aliases, and undefined functions.

The output format of **-V** was left mostly unspecified because human users are its only audience. Applications should not be written to care about this information; they can use the output of **-v** to differentiate between various types of commands, but the additional information that may be emitted by the more verbose **-V** is not needed and should not be arbitrarily constrained in its verbosity or localization for application parsing reasons.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[Section 2.9.1.1](#) (on page 2357), [Section 2.12](#) (on page 2371), [Section 2.14](#) (on page 2374), *sh*, *type*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

XSH *exec*

84098 **CHANGE HISTORY**

84099 First released in Issue 4.

84100 **Issue 7**84101 Austin Group Interpretation 1003.1-2001 #196 is applied, changing the SYNOPSIS to allow `-p` to  
84102 be used with `-v` (or `-V`).

84103 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

84104 The *command* utility is moved from the User Portability Utilities option to the Base. User  
84105 Portability Utilities is now an option for interactive utilities.84106 The APPLICATION USAGE and EXAMPLES are revised to replace the non-standard  
84107 *getconf\_CS\_PATH* with *getconf PATH*.

84108 **NAME**

84109 compress — compress data

84110 **SYNOPSIS**84111 XSI `compress [-fv] [-b bits] [file...]`84112 `compress [-cfv] [-b bits] [file]`84113 **DESCRIPTION**84114 The *compress* utility shall attempt to reduce the size of the named files by using adaptive Lempel-  
84115 Ziv coding algorithm.84116 **Note:** Lempel-Ziv is US Patent 4464650, issued to William Eastman, Abraham Lempel, Jacob Ziv,  
84117 Martin Cohn on August 7th, 1984, and assigned to Sperry Corporation.84118 Lempel-Ziv-Welch compression is covered by US Patent 4558302, issued to Terry A. Welch on  
84119 December 10th, 1985, and assigned to Sperry Corporation.84120 On systems not supporting adaptive Lempel-Ziv coding algorithm, the input files shall not be  
84121 changed and an error value greater than two shall be returned. Except when the output is to the  
84122 standard output, each file shall be replaced by one with the extension *.Z*. If the invoking process  
84123 has appropriate privileges, the ownership, modes, access time, and modification time of the  
84124 original file are preserved. If appending the *.Z* to the filename would make the name exceed  
84125 {NAME\_MAX} bytes, the command shall fail. If no files are specified, the standard input shall be  
84126 compressed to the standard output.84127 **OPTIONS**84128 The *compress* utility shall conform to XBD [Section 12.2](#) (on page 216).

84129 The following options shall be supported:

84130 **-b *bits*** Specify the maximum number of bits to use in a code. For a conforming  
84131 application, the *bits* argument shall be:84132  $9 \leq bits \leq 14$ 84133 The implementation may allow *bits* values of greater than 14. The default is 14, 15,  
84134 or 16.84135 **-c** Cause *compress* to write to the standard output; the input file is not changed, and  
84136 no *.Z* files are created.84137 **-f** Force compression of *file*, even if it does not actually reduce the size of the file, or if  
84138 the corresponding *file.Z* file already exists. If the **-f** option is not given, and the  
84139 process is not running in the background, the user is prompted as to whether an  
84140 existing *file.Z* file should be overwritten. If the response is affirmative, the existing  
84141 file will be overwritten.84142 **-v** Write the percentage reduction of each file to standard error.84143 **OPERANDS**

84144 The following operand shall be supported:

84145 *file* A pathname of a file to be compressed.84146 **STDIN**84147 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '-'.

**INPUT FILES**

If *file* operands are specified, the input files contain the data to be compressed.

**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *compress*:

**LANG** Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

**LC\_ALL** If set to a non-empty string value, override the values of all the other internationalization variables.

**LC\_COLLATE**

Determine the locale for the behavior of ranges, equivalence classes, and multi-character collating elements used in the extended regular expression defined for the **yesexpr** locale keyword in the *LC\_MESSAGES* category.

**LC\_CTYPE**

Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments), the behavior of character classes used in the extended regular expression defined for the **yesexpr** locale keyword in the *LC\_MESSAGES* category.

**LC\_MESSAGES**

Determine the locale used to process affirmative responses, and the locale used to affect the format and contents of diagnostic messages, prompts, and the output from the **-v** option written to standard error.

**NLSPATH**

Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

**ASYNCHRONOUS EVENTS**

Default.

**STDOUT**

If no *file* operands are specified, or if a *file* operand is **'-'**, or if the **-c** option is specified, the standard output contains the compressed output.

**STDERR**

The standard error shall be used only for diagnostic and prompt messages and the output from **-v**.

**OUTPUT FILES**

The output files shall contain the compressed output. The format of compressed files is unspecified and interchange of such files between implementations (including access via unspecified file sharing mechanisms) is not required by POSIX.1-2008.

**EXTENDED DESCRIPTION**

None.

**EXIT STATUS**

The following exit values shall be returned:

0 Successful completion.

1 An error occurred.

2 One or more files were not compressed because they would have increased in size (and the **-f** option was not specified).

84190 >2 An error occurred.

#### 84191 CONSEQUENCES OF ERRORS

84192 The input file shall remain unmodified.

#### 84193 APPLICATION USAGE

84194 The amount of compression obtained depends on the size of the input, the number of *bits* per  
84195 code, and the distribution of common substrings. Typically, text such as source code or English is  
84196 reduced by 50-60%. Compression is generally much better than that achieved by Huffman  
84197 coding or adaptive Huffman coding (*compact*), and takes less time to compute.

84198 Although *compress* strictly follows the default actions upon receipt of a signal or when an error  
84199 occurs, some unexpected results may occur. In some implementations it is likely that a partially  
84200 compressed file is left in place, alongside its uncompressed input file. Since the general  
84201 operation of *compress* is to delete the uncompressed file only after the *.Z* file has been  
84202 successfully filled, an application should always carefully check the exit status of *compress* before  
84203 arbitrarily deleting files that have like-named neighbors with *.Z* suffixes.

84204 The limit of 14 on the *bits* option-argument is to achieve portability to all systems (within the  
84205 restrictions imposed by the lack of an explicit published file format). Some implementations  
84206 based on 16-bit architectures cannot support 15 or 16-bit uncompression.

#### 84207 EXAMPLES

84208 None.

#### 84209 RATIONALE

84210 None.

#### 84211 FUTURE DIRECTIONS

84212 None.

#### 84213 SEE ALSO

84214 *uncompress*, *zcat*

84215 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

#### 84216 CHANGE HISTORY

84217 First released in Issue 4.

#### 84218 Issue 6

84219 The normative text is reworded to avoid use of the term “must” for application requirements.

84220 An error case is added for systems not supporting adaptive Lempel-Ziv coding.

#### 84221 Issue 7

84222 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

84223 Austin Group Interpretation 1003.1-2001 #125 is applied, revising the ENVIRONMENT  
84224 VARIABLES section.

84225 **NAME**84226 `cp` — copy files84227 **SYNOPSIS**84228 `cp [-Pfp] source_file target_file`84229 `cp [-Pfp] source_file... target`84230 `cp -R [-H|-L|-P] [-fip] source_file... target`84231 **DESCRIPTION**

84232 The first synopsis form is denoted by two operands, neither of which are existing files of type  
 84233 directory. The `cp` utility shall copy the contents of *source\_file* (or, if *source\_file* is a file of type  
 84234 symbolic link, the contents of the file referenced by *source\_file*) to the destination path named by  
 84235 *target\_file*.

84236 The second synopsis form is denoted by two or more operands where the `-R` option is not  
 84237 specified and the first synopsis form is not applicable. It shall be an error if any *source\_file* is a file  
 84238 of type directory, if *target* does not exist, or if *target* does not name a directory. The `cp` utility shall  
 84239 copy the contents of each *source\_file* (or, if *source\_file* is a file of type symbolic link, the contents of  
 84240 the file referenced by *source\_file*) to the destination path named by the concatenation of *target*, a  
 84241 single `<slash>` character if *target* did not end in a `<slash>`, and the last component of *source\_file*.

84242 The third synopsis form is denoted by two or more operands where the `-R` option is specified.  
 84243 The `cp` utility shall copy each file in the file hierarchy rooted in each *source\_file* to a destination  
 84244 path named as follows:

- 84245 • If *target* exists and names an existing directory, the name of the corresponding destination  
 84246 path for each file in the file hierarchy shall be the concatenation of *target*, a single `<slash>`  
 84247 character if *target* did not end in a `<slash>`, and the pathname of the file relative to the  
 84248 directory containing *source\_file*.
- 84249 • If *target* does not exist and two operands are specified, the name of the corresponding  
 84250 destination path for *source\_file* shall be *target*; the name of the corresponding destination  
 84251 path for all other files in the file hierarchy shall be the concatenation of *target*, a `<slash>`  
 84252 character, and the pathname of the file relative to *source\_file*.

84253 It shall be an error if *target* does not exist and more than two operands are specified, or if *target*  
 84254 exists and does not name a directory.

84255 In the following description, the term *dest\_file* refers to the file named by the destination path.  
 84256 The term *source\_file* refers to the file that is being copied, whether specified as an operand or a  
 84257 file in a file hierarchy rooted in a *source\_file* operand. If *source\_file* is a file of type symbolic link:

- 84258 • If the `-R` option was not specified, `cp` shall take actions based on the type and contents of  
 84259 the file referenced by the symbolic link, and not by the symbolic link itself, unless the `-P`  
 84260 option was specified.
- 84261 • If the `-R` option was specified:
  - 84262 — If none of the options `-H`, `-L`, nor `-P` were specified, it is unspecified which of `-H`,  
 84263 `-L`, or `-P` will be used as a default.
  - 84264 — If the `-H` option was specified, `cp` shall take actions based on the type and contents of  
 84265 the file referenced by any symbolic link specified as a *source\_file* operand.
  - 84266 — If the `-L` option was specified, `cp` shall take actions based on the type and contents of  
 84267 the file referenced by any symbolic link specified as a *source\_file* operand or any  
 84268 symbolic links encountered during traversal of a file hierarchy.

- If the **-P** option was specified, *cp* shall copy any symbolic link specified as a *source\_file* operand and any symbolic links encountered during traversal of a file hierarchy, and shall not follow any symbolic links.

For each *source\_file*, the following steps shall be taken:

1. If *source\_file* references the same file as *dest\_file*, *cp* may write a diagnostic message to standard error; it shall do nothing more with *source\_file* and shall go on to any remaining files.
2. If *source\_file* is of type directory, the following steps shall be taken:
  - a. If the **-R** option was not specified, *cp* shall write a diagnostic message to standard error, do nothing more with *source\_file*, and go on to any remaining files.
  - b. If *source\_file* was not specified as an operand and *source\_file* is dot or dot-dot, *cp* shall do nothing more with *source\_file* and go on to any remaining files.
  - c. If *dest\_file* exists and it is a file type not specified by the System Interfaces volume of POSIX.1-2008, the behavior is implementation-defined.
  - d. If *dest\_file* exists and it is not of type directory, *cp* shall write a diagnostic message to standard error, do nothing more with *source\_file* or any files below *source\_file* in the file hierarchy, and go on to any remaining files.
  - e. If the directory *dest\_file* does not exist, it shall be created with file permission bits set to the same value as those of *source\_file*, modified by the file creation mask of the user if the **-p** option was not specified, and then bitwise-inclusively OR'ed with S\_IRWXU. If *dest\_file* cannot be created, *cp* shall write a diagnostic message to standard error, do nothing more with *source\_file*, and go on to any remaining files. It is unspecified if *cp* attempts to copy files in the file hierarchy rooted in *source\_file*.
  - f. The files in the directory *source\_file* shall be copied to the directory *dest\_file*, taking the four steps (1 to 4) listed here with the files as *source\_files*.
  - g. If *dest\_file* was created, its file permission bits shall be changed (if necessary) to be the same as those of *source\_file*, modified by the file creation mask of the user if the **-p** option was not specified.
  - h. The *cp* utility shall do nothing more with *source\_file* and go on to any remaining files.
3. If *source\_file* is of type regular file, the following steps shall be taken:
  - a. The behavior is unspecified if *dest\_file* exists and was written by a previous step. Otherwise, if *dest\_file* exists, the following steps shall be taken:
    - i. If the **-i** option is in effect, the *cp* utility shall write a prompt to the standard error and read a line from the standard input. If the response is not affirmative, *cp* shall do nothing more with *source\_file* and go on to any remaining files.
    - ii. A file descriptor for *dest\_file* shall be obtained by performing actions equivalent to the *open()* function defined in the System Interfaces volume of POSIX.1-2008 called using *dest\_file* as the *path* argument, and the bitwise-inclusive OR of O\_WRONLY and O\_TRUNC as the *oflag* argument.
    - iii. If the attempt to obtain a file descriptor fails and the **-f** option is in effect, *cp* shall attempt to remove the file by performing actions equivalent to the *unlink()* function defined in the System Interfaces volume of POSIX.1-2008



- 84313 called using *dest\_file* as the *path* argument. If this attempt succeeds, *cp* shall  
84314 continue with step 3b.
- 84315 b. If *dest\_file* does not exist, a file descriptor shall be obtained by performing actions  
84316 equivalent to the *open()* function defined in the System Interfaces volume of  
84317 POSIX.1-2008 called using *dest\_file* as the *path* argument, and the bitwise-inclusive  
84318 OR of *O\_WRONLY* and *O\_CREAT* as the *oflag* argument. The file permission bits  
84319 of *source\_file* shall be the *mode* argument.
- 84320 c. If the attempt to obtain a file descriptor fails, *cp* shall write a diagnostic message to  
84321 standard error, do nothing more with *source\_file*, and go on to any remaining files.
- 84322 d. The contents of *source\_file* shall be written to the file descriptor. Any write errors  
84323 shall cause *cp* to write a diagnostic message to standard error and continue to step  
84324 3e.
- 84325 e. The file descriptor shall be closed.
- 84326 f. The *cp* utility shall do nothing more with *source\_file*. If a write error occurred in  
84327 step 3d, it is unspecified if *cp* continues with any remaining files. If no write error  
84328 occurred in step 3d, *cp* shall go on to any remaining files.
- 84329 4. Otherwise, the **-R** option was specified, and the following steps shall be taken:
- 84330 a. The *dest\_file* shall be created with the same file type as *source\_file*.
- 84331 b. If *source\_file* is a file of type FIFO, the file permission bits shall be the same as those  
84332 of *source\_file*, modified by the file creation mask of the user if the **-p** option was not  
84333 specified. Otherwise, the permissions, owner ID, and group ID of *dest\_file* are  
84334 implementation-defined.
- 84335 If this creation fails for any reason, *cp* shall write a diagnostic message to standard  
84336 error, do nothing more with *source\_file*, and go on to any remaining files.
- 84337 c. If *source\_file* is a file of type symbolic link, and the options require the symbolic link  
84338 itself to be acted upon, the pathname contained in *dest\_file* shall be the same as the  
84339 pathname contained in *source\_file*.
- 84340 If this fails for any reason, *cp* shall write a diagnostic message to standard error, do  
84341 nothing more with *source\_file*, and go on to any remaining files.
- 84342 If the implementation provides additional or alternate access control mechanisms (see XBD  
84343 [Section 4.5](#), on page 108), their effect on copies of files is implementation-defined.

## 84344 OPTIONS

84345 The *cp* utility shall conform to XBD [Section 12.2](#) (on page 216).

84346 The following options shall be supported:

- 84347 **-f** If a file descriptor for a destination file cannot be obtained, as described in step  
84348 3.a.ii., attempt to unlink the destination file and proceed.
- 84349 **-H** Take actions based on the type and contents of the file referenced by any symbolic  
84350 link specified as a *source\_file* operand.
- 84351 **-i** Write a prompt to standard error before copying to any existing non-directory  
84352 destination file. If the response from the standard input is affirmative, the copy  
84353 shall be attempted; otherwise, it shall not.



|       |                              |                                                                                                                                                                                                                                      |
|-------|------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 84354 | –L                           | Take actions based on the type and contents of the file referenced by any symbolic link specified as a <i>source_file</i> operand or any symbolic links encountered during traversal of a file hierarchy.                            |
| 84355 |                              |                                                                                                                                                                                                                                      |
| 84356 |                              |                                                                                                                                                                                                                                      |
| 84357 | –P                           | Take actions on any symbolic link specified as a <i>source_file</i> operand or any symbolic link encountered during traversal of a file hierarchy.                                                                                   |
| 84358 |                              |                                                                                                                                                                                                                                      |
| 84359 | –p                           | Duplicate the following characteristics of each source file in the corresponding destination file:                                                                                                                                   |
| 84360 |                              |                                                                                                                                                                                                                                      |
| 84361 |                              | 1. The time of last data modification and time of last access. If this duplication fails for any reason, <i>cp</i> shall write a diagnostic message to standard error.                                                               |
| 84362 |                              |                                                                                                                                                                                                                                      |
| 84363 |                              | 2. The user ID and group ID. If this duplication fails for any reason, it is unspecified whether <i>cp</i> writes a diagnostic message to standard error.                                                                            |
| 84364 |                              |                                                                                                                                                                                                                                      |
| 84365 |                              | 3. The file permission bits and the S_ISUID and S_ISGID bits. Other, implementation-defined, bits may be duplicated as well. If this duplication fails for any reason, <i>cp</i> shall write a diagnostic message to standard error. |
| 84366 |                              |                                                                                                                                                                                                                                      |
| 84367 |                              |                                                                                                                                                                                                                                      |
| 84368 |                              | If the user ID or the group ID cannot be duplicated, the file permission bits S_ISUID and S_ISGID shall be cleared. If these bits are present in the source file but                                                                 |
| 84369 |                              | are not duplicated in the destination file, it is unspecified whether <i>cp</i> writes a                                                                                                                                             |
| 84370 |                              | diagnostic message to standard error.                                                                                                                                                                                                |
| 84371 |                              |                                                                                                                                                                                                                                      |
| 84372 |                              | The order in which the preceding characteristics are duplicated is unspecified. The                                                                                                                                                  |
| 84373 |                              | <i>dest_file</i> shall not be deleted if these characteristics cannot be preserved.                                                                                                                                                  |
| 84374 | –R                           | Copy file hierarchies.                                                                                                                                                                                                               |
| 84375 |                              | Specifying more than one of the mutually-exclusive options –H, –L, and –P shall not be                                                                                                                                               |
| 84376 |                              | considered an error. The last option specified shall determine the behavior of the utility.                                                                                                                                          |
| 84377 | <b>OPERANDS</b>              |                                                                                                                                                                                                                                      |
| 84378 |                              | The following operands shall be supported:                                                                                                                                                                                           |
| 84379 | <i>source_file</i>           | A pathname of a file to be copied. If a <i>source_file</i> operand is ‘–’, it shall refer to a                                                                                                                                       |
| 84380 |                              | file named –; implementations shall not treat it as meaning standard input.                                                                                                                                                          |
| 84381 | <i>target_file</i>           | A pathname of an existing or nonexistent file, used for the output when a single                                                                                                                                                     |
| 84382 |                              | file is copied. If a <i>target_file</i> operand is ‘–’, it shall refer to a file named –;                                                                                                                                            |
| 84383 |                              | implementations shall not treat it as meaning standard output.                                                                                                                                                                       |
| 84384 | <i>target</i>                | A pathname of a directory to contain the copied files.                                                                                                                                                                               |
| 84385 | <b>STDIN</b>                 |                                                                                                                                                                                                                                      |
| 84386 |                              | The standard input shall be used to read an input line in response to each prompt specified in                                                                                                                                       |
| 84387 |                              | the STDERR section. Otherwise, the standard input shall not be used.                                                                                                                                                                 |
| 84388 | <b>INPUT FILES</b>           |                                                                                                                                                                                                                                      |
| 84389 |                              | The input files specified as operands may be of any file type.                                                                                                                                                                       |
| 84390 | <b>ENVIRONMENT VARIABLES</b> |                                                                                                                                                                                                                                      |
| 84391 |                              | The following environment variables shall affect the execution of <i>cp</i> :                                                                                                                                                        |
| 84392 | LANG                         | Provide a default value for the internationalization variables that are unset or null.                                                                                                                                               |
| 84393 |                              | (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization                                                                                                                                        |
| 84394 |                              | variables used to determine the values of locale categories.)                                                                                                                                                                        |

|       |                               |                                                                                                                                                                                                                                                                                                                                                          |
|-------|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 84395 | <b>LC_ALL</b>                 | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                 |
| 84396 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 84397 | <b>LC_COLLATE</b>             | Determine the locale for the behavior of ranges, equivalence classes, and multi-character collating elements used in the extended regular expression defined for the <b>yesexpr</b> locale keyword in the <b>LC_MESSAGES</b> category.                                                                                                                   |
| 84398 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 84399 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 84400 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 84401 | <b>LC_CTYPE</b>               | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files) and the behavior of character classes used in the extended regular expression defined for the <b>yesexpr</b> locale keyword in the <b>LC_MESSAGES</b> category. |
| 84402 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 84403 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 84404 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 84405 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 84406 | <b>LC_MESSAGES</b>            | Determine the locale used to process affirmative responses, and the locale used to affect the format and contents of diagnostic messages and prompts written to standard error.                                                                                                                                                                          |
| 84407 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 84408 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 84409 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 84410 | XSI <b>NLSPATH</b>            | Determine the location of message catalogs for the processing of <b>LC_MESSAGES</b> .                                                                                                                                                                                                                                                                    |
| 84411 | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                                                                                                                                                                                                                          |
| 84412 |                               | Default.                                                                                                                                                                                                                                                                                                                                                 |
| 84413 | <b>STDOUT</b>                 |                                                                                                                                                                                                                                                                                                                                                          |
| 84414 |                               | Not used.                                                                                                                                                                                                                                                                                                                                                |
| 84415 | <b>STDERR</b>                 |                                                                                                                                                                                                                                                                                                                                                          |
| 84416 |                               | A prompt shall be written to standard error under the conditions specified in the <b>DESCRIPTION</b> section. The prompt shall contain the destination pathname, but its format is otherwise unspecified. Otherwise, the standard error shall be used only for diagnostic messages.                                                                      |
| 84417 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 84418 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 84419 | <b>OUTPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                          |
| 84420 |                               | The output files may be of any type.                                                                                                                                                                                                                                                                                                                     |
| 84421 | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                                                                                                                                                                                                                          |
| 84422 |                               | None.                                                                                                                                                                                                                                                                                                                                                    |
| 84423 | <b>EXIT STATUS</b>            |                                                                                                                                                                                                                                                                                                                                                          |
| 84424 |                               | The following exit values shall be returned:                                                                                                                                                                                                                                                                                                             |
| 84425 |                               | 0 All files were copied successfully.                                                                                                                                                                                                                                                                                                                    |
| 84426 |                               | >0 An error occurred.                                                                                                                                                                                                                                                                                                                                    |
| 84427 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                                                                                                                                                                                                                          |
| 84428 |                               | If <b>cp</b> is prematurely terminated by a signal or error, files or file hierarchies may be only partially copied and files and directories may have incorrect permissions or access and modification times.                                                                                                                                           |
| 84429 |                               |                                                                                                                                                                                                                                                                                                                                                          |
| 84430 |                               |                                                                                                                                                                                                                                                                                                                                                          |

**APPLICATION USAGE**

The set-user-ID and set-group-ID bits are explicitly cleared when files are created. This is to prevent users from creating programs that are set-user-ID or set-group-ID to them when copying files or to make set-user-ID or set-group-ID files accessible to new groups of users. For example, if a file is set-user-ID and the copy has a different group ID than the source, a new group of users has execute permission to a set-user-ID program than did previously. In particular, this is a problem for superusers copying users' trees.

**EXAMPLES**

None.

**RATIONALE**

The `-i` option exists on BSD systems, giving applications and users a way to avoid accidentally removing files when copying. Although the 4.3 BSD version does not prompt if the standard input is not a terminal, the standard developers decided that use of `-i` is a request for interaction, so when the destination path exists, the utility takes instructions from whatever responds on standard input.

The exact format of the interactive prompts is unspecified. Only the general nature of the contents of prompts are specified because implementations may desire more descriptive prompts than those used on historical implementations. Therefore, an application using the `-i` option relies on the system to provide the most suitable dialog directly with the user, based on the behavior specified.

The `-p` option is historical practice on BSD systems, duplicating the time of last data modification and time of last access. This volume of POSIX.1-2008 extends it to preserve the user and group IDs, as well as the file permissions. This requirement has obvious problems in that the directories are almost certainly modified after being copied. This volume of POSIX.1-2008 requires that the modification times be preserved. The statement that the order in which the characteristics are duplicated is unspecified is to permit implementations to provide the maximum amount of security for the user. Implementations should take into account the obvious security issues involved in setting the owner, group, and mode in the wrong order or creating files with an owner, group, or mode different from the final value.

It is unspecified whether `cp` writes diagnostic messages when the user and group IDs cannot be set due to the widespread practice of users using `-p` to duplicate some portion of the file characteristics, indifferent to the duplication of others. Historic implementations only write diagnostic messages on errors other than [EPERM].

Earlier versions of this standard included support for the `-r` option to copy file hierarchies. The `-r` option is historical practice on BSD and BSD-derived systems. This option is no longer specified by POSIX.1-2008 but may be present in some implementations. The `-R` option was added as a close synonym to the `-r` option, selected for consistency with all other options in this volume of POSIX.1-2008 that do recursive directory descent.

The difference between `-R` and the removed `-r` option is in the treatment by `cp` of file types other than regular and directory. It was implementation-defined how the `-` option treated special files to allow both historical implementations and those that chose to support `-r` with the same abilities as `-R` defined by this volume of POSIX.1-2008. The original `-r` flag, for historic reasons, did not handle special files any differently from regular files, but always read the file and copied its contents. This had obvious problems in the presence of special file types; for example, character devices, FIFOs, and sockets.

When a failure occurs during the copying of a file hierarchy, `cp` is required to attempt to copy files that are on the same level in the hierarchy or above the file where the failure occurred. It is unspecified if `cp` shall attempt to copy files below the file where the failure occurred (which

cannot succeed in any case).

Permissions, owners, and groups of created special file types have been deliberately left as implementation-defined. This is to allow systems to satisfy special requirements (for example, allowing users to create character special devices, but requiring them to be owned by a certain group). In general, it is strongly suggested that the permissions, owner, and group be the same as if the user had run the historical *mknod*, *ln*, or other utility to create the file. It is also probable that additional privileges are required to create block, character, or other implementation-defined special file types.

Additionally, the *-p* option explicitly requires that all set-user-ID and set-group-ID permissions be discarded if any of the owner or group IDs cannot be set. This is to keep users from unintentionally giving away special privilege when copying programs.

When creating regular files, historical versions of *cp* use the mode of the source file as modified by the file mode creation mask. Other choices would have been to use the mode of the source file unmodified by the creation mask or to use the same mode as would be given to a new file created by the user (plus the execution bits of the source file) and then modify it by the file mode creation mask. In the absence of any strong reason to change historic practice, it was in large part retained.

When creating directories, historical versions of *cp* use the mode of the source directory, plus read, write, and search bits for the owner, as modified by the file mode creation mask. This is done so that *cp* can copy trees where the user has read permission, but the owner does not. A side-effect is that if the file creation mask denies the owner permissions, *cp* fails. Also, once the copy is done, historical versions of *cp* set the permissions on the created directory to be the same as the source directory, unmodified by the file creation mask.

This behavior has been modified so that *cp* is always able to create the contents of the directory, regardless of the file creation mask. After the copy is done, the permissions are set to be the same as the source directory, as modified by the file creation mask. This latter change from historical behavior is to prevent users from accidentally creating directories with permissions beyond those they would normally set and for consistency with the behavior of *cp* in creating files.

It is not a requirement that *cp* detect attempts to copy a file to itself; however, implementations are strongly encouraged to do so. Historical implementations have detected the attempt in most cases.

There are two methods of copying subtrees in this volume of POSIX.1-2008. The other method is described as part of the *pax* utility (see *pax*). Both methods are historical practice. The *cp* utility provides a simpler, more intuitive interface, while *pax* offers a finer granularity of control. Each provides additional functionality to the other; in particular, *pax* maintains the hard-link structure of the hierarchy, while *cp* does not. It is the intention of the standard developers that the results be similar (using appropriate option combinations in both utilities). The results are not required to be identical; there seemed insufficient gain to applications to balance the difficulty of implementations having to guarantee that the results would be exactly identical.

The wording allowing *cp* to copy a directory to implementation-defined file types not specified by the System Interfaces volume of POSIX.1-2008 is provided so that implementations supporting symbolic links are not required to prohibit copying directories to symbolic links. Other extensions to the System Interfaces volume of POSIX.1-2008 file types may need to use this loophole as well.

**84523 FUTURE DIRECTIONS**

84524 None.

**84525 SEE ALSO**

84526 *mv, find, ln, pax*

84527 XBD [Section 4.5](#) (on page 108), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

84528 XSH *open()*, *unlink()*

**84529 CHANGE HISTORY**

84530 First released in Issue 2.

**84531 Issue 6**

84532 The **-r** option is marked obsolescent.

84533 The new options **-H**, **-L**, and **-P** are added to align with the IEEE P1003.2b draft standard. These  
84534 options affect the processing of symbolic links.

84535 IEEE PASC Interpretation 1003.2 #194 is applied, adding a description of the **-P** option.

84536 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/18 is applied, correcting an error in the  
84537 SEE ALSO section.

**84538 Issue 7**

84539 Austin Group Interpretation 1003.1-2001 #126 is applied, changing the description of the  
84540 *LC\_MESSAGES* environment variable.

84541 Austin Group Interpretations 1003.1-2001 #092, #164, #165, and #168 are applied.

84542 SD5-XCU-ERN-31 and SD5-XCU-ERN-42 are applied, updating the DESCRIPTION.

84543 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

84544 SD5-XCU-ERN-102 is applied, clarifying the **-i** option within the OPTIONS section.

84545 The obsolescent **-r** option is removed.

84546 The **-P** option is added to the SYNOPSIS and to the DESCRIPTION with respect to the **-R**  
84547 option.

84548 **NAME**

84549 crontab — schedule periodic background work

84550 **SYNOPSIS**84551 crontab [*file*]84552 UP crontab [**-e** | **-l** | **-r**]84553 **DESCRIPTION**

84554 UP The *crontab* utility shall create, replace, or edit a user's crontab entry; a crontab entry is a list of  
 84555 commands and the times at which they shall be executed. The new crontab entry can be input by  
 84556 UP specifying *file* or input from standard input if no *file* operand is specified, or by using an editor,  
 84557 if **-e** is specified.

84558 Upon execution of a command from a crontab entry, the implementation shall supply a default  
 84559 environment, defining at least the following environment variables:

84560 *HOME* A pathname of the user's home directory.

84561 *LOGNAME* The user's login name.

84562 *PATH* A string representing a search path guaranteed to find all of the standard utilities.

84563 *SHELL* A pathname of the command interpreter. When *crontab* is invoked as specified by  
 84564 this volume of POSIX.1-2008, the value shall be a pathname for *sh*.

84565 The values of these variables when *crontab* is invoked as specified by this volume of  
 84566 POSIX.1-2008 shall not affect the default values provided when the scheduled command is run.

84567 If standard output and standard error are not redirected by commands executed from the  
 84568 crontab entry, any generated output or errors shall be mailed, via an implementation-defined  
 84569 method, to the user.

84570 XSI Users shall be permitted to use *crontab* if their names appear in the file **cron.allow** which is  
 84571 located in an implementation-defined directory. If that file does not exist, the file **cron.deny**,  
 84572 which is located in an implementation-defined directory, shall be checked to determine whether  
 84573 the user shall be denied access to *crontab*. If neither file exists, only a process with appropriate  
 84574 privileges shall be allowed to submit a job. If only **cron.deny** exists and is empty, global usage  
 84575 shall be permitted. The **cron.allow** and **cron.deny** files shall consist of one user name per line.

84576 **OPTIONS**

84577 The *crontab* utility shall conform to XBD [Section 12.2](#) (on page 216).

84578 The following options shall be supported:

84579 UP **-e** Edit a copy of the invoking user's crontab entry, or create an empty entry to edit if  
 84580 the crontab entry does not exist. When editing is complete, the entry shall be  
 84581 installed as the user's crontab entry.

84582 **-l** (The letter ell.) List the invoking user's crontab entry.

84583 **-r** Remove the invoking user's crontab entry.

84584 **OPERANDS**

84585 The following operand shall be supported:

84586 *file* The pathname of a file that contains specifications, in the format defined in the  
 84587 INPUT FILES section, for crontab entries.



**STDIN**

See the INPUT FILES section.

**INPUT FILES**

In the POSIX locale, the user or application shall ensure that a crontab entry is a text file consisting of lines of six fields each. The fields shall be separated by <blank> characters. The first five fields shall be integer patterns that specify the following:

1. Minute [0,59]
2. Hour [0,23]
3. Day of the month [1,31]
4. Month of the year [1,12]
5. Day of the week ([0,6] with 0=Sunday)

Each of these patterns can be either an <asterisk> (meaning all valid values), an element, or a list of elements separated by <comma> characters. An element shall be either a number or two numbers separated by a <hyphen-minus> (meaning an inclusive range). The specification of days can be made by two fields (day of the month and day of the week). If month, day of month, and day of week are all <asterisk> characters, every day shall be matched. If either the month or day of month is specified as an element or list, but the day of week is an <asterisk>, the month and day of month fields shall specify the days that match. If both month and day of month are specified as an <asterisk>, but day of week is an element or list, then only the specified days of the week match. Finally, if either the month or day of month is specified as an element or list, and the day of week is also specified as an element or list, then any day matching either the month and day of month, or the day of week, shall be matched.

The sixth field of a line in a crontab entry is a string that shall be executed by *sh* at the specified times. A <percent-sign> character in this field shall be translated to a <newline>. Any character preceded by a <backslash> (including the ' % ') shall cause that character to be treated literally. Only the first line (up to a ' % ' or end-of-line) of the command field shall be executed by the command interpreter. The other lines shall be made available to the command as standard input.

Blank lines and those whose first non-<blank> is ' # ' shall be ignored.

XSI

The text files **cron.allow** and **cron.deny**, which are located in an implementation-defined directory, shall contain zero or more user names, one per line, of users who are, respectively, authorized or denied access to the service underlying the *crontab* utility.

**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *crontab*:

- |                 |                                                                                                                                                                                                                                                    |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>EDITOR</i>   | Determine the editor to be invoked when the <b>-e</b> option is specified. The default editor shall be <i>vi</i> .                                                                                                                                 |
| <i>LANG</i>     | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.) |
| <i>LC_ALL</i>   | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                           |
| <i>LC_CTYPE</i> | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files).                                                          |

84631 *LC\_MESSAGES*  
 84632 Determine the locale that should be used to affect the format and contents of  
 84633 diagnostic messages written to standard error.

84634 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

84635 **ASYNCHRONOUS EVENTS**  
 84636 Default.

84637 **STDOUT**  
 84638 If the *-l* option is specified, the crontab entry shall be written to the standard output.

84639 **STDERR**  
 84640 The standard error shall be used only for diagnostic messages.

84641 **OUTPUT FILES**  
 84642 None.

84643 **EXTENDED DESCRIPTION**  
 84644 None.

84645 **EXIT STATUS**  
 84646 The following exit values shall be returned:  
 84647 0 Successful completion.  
 84648 >0 An error occurred.

84649 **CONSEQUENCES OF ERRORS**  
 84650 UP The user's crontab entry is not submitted, removed, *edited*, or listed.

84651 **APPLICATION USAGE**  
 84652 The format of the crontab entry shown here is guaranteed only for the POSIX locale. Other  
 84653 cultures may be supported with substantially different interfaces, although implementations are  
 84654 encouraged to provide comparable levels of functionality.

84655 The default settings of the *HOME*, *LOGNAME*, *PATH*, and *SHELL* variables that are given to the  
 84656 scheduled job are not affected by the settings of those variables when *crontab* is run; as stated,  
 84657 they are defaults. The text about "invoked as specified by this volume of POSIX.1-2008" means  
 84658 that the implementation may provide extensions that allow these variables to be affected at  
 84659 runtime, but that the user has to take explicit action in order to access the extension, such as give  
 84660 a new option flag or modify the format of the crontab entry.

84661 A typical user error is to type only *crontab*; this causes the system to wait for the new crontab  
 84662 entry on standard input. If end-of-file is typed (generally <control>-D), the crontab entry is  
 84663 replaced by an empty file. In this case, the user should type the interrupt character, which  
 84664 prevents the crontab entry from being replaced.

84665 **EXAMPLES**

84666 1. Clean up **core** files every weekday morning at 3:15 am:  
 84667 15 3 \* \* 1-5 find "\$HOME" -name core -exec rm -f {} + 2>/dev/null

84668 2. Mail a birthday greeting:  
 84669 0 12 14 2 \* mailx john%Happy Birthday!%Time for lunch.

84670 3. As an example of specifying the two types of days:  
 84671 0 0 1,15 \* 1



84672 would run a command on the first and fifteenth of each month, as well as on every  
84673 Monday. To specify days by only one field, the other field should be set to '\*'; for  
84674 example:

84675 0 0 \* \* 1

84676 would run a command only on Mondays.

#### 84677 RATIONALE

84678 All references to a *cron* daemon and to *cron files* have been omitted. Although historical  
84679 implementations have used this arrangement, there is no reason to limit future implementations.

84680 This description of *crontab* is designed to support only users with normal privileges. The format  
84681 of the input is based on the System V *crontab*; however, there is no requirement here that the  
84682 actual system database used by the *cron* daemon (or a similar mechanism) use this format  
84683 internally. For example, systems derived from BSD are likely to have an additional field  
84684 appended that indicates the user identity to be used when the job is submitted.

84685 The *-e* option was adopted from the SVID as a user convenience, although it does not exist in all  
84686 historical implementations.

#### 84687 FUTURE DIRECTIONS

84688 None.

#### 84689 SEE ALSO

84690 *at*

84691 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

#### 84692 CHANGE HISTORY

84693 First released in Issue 2.

#### 84694 Issue 6

84695 This utility is marked as part of the User Portability Utilities option.

84696 The normative text is reworded to avoid use of the term “must” for application requirements.

#### 84697 Issue 7

84698 The *crontab* utility (except for the *-e* option) is moved from the User Portability Utilities option  
84699 to the Base. User Portability Utilities is now an option for interactive utilities.

84700 SD5-XCU-ERN-95 is applied, removing the references to fixed locations for the files referenced  
84701 by the *crontab* utility.

84702 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

84703 The first example is changed to remove the unreliable use of *find | xargs*.

84704 **NAME**

84705       csplit — split files based on context

84706 **SYNOPSIS**84707       csplit [-ks] [-f *prefix*] [-n *number*] *file arg...*84708 **DESCRIPTION**84709       The *csplit* utility shall read the file named by the *file* operand, write all or part of that file into  
84710       other files as directed by the *arg* operands, and write the sizes of the files.84711 **OPTIONS**84712       The *csplit* utility shall conform to XBD [Section 12.2](#) (on page 216).

84713       The following options shall be supported:

84714       **-f** *prefix*     Name the created files *prefix00*, *prefix01*, ..., *prefixn*. The default is *xx00* ... *xxn*. If  
84715       the *prefix* argument would create a filename exceeding {NAME\_MAX} bytes, an  
84716       error shall result, *csplit* shall exit with a diagnostic message, and no files shall be  
84717       created.84718       **-k**            Leave previously created files intact. By default, *csplit* shall remove created files if  
84719       an error occurs.84720       **-n** *number*    Use *number* decimal digits to form filenames for the file pieces. The default shall be  
84721       2.84722       **-s**            Suppress the output of file size messages.84723 **OPERANDS**

84724       The following operands shall be supported:

84725       *file*           The pathname of a text file to be split. If *file* is '-', the standard input shall be  
84726       used.84727       Each *arg* operand can be one of the following:84728       /*regexp*/[*offset*]84729       A file shall be created using the content of the lines from the current line up to, but  
84730       not including, the line that results from the evaluation of the regular expression  
84731       with *offset*, if any, applied. The regular expression *regexp* shall follow the rules for  
84732       basic regular expressions described in XBD [Section 9.3](#) (on page 183). The  
84733       application shall use the sequence "\/" to specify a <slash> character within the  
84734       *regexp*. The optional offset shall be a positive or negative integer value representing a  
84735       number of lines. A positive integer value can be preceded by '+'. If the selection  
84736       of lines from an *offset* expression of this type would create a file with zero lines, or  
84737       one with greater than the number of lines left in the input file, the results are  
84738       unspecified. After the section is created, the current line shall be set to the line that  
84739       results from the evaluation of the regular expression with any offset applied. If the  
84740       current line is the first line in the file and a regular expression operation has not yet  
84741       been performed, the pattern match of *regexp* shall be applied from the current line to  
84742       the end of the file. Otherwise, the pattern match of *regexp* shall be applied from the  
84743       line following the current line to the end of the file.84744       %*regexp*%[*offset*]84745       Equivalent to /*regexp*/[*offset*], except that no file shall be created for the selected  
84746       section of the input file. The application shall use the sequence "%%" to specify a  
84747       <percent-sign> character within the *regexp*.

84748 *line\_no* Create a file from the current line up to (but not including) the line number *line\_no*.  
 84749 Lines in the file shall be numbered starting at one. The current line becomes  
 84750 *line\_no*.

84751 {*num*} Repeat operand. This operand can follow any of the operands described  
 84752 previously. If it follows a *rexp* type operand, that operand shall be applied *num*  
 84753 more times. If it follows a *line\_no* operand, the file shall be split every *line\_no* lines,  
 84754 *num* times, from that point.

84755 An error shall be reported if an operand does not reference a line between the current position  
 84756 and the end of the file.

## 84757 STDIN

84758 See the INPUT FILES section.

## 84759 INPUT FILES

84760 The input file shall be a text file.

## 84761 ENVIRONMENT VARIABLES

84762 The following environment variables shall affect the execution of *csplit*:

84763 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 84764 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 84765 variables used to determine the values of locale categories.)

84766 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 84767 internationalization variables.

84768 *LC\_COLLATE*  
 84769 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 84770 character collating elements within regular expressions.

84771 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 84772 characters (for example, single-byte as opposed to multi-byte characters in  
 84773 arguments and input files) and the behavior of character classes within regular  
 84774 expressions.

84775 *LC\_MESSAGES*  
 84776 Determine the locale that should be used to affect the format and contents of  
 84777 diagnostic messages written to standard error.

84778 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 84779 ASYNCHRONOUS EVENTS

84780 If the *-k* option is specified, created files shall be retained. Otherwise, the default action occurs.

## 84781 STDOUT

84782 Unless the *-s* option is used, the standard output shall consist of one line per file created, with a  
 84783 format as follows:

84784 "%d\n", <file size in bytes>

## 84785 STDERR

84786 The standard error shall be used only for diagnostic messages.

## 84787 OUTPUT FILES

84788 The output files shall contain portions of the original input file; otherwise, unchanged.

84789 **EXTENDED DESCRIPTION**

84790 None.

84791 **EXIT STATUS**

84792 The following exit values shall be returned:

84793 0 Successful completion.

84794 &gt;0 An error occurred.

84795 **CONSEQUENCES OF ERRORS**

84796 By default, created files shall be removed if an error occurs. When the **-k** option is specified,  
 84797 created files shall not be removed if an error occurs.

84798 **APPLICATION USAGE**

84799 None.

84800 **EXAMPLES**84801 1. This example creates four files, **cobol00** ... **cobol03**:84802 `csplit -f cobol file '/procedure division/' /par5./ /par16./`

84803 After editing the split files, they can be recombined as follows:

84804 `cat cobol0[0-3] > file`

84805 Note that this example overwrites the original file.

84806 2. This example would split the file after the first 99 lines, and every 100 lines thereafter, up  
 84807 to 9999 lines; this is because lines in the file are numbered from 1 rather than zero, for  
 84808 historical reasons:

84809 `csplit -k file 100 {99}`

84810 3. Assuming that **prog.c** follows the C-language coding convention of ending routines with  
 84811 a '}' at the beginning of the line, this example creates a file containing each separate C  
 84812 routine (up to 21) in **prog.c**:

84813 `csplit -k prog.c '%main(%' '/^}'+1' {20}`84814 **RATIONALE**84815 The **-n** option was added to extend the range of filenames that could be handled.

84816 Consideration was given to adding a **-a** flag to use the alphabetic filename generation used by  
 84817 the historical *split* utility, but the functionality added by the **-n** option was deemed to make  
 84818 alphabetic naming unnecessary.

84819 **FUTURE DIRECTIONS**

84820 None.

84821 **SEE ALSO**84822 *sed*, *split*84823 XBD [Chapter 8](#) (on page 173), [Section 9.3](#) (on page 183), [Section 12.2](#) (on page 216)84824 **CHANGE HISTORY**

84825 First released in Issue 2.

84826 **Issue 5**

84827 The FUTURE DIRECTIONS section is added.

84828 **Issue 6**

84829 This utility is marked as part of the User Portability Utilities option.

84830 The APPLICATION USAGE section is added.

84831 The description of regular expression operands is changed to align with the IEEE P1003.2b draft  
84832 standard.

84833 The normative text is reworded to avoid use of the term “must” for application requirements.

84834 **Issue 7**

84835 The *csplit* utility is moved from the User Portability Utilities option to the Base. User Portability  
84836 Utilities is now an option for interactive utilities.

84837 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

84838 The SYNOPSIS and OPERANDS sections are revised to use a single *arg* to split a file into two  
84839 pieces.

84840 **NAME**84841 `ctags` — create a tags file (**DEVELOPMENT**, **FORTTRAN**)84842 **SYNOPSIS**84843 SD `ctags [-a] [-f tagsfile] pathname...`84844 `ctags -x pathname...`84845 **DESCRIPTION**

84846 The *ctags* utility shall be provided on systems that support the the Software Development  
 84847 Utilities option, and either or both of the C-Language Development Utilities option and  
 84848 FORTRAN Development Utilities option. On other systems, it is optional.

84849 The *ctags* utility shall write a *tagsfile* or an index of objects from C-language or FORTRAN source  
 84850 files specified by the *pathname* operands. The *tagsfile* shall list the locators of language-specific  
 84851 objects within the source files. A locator consists of a name, *pathname*, and either a search  
 84852 pattern or a line number that can be used in searching for the object definition. The objects that  
 84853 shall be recognized are specified in the EXTENDED DESCRIPTION section.

84854 **OPTIONS**84855 The *ctags* utility shall conform to XBD [Section 12.2](#) (on page 216).

84856 The following options shall be supported:

84857 **-a** Append to *tagsfile*.

84858 **-f** *tagsfile* Write the object locator lists into *tagsfile* instead of the default file named **tags** in the  
 84859 current directory.

84860 **-x** Produce a list of object names, the line number, and filename in which each is  
 84861 defined, as well as the text of that line, and write this to the standard output. A  
 84862 *tagsfile* shall not be created when **-x** is specified.

84863 **OPERANDS**84864 The following *pathname* operands are supported:

84865 *file.c* Files with basenames ending with the **.c** suffix shall be treated as C-language  
 84866 source code. Such files that are not valid input to *c99* produce unspecified results.

84867 *file.h* Files with basenames ending with the **.h** suffix shall be treated as C-language  
 84868 source code. Such files that are not valid input to *c99* produce unspecified results.

84869 *file.f* Files with basenames ending with the **.f** suffix shall be treated as FORTRAN-  
 84870 language source code. Such files that are not valid input to *fort77* produce  
 84871 unspecified results.

84872 The handling of other files is implementation-defined.

84873 **STDIN**

84874 See the INPUT FILES section.

84875 **INPUT FILES**

84876 The input files shall be text files containing source code in the language indicated by the  
 84877 operand filename suffixes.

**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *ctags*:

**LANG** Provide a default value for the internationalization variables that are unset or null. (See XBD Section 8.2 (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

**LC\_ALL** If set to a non-empty string value, override the values of all the other internationalization variables.

**LC\_COLLATE**

Determine the order in which output is sorted for the *-x* option. The POSIX locale determines the order in which the *tagsfile* is written.

**LC\_CTYPE**

Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files). When processing C-language source code, if the locale is not compatible with the C locale described by the ISO C standard, the results are unspecified.

**LC\_MESSAGES**

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

**XSI NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

**ASYNCHRONOUS EVENTS**

Default.

**STDOUT**

The list of object name information produced by the *-x* option shall be written to standard output in the following format:

"%s %d %s %s", <object-name>, <line-number>, <filename>, <text>

where <text> is the text of line <line-number> of file <filename>.

**STDERR**

The standard error shall be used only for diagnostic messages.

**OUTPUT FILES**

When the *-x* option is not specified, the format of the output file shall be:

"%s\t%s\t/%s/\n", <identifier>, <filename>, <pattern>

where <pattern> is a search pattern that could be used by an editor to find the defining instance of <identifier> in <filename> (where *defining instance* is indicated by the declarations listed in the EXTENDED DESCRIPTION).

An optional <circumflex> ('^') can be added as a prefix to <pattern>, and an optional <dollar-sign> can be appended to <pattern> to indicate that the pattern is anchored to the beginning (end) of a line of text. Any <slash> or <backslash> characters in <pattern> shall be preceded by a <backslash> character. The anchoring <circumflex>, <dollar-sign>, and escaping <backslash> characters shall not be considered part of the search pattern. All other characters in the search pattern shall be considered literal characters.

84918 An alternative format is:

84919 "%s\t%s\t%s?\n", <identifier>, <filename>, <pattern>

84920 which is identical to the first format except that <slash> characters in <pattern> shall not be  
84921 preceded by escaping <backslash> characters, and <question-mark> characters in <pattern>  
84922 shall be preceded by <backslash> characters.

84923 A second alternative format is:

84924 "%s\t%s\t%d\n", <identifier>, <filename>, <lineno>

84925 where <lineno> is a decimal line number that could be used by an editor to find <identifier> in  
84926 <filename>.

84927 Neither alternative format shall be produced by *ctags* when it is used as described by  
84928 POSIX.1-2008, but the standard utilities that process tags files shall be able to process those  
84929 formats as well as the first format.

84930 In any of these formats, the file shall be sorted by identifier, based on the collation sequence in  
84931 the POSIX locale.

## 84932 EXTENDED DESCRIPTION

84933 If the operand identifies C-language source, the *ctags* utility shall attempt to produce an output  
84934 line for each of the following objects:

- 84935 • Function definitions
- 84936 • Type definitions
- 84937 • Macros with arguments

84938 It may also produce output for any of the following objects:

- 84939 • Function prototypes
- 84940 • Structures
- 84941 • Unions
- 84942 • Global variable definitions
- 84943 • Enumeration types
- 84944 • Macros without arguments
- 84945 • **#define** statements
- 84946 • **#line** statements

84947 Any **#if** and **#ifdef** statements shall produce no output. The tag **main** is treated specially in C  
84948 programs. The tag formed shall be created by prefixing **M** to the name of the file, with the  
84949 trailing **.c**, and leading pathname components (if any) removed.

84950 On systems that do not support the C-Language Development Utilities option, *ctags* produces  
84951 unspecified results for C-language source code files. It should write to standard error a message  
84952 identifying this condition and cause a non-zero exit status to be produced.

84953 If the operand identifies FORTRAN source, the *ctags* utility shall produce an output line for each  
84954 function definition. It may also produce output for any of the following objects:

- 84955 • Subroutine definitions



- 84956 • COMMON statements
- 84957 • PARAMETER statements
- 84958 • DATA and BLOCK DATA statements
- 84959 • Statement numbers

84960 On systems that do not support the FORTRAN Development Utilities option, *ctags* produces  
 84961 unspecified results for FORTRAN source code files. It should write to standard error a message  
 84962 identifying this condition and cause a non-zero exit status to be produced.

84963 It is implementation-defined what other objects (including duplicate identifiers) produce output.

#### 84964 EXIT STATUS

84965 The following exit values shall be returned:

- 84966 0 Successful completion.
- 84967 >0 An error occurred.

#### 84968 CONSEQUENCES OF ERRORS

84969 Default.

#### 84970 APPLICATION USAGE

84971 The output with `-x` is meant to be a simple index that can be written out as an off-line readable  
 84972 function index. If the input files to *ctags* (such as `.c` files) were not created using the same locale  
 84973 as that in effect when *ctags -x* is run, results might not be as expected.

84974 The description of C-language processing says “attempts to” because the C language can be  
 84975 greatly confused, especially through the use of `#defines`, and this utility would be of no use if  
 84976 the real C preprocessor were run to identify them. The output from *ctags* may be fooled and  
 84977 incorrect for various constructs.

#### 84978 EXAMPLES

84979 None.

#### 84980 RATIONALE

84981 The option list was significantly reduced from that provided by historical implementations. The  
 84982 `-F` option was omitted as redundant, since it is the default. The `-B` option was omitted as being  
 84983 of very limited usefulness. The `-t` option was omitted since the recognition of `typedefs` is now  
 84984 required for C source files. The `-u` option was omitted because the update function was judged  
 84985 to be not only inefficient, but also rarely needed.

84986 An early proposal included a `-w` option to suppress warning diagnostics. Since the types of such  
 84987 diagnostics could not be described, the option was omitted as being not useful.

84988 The text for `LC_CTYPE` about compatibility with the C locale acknowledges that the ISO C  
 84989 standard imposes requirements on the locale used to process C source. This could easily be a  
 84990 superset of that known as “the C locale” by way of implementation extensions, or one of a few  
 84991 alternative locales for systems supporting different codesets. No statement is made for  
 84992 FORTRAN because the ANSI X3.9-1978 standard (FORTRAN 77) does not (yet) define a similar  
 84993 locale concept. However, a general rule in this volume of POSIX.1-2008 is that any time that  
 84994 locales do not match (preparing a file for one locale and processing it in another), the results are  
 84995 suspect.

84996 The collation sequence of the tags file is not affected by `LC_COLLATE` because it is typically not  
 84997 used by human readers, but only by programs such as *vi* to locate the tag within the source files.  
 84998 Using the POSIX locale eliminates some of the problems of coordinating locales between the  
 84999 *ctags* file creator and the *vi* file reader.

Historically, the tags file has been used only by *ex* and *vi*. However, the format of the tags file has been published to encourage other programs to use the tags in new ways. The format allows either patterns or line numbers to find the identifiers because the historical *vi* recognizes either. The *ctags* utility does not produce the format using line numbers because it is not useful following any source file changes that add or delete lines. The documented search patterns match historical practice. It should be noted that literal leading `<circumflex>` or trailing `<dollar-sign>` characters in the search pattern will only behave correctly if anchored to the beginning of the line or end of the line by an additional `<circumflex>` or `<dollar-sign>` character.

Historical implementations also understand the objects used by the languages Pascal and sometimes LISP, and they understand the C source output by *lex* and *yacc*. The *ctags* utility is not required to accommodate these languages, although implementors are encouraged to do so.

The following historical option was not specified, as *vgrind* is not included in this volume of POSIX.1-2008:

**-v** If the `-v` flag is given, an index of the form expected by *vgrind* is produced on the standard output. This listing contains the function name, filename, and page number (assuming 64-line pages). Since the output is sorted into lexicographic order, it may be desired to run the output through *sort -f*. Sample use:

```
ctags -v files | sort -f > index vgrind -x index
```

The special treatment of the tag **main** makes the use of *ctags* practical in directories with more than one program.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[c99](#), [fort77](#), [vi](#)

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

## CHANGE HISTORY

First released in Issue 4.

### Issue 5

The FUTURE DIRECTIONS section is added.

### Issue 6

This utility is marked as part of the User Portability Utilities option.

The OUTPUT FILES section is changed to align with the IEEE P1003.2b draft standard.

The normative text is reworded to avoid use of the term “must” for application requirements.

IEEE PASC Interpretation 1003.2 #168 is applied, changing “create” to “write” in the DESCRIPTION.

### Issue 7

The *ctags* utility is no longer dependent on support for the User Portability Utilities option.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

**NAME**

cut — cut out selected fields of each line of a file

**SYNOPSIS**

cut -b *list* [-n] [*file...*]

cut -c *list* [*file...*]

cut -f *list* [-d *delim*] [-s] [*file...*]

**DESCRIPTION**

The *cut* utility shall cut out bytes (**-b** option), characters (**-c** option), or character-delimited fields (**-f** option) from each line in one or more files, concatenate them, and write them to standard output.

**OPTIONS**

The *cut* utility shall conform to XBD [Section 12.2](#) (on page 216).

The application shall ensure that the option-argument *list* (see options **-b**, **-c**, and **-f** below) is a <comma>-separated list or <blank>-separated list of positive numbers and ranges. Ranges can be in three forms. The first is two positive numbers separated by a <hyphen-minus> (*low-high*), which represents all fields from the first number to the second number. The second is a positive number preceded by a <hyphen-minus> (*-high*), which represents all fields from field number 1 to that number. The third is a positive number followed by a <hyphen-minus> (*low-*), which represents that number to the last field, inclusive. The elements in *list* can be repeated, can overlap, and can be specified in any order, but the bytes, characters, or fields selected shall be written in the order of the input data. If an element appears in the selection list more than once, it shall be written exactly once.

The following options shall be supported:

**-b list** Cut based on a *list* of bytes. Each selected byte shall be output unless the **-n** option is also specified. It shall not be an error to select bytes not present in the input line.

**-c list** Cut based on a *list* of characters. Each selected character shall be output. It shall not be an error to select characters not present in the input line.

**-d delim** Set the field delimiter to the character *delim*. The default is the <tab>.

**-f list** Cut based on a *list* of fields, assumed to be separated in the file by a delimiter character (see **-d**). Each selected field shall be output. Output fields shall be separated by a single occurrence of the field delimiter character. Lines with no field delimiters shall be passed through intact, unless **-s** is specified. It shall not be an error to select fields not present in the input line.

**-n** Do not split characters. When specified with the **-b** option, each element in *list* of the form *low-high* (<hyphen-minus>-separated numbers) shall be modified as follows:

- If the byte selected by *low* is not the first byte of a character, *low* shall be decremented to select the first byte of the character originally selected by *low*. If the byte selected by *high* is not the last byte of a character, *high* shall be decremented to select the last byte of the character prior to the character originally selected by *high*, or zero if there is no prior character. If the resulting range element has *high* equal to zero or *low* greater than *high*, the list element shall be dropped from *list* for that input line without causing an error.

Each element in *list* of the form *low-* shall be treated as above with *high* set to the

85083 number of bytes in the current line, not including the terminating <newline>. Each  
 85084 element in *list* of the form *-high* shall be treated as above with *low* set to 1. Each  
 85085 element in *list* of the form *num* (a single number) shall be treated as above with *low*  
 85086 set to *num* and *high* set to *num*.

85087 **-s** Suppress lines with no delimiter characters, when used with the **-f** option. Unless  
 85088 specified, lines with no delimiters shall be passed through untouched.

## 85089 OPERANDS

85090 The following operand shall be supported:

85091 *file* A pathname of an input file. If no *file* operands are specified, or if a *file* operand is  
 85092 '*-*', the standard input shall be used.

## 85093 STDIN

85094 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '*-*'.  
 85095 See the INPUT FILES section.

## 85096 INPUT FILES

85097 The input files shall be text files, except that line lengths shall be unlimited.

## 85098 ENVIRONMENT VARIABLES

85099 The following environment variables shall affect the execution of *cut*:

85100 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 85101 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 85102 variables used to determine the values of locale categories.)

85103 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 85104 internationalization variables.

85105 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 85106 characters (for example, single-byte as opposed to multi-byte characters in  
 85107 arguments and input files).

85108 *LC\_MESSAGES*

85109 Determine the locale that should be used to affect the format and contents of  
 85110 diagnostic messages written to standard error.

85111 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 85112 ASYNCHRONOUS EVENTS

85113 Default.

## 85114 STDOUT

85115 The *cut* utility output shall be a concatenation of the selected bytes, characters, or fields (one of  
 85116 the following):

85117 "%s\n", <concatenation of bytes>

85118 "%s\n", <concatenation of characters>

85119 "%s\n", <concatenation of fields and field delimiters>

## 85120 STDERR

85121 The standard error shall be used only for diagnostic messages.

85122 **OUTPUT FILES**

85123 None.

85124 **EXTENDED DESCRIPTION**

85125 None.

85126 **EXIT STATUS**

85127 The following exit values shall be returned:

85128 0 All input files were output successfully.

85129 &gt;0 An error occurred.

85130 **CONSEQUENCES OF ERRORS**

85131 Default.

85132 **APPLICATION USAGE**

85133 The *cut* and *fold* utilities can be used to create text files out of files with arbitrary line lengths.  
 85134 The *cut* utility should be used when the number of lines (or records) needs to remain constant.  
 85135 The *fold* utility should be used when the contents of long lines need to be kept contiguous.

85136 Earlier versions of the *cut* utility worked in an environment where bytes and characters were  
 85137 considered equivalent (modulo <backspace> and <tab> processing in some implementations). In  
 85138 the extended world of multi-byte characters, the new **-b** option has been added. The **-n** option  
 85139 (used with **-b**) allows it to be used to act on bytes rounded to character boundaries. The  
 85140 algorithm specified for **-n** guarantees that:

85141 `cut -b 1-500 -n file > file1`85142 `cut -b 501- -n file > file2`

85143 ends up with all the characters in **file** appearing exactly once in **file1** or **file2**. (There is,  
 85144 however, a <newline> in both **file1** and **file2** for each <newline> in **file**.)

85145 **EXAMPLES**

85146 Examples of the option qualifier list:

85147 1,4,7 Select the first, fourth, and seventh bytes, characters, or fields and field delimiters.

85148 1-3,8 Equivalent to 1,2,3,8.

85149 -5,10 Equivalent to 1,2,3,4,5,10.

85150 3- Equivalent to third to last, inclusive.

85151 The *low-high* forms are not always equivalent when used with **-b** and **-n** and multi-byte  
 85152 characters; see the description of **-n**.

85153 The following command:

85154 `cut -d : -f 1,6 /etc/passwd`

85155 reads the System V password file (user database) and produces lines of the form:

85156 `<user ID>:<home directory>`

85157 Most utilities in this volume of POSIX.1-2008 work on text files. The *cut* utility can be used to  
 85158 turn files with arbitrary line lengths into a set of text files containing the same data. The *paste*  
 85159 utility can be used to create (or recreate) files with arbitrary line lengths. For example, if **file**  
 85160 contains long lines:

85161 `cut -b 1-500 -n file > file1`85162 `cut -b 501- -n file > file2`

creates **file1** (a text file) with lines no longer than 500 bytes (plus the <newline>) and **file2** that contains the remainder of the data from **file**. (Note that **file2** is not a text file if there are lines in **file** that are longer than 500 + {LINE\_MAX} bytes.) The original file can be recreated from **file1** and **file2** using the command:

```
paste -d "\0" file1 file2 > file
```

## RATIONALE

Some historical implementations do not count <backspace> characters in determining character counts with the **-c** option. This may be useful for using *cut* for processing *nroff* output. It was deliberately decided not to have the **-c** option treat either <backspace> or <tab> characters in any special fashion. The *fold* utility does treat these characters specially.

Unlike other utilities, some historical implementations of *cut* exit after not finding an input file, rather than continuing to process the remaining *file* operands. This behavior is prohibited by this volume of POSIX.1-2008, where only the exit status is affected by this problem.

The behavior of *cut* when provided with either mutually-exclusive options or options that do not work logically together has been deliberately left unspecified in favor of global wording in [Section 1.4](#) (on page 2326).

The OPTIONS section was changed in response to IEEE PASC Interpretation 1003.2 #149. The change represents historical practice on all known systems. The original standard was ambiguous on the nature of the output.

The *list* option-arguments are historically used to select the portions of the line to be written, but do not affect the order of the data. For example:

```
echo abcdefghi | cut -c6,2,4-7,1
```

yields "abdefg".

A proposal to enhance *cut* with the following option:

**-o** Preserve the selected field order. When this option is specified, each byte, character, or field (or ranges of such) shall be written in the order specified by the *list* option-argument, even if this requires multiple outputs of the same bytes, characters, or fields.

was rejected because this type of enhancement is outside the scope of the IEEE P1003.2b draft standard.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[Section 2.5](#) (on page 2339), *fold*, *grep*, *paste*

[XBD Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

## CHANGE HISTORY

First released in Issue 2.

## Issue 6

The OPTIONS section is changed to align with the IEEE P1003.2b draft standard.

The normative text is reworded to avoid use of the term “must” for application requirements.

85202 **Issue 7**

85203 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

85204 SD5-XCU-ERN-171 is applied, adding APPLICATION USAGE.



85205 **NAME**85206 cxref — generate a C-language program cross-reference table (**DEVELOPMENT**)85207 **SYNOPSIS**

```
85208 XSI  cxref [-cs] [-o file] [-w num] [-D name[=def]]... [-I dir]...
85209      [-U name]... file...
```

85210 **DESCRIPTION**

85211 The *cxref* utility shall analyze a collection of C-language *files* and attempt to build a cross-  
 85212 reference table. Information from **#define** lines shall be included in the symbol table. A sorted  
 85213 listing shall be written to standard output of all symbols (auto, static, and global) in each *file*  
 85214 separately, or with the **-c** option, in combination. Each symbol shall contain an <asterisk> before  
 85215 the declaring reference.

85216 **OPTIONS**

85217 The *cxref* utility shall conform to XBD [Section 12.2](#) (on page 216), except that the order of the **-D**,  
 85218 **-I**, and **-U** options (which are identical to their interpretation by *c99*) is significant. The  
 85219 following options shall be supported:

- 85220 **-c** Write a combined cross-reference of all input files.
- 85221 **-s** Operate silently; do not print input filenames.
- 85222 **-o file** Direct output to named *file*.
- 85223 **-w num** Format output no wider than *num* (decimal) columns. This option defaults to 80 if  
 85224 *num* is not specified or is less than 51.
- 85225 **-D** Equivalent to *c99*.
- 85226 **-I** Equivalent to *c99*.
- 85227 **-U** Equivalent to *c99*.

85228 **OPERANDS**

85229 The following operand shall be supported:

- 85230 *file* A pathname of a C-language source file.

85231 **STDIN**

85232 Not used.

85233 **INPUT FILES**

85234 The input files are C-language source files.

85235 **ENVIRONMENT VARIABLES**

85236 The following environment variables shall affect the execution of *cxref*:

- 85237 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 85238 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 85239 variables used to determine the values of locale categories.)
- 85240 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 85241 internationalization variables.
- 85242 **LC\_COLLATE**  
 85243 Determine the locale for the ordering of the output.
- 85244 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 85245 characters (for example, single-byte as opposed to multi-byte characters in  
 85246 arguments and input files).



**LC\_MESSAGES**

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

**NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

**ASYNCHRONOUS EVENTS**

Default.

**STDOUT**

The standard output shall be used for the cross-reference listing, unless the **-o** option is used to select a different output file.

The format of standard output is unspecified, except that the following information shall be included:

- If the **-c** option is not specified, each portion of the listing shall start with the name of the input file on a separate line.
- The name line shall be followed by a sorted list of symbols, each with its associated location pathname, the name of the function in which it appears (if it is not a function name itself), and line number references.
- Each line number may be preceded by an <asterisk> ('\*') flag, meaning that this is the declaring reference. Other single-character flags, with implementation-defined meanings, may be included.

**STDERR**

The standard error shall be used only for diagnostic messages.

**OUTPUT FILES**

The output file named by the **-o** option shall be used instead of standard output.

**EXTENDED DESCRIPTION**

None.

**EXIT STATUS**

The following exit values shall be returned:

- 0 Successful completion.
- >0 An error occurred.

**CONSEQUENCES OF ERRORS**

Default.

**APPLICATION USAGE**

None.

**EXAMPLES**

None.

**RATIONALE**

None.

**FUTURE DIRECTIONS**

None.

85286 **SEE ALSO**85287 [c99](#)85288 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)85289 **CHANGE HISTORY**

85290 First released in Issue 2.

85291 **Issue 5**85292 In the SYNOPSIS, `[-U dir]` is changed to `[-U name]`.85293 **Issue 6**

85294 The APPLICATION USAGE section is added.

85295 **Issue 7**

85296 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

85297 **NAME**

85298           date — write the date and time

85299 **SYNOPSIS**

85300           date [-u] [+format]

85301 XSI       date [-u] mmddhhmm[[cc]yy]

85302 **DESCRIPTION**

85303 XSI       The *date* utility shall write the date and time to standard output or attempt to set the system  
 85304       date and time. By default, the current date and time shall be written. If an operand beginning  
 85305       with '+' is specified, the output format of *date* shall be controlled by the conversion  
 85306       specifications and other text in the operand.

85307 **OPTIONS**85308       The *date* utility shall conform to XBD Section 12.2 (on page 216).

85309       The following option shall be supported:

85310       -u       Perform operations as if the *TZ* environment variable was set to the string "UTC0",  
 85311       or its equivalent historical value of "GMT0". Otherwise, *date* shall use the timezone  
 85312       indicated by the *TZ* environment variable or the system default if that variable is  
 85313       unset or null.

85314 **OPERANDS**

85315       The following operands shall be supported:

85316       +format   When the format is specified, each conversion specifier shall be replaced in the  
 85317       standard output by its corresponding value. All other characters shall be copied to  
 85318       the output without change. The output shall always be terminated with a  
 85319       <newline>.

85320 **Conversion Specifications**

85321       %a       Locale's abbreviated weekday name.  
 85322       %A       Locale's full weekday name.  
 85323       %b       Locale's abbreviated month name.  
 85324       %B       Locale's full month name.  
 85325       %c       Locale's appropriate date and time representation.  
 85326       %C       Century (a year divided by 100 and truncated to an integer) as a decimal  
 85327       number [00,99].  
 85328       %d       Day of the month as a decimal number [01,31].  
 85329       %D       Date in the format *mm/dd/yy*.  
 85330       %e       Day of the month as a decimal number [1,31] in a two-digit field with  
 85331       leading <space> character fill.  
 85332       %h       A synonym for %b.  
 85333       %H       Hour (24-hour clock) as a decimal number [00,23].  
 85334       %I       Hour (12-hour clock) as a decimal number [01,12].

|       |    |                                                                              |
|-------|----|------------------------------------------------------------------------------|
| 85335 | %j | Day of the year as a decimal number [001,366].                               |
| 85336 | %m | Month as a decimal number [01,12].                                           |
| 85337 | %M | Minute as a decimal number [00,59].                                          |
| 85338 | %n | A <newline>.                                                                 |
| 85339 | %p | Locale's equivalent of either AM or PM.                                      |
| 85340 | %r | 12-hour clock time [01,12] using the AM/PM notation; in the POSIX            |
| 85341 |    | locale, this shall be equivalent to %I:%M:%S %p.                             |
| 85342 | %S | Seconds as a decimal number [00,60].                                         |
| 85343 | %t | A <tab>.                                                                     |
| 85344 | %T | 24-hour clock time [00,23] in the format <i>HH:MM:SS</i> .                   |
| 85345 | %u | Weekday as a decimal number [1,7] (1=Monday).                                |
| 85346 | %U | Week of the year (Sunday as the first day of the week) as a decimal          |
| 85347 |    | number [00,53]. All days in a new year preceding the first Sunday shall be   |
| 85348 |    | considered to be in week 0.                                                  |
| 85349 | %V | Week of the year (Monday as the first day of the week) as a decimal          |
| 85350 |    | number [01,53]. If the week containing January 1 has four or more days in    |
| 85351 |    | the new year, then it shall be considered week 1; otherwise, it shall be the |
| 85352 |    | last week of the previous year, and the next week shall be week 1.           |
| 85353 | %w | Weekday as a decimal number [0,6] (0=Sunday).                                |
| 85354 | %W | Week of the year (Monday as the first day of the week) as a decimal          |
| 85355 |    | number [00,53]. All days in a new year preceding the first Monday shall      |
| 85356 |    | be considered to be in week 0.                                               |
| 85357 | %x | Locale's appropriate date representation.                                    |
| 85358 | %X | Locale's appropriate time representation.                                    |
| 85359 | %y | Year within century [00,99].                                                 |
| 85360 | %Y | Year with century as a decimal number.                                       |
| 85361 | %Z | Timezone name, or no characters if no timezone is determinable.              |
| 85362 | %% | A <percent-sign> character.                                                  |

See XBD [Section 7.3.5](#) (on page 159) for the conversion specifier values in the POSIX locale.

### Modified Conversion Specifications

Some conversion specifiers can be modified by the E and O modifier characters to indicate a different format or specification as specified in the *LC\_TIME* locale description (see XBD [Section 7.3.5](#), on page 159). If the corresponding keyword (see *era*, *era\_year*, *era\_d\_fmt*, and *alt\_digits* in XBD [Section 7.3.5](#), on page 159) is not specified or not supported for the current locale, the unmodified conversion specifier value shall be used.

|       |     |                                                                |
|-------|-----|----------------------------------------------------------------|
| 85372 | %Ec | Locale's alternative appropriate date and time representation. |
|-------|-----|----------------------------------------------------------------|

|       |              |                                                                                                                                           |
|-------|--------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| 85373 | %EC          | The name of the base year (period) in the locale's alternative representation.                                                            |
| 85374 |              |                                                                                                                                           |
| 85375 | %Ex          | Locale's alternative date representation.                                                                                                 |
| 85376 | %EX          | Locale's alternative time representation.                                                                                                 |
| 85377 | %Ey          | Offset from %EC (year only) in the locale's alternative representation.                                                                   |
| 85378 | %EY          | Full alternative year representation.                                                                                                     |
| 85379 | %Od          | Day of month using the locale's alternative numeric symbols.                                                                              |
| 85380 | %Oe          | Day of month using the locale's alternative numeric symbols.                                                                              |
| 85381 | %OH          | Hour (24-hour clock) using the locale's alternative numeric symbols.                                                                      |
| 85382 | %OI          | Hour (12-hour clock) using the locale's alternative numeric symbols.                                                                      |
| 85383 | %Om          | Month using the locale's alternative numeric symbols.                                                                                     |
| 85384 | %OM          | Minutes using the locale's alternative numeric symbols.                                                                                   |
| 85385 | %OS          | Seconds using the locale's alternative numeric symbols.                                                                                   |
| 85386 | %Ou          | Weekday as a number in the locale's alternative representation (Monday = 1).                                                              |
| 85387 |              |                                                                                                                                           |
| 85388 | %OU          | Week number of the year (Sunday as the first day of the week) using the locale's alternative numeric symbols.                             |
| 85389 |              |                                                                                                                                           |
| 85390 | %OV          | Week number of the year (Monday as the first day of the week, rules corresponding to %V), using the locale's alternative numeric symbols. |
| 85391 |              |                                                                                                                                           |
| 85392 | %Ow          | Weekday as a number in the locale's alternative representation (Sunday = 0).                                                              |
| 85393 |              |                                                                                                                                           |
| 85394 | %OW          | Week number of the year (Monday as the first day of the week) using the locale's alternative numeric symbols.                             |
| 85395 |              |                                                                                                                                           |
| 85396 | %Oy          | Year (offset from %C) in alternative representation.                                                                                      |
| 85397 | XSI          | <b><code>mmdhmm[[cc]yy]</code></b>                                                                                                        |
| 85398 |              | Attempt to set the system date and time from the value given in the operand. This                                                         |
| 85399 |              | is only possible if the user has appropriate privileges and the system permits the                                                        |
| 85400 |              | setting of the system date and time. The first <i>mm</i> is the month (number); <i>dd</i> is the                                          |
| 85401 |              | day (number); <i>hh</i> is the hour (number, 24-hour system); the second <i>mm</i> is the                                                 |
| 85402 |              | minute (number); <i>cc</i> is the century and is the first two digits of the year (this is                                                |
| 85403 |              | optional); <i>yy</i> is the last two digits of the year and is optional. If century is not                                                |
| 85404 |              | specified, then values in the range [69,99] shall refer to years 1969 to 1999 inclusive,                                                  |
| 85405 |              | and values in the range [00,68] shall refer to years 2000 to 2068 inclusive. The                                                          |
| 85406 |              | current year is the default if <i>yy</i> is omitted.                                                                                      |
| 85407 | <b>Note:</b> | It is expected that in a future version of this standard the default century inferred                                                     |
| 85408 |              | from a 2-digit year will change. (This would apply to all commands accepting a                                                            |
| 85409 |              | 2-digit year as input.)                                                                                                                   |
| 85410 | <b>STDIN</b> |                                                                                                                                           |
| 85411 |              | Not used.                                                                                                                                 |

85412 **INPUT FILES**

85413 None.

85414 **ENVIRONMENT VARIABLES**85415 The following environment variables shall affect the execution of *date*:

85416 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 85417 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 85418 variables used to determine the values of locale categories.)

85419 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 85420 internationalization variables.

85421 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 85422 characters (for example, single-byte as opposed to multi-byte characters in  
 85423 arguments).

85424 *LC\_MESSAGES*

85425 Determine the locale that should be used to affect the format and contents of  
 85426 diagnostic messages written to standard error.

85427 *LC\_TIME* Determine the format and contents of date and time strings written by *date*.

85428 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

85429 *TZ* Determine the timezone in which the time and date are written, unless the *-u*  
 85430 option is specified. If the *TZ* variable is unset or null and *-u* is not specified, an  
 85431 unspecified system default timezone is used.

85432 **ASYNCHRONOUS EVENTS**

85433 Default.

85434 **STDOUT**

85435 When no formatting operand is specified, the output in the POSIX locale shall be equivalent to  
 85436 specifying:

85437 *date* "+%a %b %e %H:%M:%S %Z %Y"85438 **STDERR**

85439 The standard error shall be used only for diagnostic messages.

85440 **OUTPUT FILES**

85441 None.

85442 **EXTENDED DESCRIPTION**

85443 None.

85444 **EXIT STATUS**

85445 The following exit values shall be returned:

85446 0 The date was written successfully.

85447 &gt;0 An error occurred.

85448 **CONSEQUENCES OF ERRORS**

85449 Default.

## APPLICATION USAGE

Conversion specifiers are of unspecified format when not in the POSIX locale. Some of them can contain <newline> characters in some locales, so it may be difficult to use the format shown in standard output for parsing the output of *date* in those locales.

The range of values for %S extends from 0 to 60 seconds to accommodate the occasional leap second.

Although certain of the conversion specifiers in the POSIX locale (such as the name of the month) are shown with initial capital letters, this need not be the case in other locales. Programs using these fields may need to adjust the capitalization if the output is going to be used at the beginning of a sentence.

The date string formatting capabilities are intended for use in Gregorian-style calendars, possibly with a different starting year (or years). The %x and %c conversion specifications, however, are intended for local representation; these may be based on a different, non-Gregorian calendar.

The %C conversion specification was introduced to allow a fallback for the %EC (alternative year format base year); it can be viewed as the base of the current subdivision in the Gregorian calendar. The century number is calculated as the year divided by 100 and truncated to an integer; it should not be confused with the use of ordinal numbers for centuries (for example, "twenty-first century".) Both the %Ey and %y can then be viewed as the offset from %EC and %C, respectively.

The E and O modifiers modify the traditional conversion specifiers, so that they can always be used, even if the implementation (or the current locale) does not support the modifier.

The E modifier supports alternative date formats, such as the Japanese Emperor's Era, as long as these are based on the Gregorian calendar system. Extending the E modifiers to other date elements may provide an implementation-defined extension capable of supporting other calendar systems, especially in combination with the O modifier.

The O modifier supports time and date formats using the locale's alternative numerical symbols, such as Kanji or Hindi digits or ordinal number representation.

Non-European locales, whether they use Latin digits in computational items or not, often have local forms of the digits for use in date formats. This is not totally unknown even in Europe; a variant of dates uses Roman numerals for the months: the third day of September 1991 would be written as 3.IX.1991. In Japan, Kanji digits are regularly used for dates; in Arabic-speaking countries, Hindi digits are used. The %d, %e, %H, %I, %m, %S, %U, %w, %W, and %Y conversion specifications always return the date and time field in Latin digits (that is, 0 to 9). The %O modifier was introduced to support the use for display purposes of non-Latin digits. In the *LC\_TIME* category in *localedef*, the optional **alt\_digits** keyword is intended for this purpose. As an example, assume the following (partial) *localedef* source:

```
alt_digits  " "; "I"; "II"; "III"; "IV"; "V"; "VI"; "VII"; "VIII" \
            "IX"; "X"; "XI"; "XII"
d_fmt      "%e.%Om.%Y"
```

With the above date, the command:

```
date "+%x"
```

would yield 3.IX.1991. With the same **d\_fmt**, but without the **alt\_digits**, the command would yield 3.9.1991.

## 85494 EXAMPLES

- 85495 1. The following are input/output examples of *date* used at arbitrary times in the POSIX  
85496 locale:

85497 \$ date  
85498 **Tue Jun 26 09:58:10 PDT 1990**

85499 \$ date "+DATE: %m/%d/%y%nTIME: %H:%M:%S"  
85500 **DATE: 11/02/91**  
85501 **TIME: 13:36:16**

85502 \$ date "+TIME: %r"  
85503 **TIME: 01:36:32 PM**

- 85504 2. Examples for Denmark, where the default date and time format is %a %d %b %Y %T %Z:

85505 \$ LANG=da\_DK.iso\_8859-1 date  
85506 **ons 02 okt 1991 15:03:32 CET**

85507 \$ LANG=da\_DK.iso\_8859-1 \  
85508 date "+DATO: %A den %e. %B %Y%nKLOKKEN: %H:%M:%S"  
85509 **DATO: onsdag den 2. oktober 1991**  
85510 **KLOKKEN: 15:03:56**

- 85511 3. Examples for Germany, where the default date and time format is %a %d.%h.%Y, %T %Z:

85512 \$ LANG=De\_DE.88591 date  
85513 **Mi 02.Okt.1991, 15:01:21 MEZ**

85514 \$ LANG=De\_DE.88591 date "+DATUM: %A, %d. %B %Y%nZEIT: %H:%M:%S"  
85515 **DATUM: Mittwoch, 02. Oktober 1991**  
85516 **ZEIT: 15:02:02**

- 85517 4. Examples for France, where the default date and time format is %a %d %h %Y %Z %T:

85518 \$ LANG=Fr\_FR.88591 date  
85519 **Mer 02 oct 1991 MET 15:03:32**

85520 \$ LANG=Fr\_FR.88591 date "+JOUR: %A %d %B %Y%nHEURE: %H:%M:%S"  
85521 **JOUR: Mercredi 02 octobre 1991**  
85522 **HEURE: 15:03:56**

## 85523 RATIONALE

85524 Some of the new options for formatting are from the ISO C standard. The **-u** option was  
85525 introduced to allow portable access to Coordinated Universal Time (UTC). The string "GMT0" is  
85526 allowed as an equivalent TZ value to be compatible with all of the systems using the BSD  
85527 implementation, where this option originated.

85528 The %e format conversion specification (adopted from System V) was added because the ISO C  
85529 standard conversion specifications did not provide any way to produce the historical default  
85530 *date* output during the first nine days of any month.

85531 There are two varieties of day and week numbering supported (in addition to any others created  
85532 with the locale-dependent %E and %O modifier characters):

- 85533 • The historical variety in which Sunday is the first day of the week and the weekdays
- 85534 preceding the first Sunday of the year are considered week 0. These are represented by %w
- 85535 and %U. A variant of this is %W, using Monday as the first day of the week, but still
- 85536 referring to week 0. This view of the calendar was retained because so many historical



85537 applications depend on it and the ISO C standard *strftime()* function, on which many *date*  
85538 implementations are based, was defined in this way.

- 85539 • The international standard, based on the ISO 8601: 2004 standard where Monday is the first  
85540 weekday and the algorithm for the first week number is more complex: If the week  
85541 (Monday to Sunday) containing January 1 has four or more days in the new year, then it is  
85542 week 1; otherwise, it is week 53 of the previous year, and the next week is week 1. These  
85543 are represented by the new conversion specifications %u and %V, added as a result of  
85544 international comments.

#### 85545 FUTURE DIRECTIONS

85546 None.

#### 85547 SEE ALSO

85548 XBD [Section 7.3.5](#) (on page 159), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

85549 XSH [fprintf\(\)](#), [strftime\(\)](#)

#### 85550 CHANGE HISTORY

85551 First released in Issue 2.

#### 85552 Issue 5

85553 Changes are made for Year 2000 alignment.

#### 85554 Issue 6

85555 The following new requirements on POSIX implementations derive from alignment with the  
85556 Single UNIX Specification:

- 85557 • The %EX modified conversion specification is added.

85558 The Open Group Corrigendum U048/2 is applied, correcting the examples.

85559 The DESCRIPTION is updated to refer to conversion specifications, instead of field descriptors  
85560 for consistency with the *LC\_TIME* category.

85561 A clarification is made such that the current year is the default if the *yy* argument is omitted  
85562 when setting the system date and time.

85563 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/19 is applied, correcting the CHANGE  
85564 HISTORY section.

## 85565 NAME

85566 dd — convert and copy a file

## 85567 SYNOPSIS

85568 dd [*operand*...]

## 85569 DESCRIPTION

85570 The *dd* utility shall copy the specified input file to the specified output file with possible  
 85571 conversions using specific input and output block sizes. It shall read the input one block at a  
 85572 time, using the specified input block size; it shall then process the block of data actually  
 85573 returned, which could be smaller than the requested block size. It shall apply any conversions  
 85574 that have been specified and write the resulting data to the output in blocks of the specified  
 85575 output block size. If the **bs=expr** operand is specified and no conversions other than **sync**,  
 85576 **noerror**, or **notrunc** are requested, the data returned from each input block shall be written as a  
 85577 separate output block; if the read returns less than a full block and the **sync** conversion is not  
 85578 specified, the resulting output block shall be the same size as the input block. If the **bs=expr**  
 85579 operand is not specified, or a conversion other than **sync**, **noerror**, or **notrunc** is requested, the  
 85580 input shall be processed and collected into full-sized output blocks until the end of the input is  
 85581 reached.

85582 The processing order shall be as follows:

- 85583 1. An input block is read.
- 85584 2. If the input block is shorter than the specified input block size and the **sync** conversion is  
 85585 specified, null bytes shall be appended to the input data up to the specified size. (If either  
 85586 **block** or **unblock** is also specified, <space> characters shall be appended instead of null  
 85587 bytes.) The remaining conversions and output shall include the pad characters as if they  
 85588 had been read from the input.
- 85589 3. If the **bs=expr** operand is specified and no conversion other than **sync** or **noerror** is  
 85590 requested, the resulting data shall be written to the output as a single block, and the  
 85591 remaining steps are omitted.
- 85592 4. If the **swab** conversion is specified, each pair of input data bytes shall be swapped. If  
 85593 there is an odd number of bytes in the input block, the last byte in the input record shall  
 85594 not be swapped.
- 85595 5. Any remaining conversions (**block**, **unblock**, **lcase**, and **ucase**) shall be performed. These  
 85596 conversions shall operate on the input data independently of the input blocking; an input  
 85597 or output fixed-length record may span block boundaries.
- 85598 6. The data resulting from input or conversion or both shall be aggregated into output  
 85599 blocks of the specified size. After the end of input is reached, any remaining output shall  
 85600 be written as a block without padding if **conv=sync** is not specified; thus, the final output  
 85601 block may be shorter than the output block size.

## 85602 OPTIONS

85603 None.

## 85604 OPERANDS

85605 All of the operands shall be processed before any input is read. The following operands shall be  
 85606 supported:

- 85607 **if=file** Specify the input pathname; the default is standard input.
- 85608 **of=file** Specify the output pathname; the default is standard output. If the **seek=expr**  
 85609 conversion is not also specified, the output file shall be truncated before the copy  
 85610 begins if an explicit **of=file** operand is specified, unless **conv=notrunc** is specified.

|       |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 85611 |                               | If <b>seek=expr</b> is specified, but <b>conv=notrunc</b> is not, the effect of the copy shall be to preserve the blocks in the output file over which <i>dd</i> seeks, but no other portion of the output file shall be preserved. (If the size of the seek plus the size of the input file is less than the previous size of the output file, the output file shall be shortened by the copy. If the input file is empty and either the size of the seek is greater than the previous size of the output file or the output file did not previously exist, the size of the output file shall be set to the file offset after the seek.)                          |
| 85612 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85613 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85614 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85615 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85616 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85617 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85618 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85619 | <b>ibs=expr</b>               | Specify the input block size, in bytes, by <i>expr</i> (default is 512).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 85620 | <b>obs=expr</b>               | Specify the output block size, in bytes, by <i>expr</i> (default is 512).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 85621 | <b>bs=expr</b>                | Set both input and output block sizes to <i>expr</i> bytes, superseding <b>ibs=</b> and <b>obs=</b> . If no conversion other than <b>sync</b> , <b>noerror</b> , and <b>notrunc</b> is specified, each input block shall be copied to the output as a single block without aggregating short blocks.                                                                                                                                                                                                                                                                                                                                                               |
| 85622 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85623 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85624 | <b>cbs=expr</b>               | Specify the conversion block size for <b>block</b> and <b>unblock</b> in bytes by <i>expr</i> (default is zero). If <b>cbs=</b> is omitted or given a value of zero, using <b>block</b> or <b>unblock</b> produces unspecified results.                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 85625 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85626 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85627 | XSI                           | The application shall ensure that this operand is also specified if the <b>conv=</b> operand is specified with a value of <b>ascii</b> , <b>ebcdic</b> , or <b>ibm</b> . For a <b>conv=</b> operand with an <b>ascii</b> value, the input is handled as described for the <b>unblock</b> value, except that characters are converted to ASCII before any trailing <space> characters are deleted. For <b>conv=</b> operands with <b>ebcdic</b> or <b>ibm</b> values, the input is handled as described for the <b>block</b> value except that the characters are converted to EBCDIC or IBM EBCDIC, respectively, after any trailing <space> characters are added. |
| 85628 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85629 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85630 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85631 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85632 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85633 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85634 | <b>skip=n</b>                 | Skip <i>n</i> input blocks (using the specified input block size) before starting to copy. On seekable files, the implementation shall read the blocks or seek past them; on non-seekable files, the blocks shall be read and the data shall be discarded.                                                                                                                                                                                                                                                                                                                                                                                                         |
| 85635 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85636 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85637 | <b>seek=n</b>                 | Skip <i>n</i> blocks (using the specified output block size) from the beginning of the output file before copying. On non-seekable files, existing blocks shall be read and space from the current end-of-file to the specified offset, if any, filled with null bytes; on seekable files, the implementation shall seek to the specified offset or read the blocks as described for non-seekable files.                                                                                                                                                                                                                                                           |
| 85638 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85639 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85640 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85641 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85642 | <b>count=n</b>                | Copy only <i>n</i> input blocks. If <i>n</i> is zero, it is unspecified whether no blocks or all blocks are copied.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 85643 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85644 | <b>conv=value[,value ...]</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85645 |                               | Where <i>values</i> are <comma>-separated symbols from the following list:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 85646 | XSI                           | <b>ascii</b> Convert EBCDIC to ASCII; see <a href="#">Table 4-7</a> (on page 2633).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 85647 | XSI                           | <b>ebcdic</b> Convert ASCII to EBCDIC; see <a href="#">Table 4-7</a> (on page 2633).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 85648 | XSI                           | <b>ibm</b> Convert ASCII to a different EBCDIC set; see <a href="#">Table 4-8</a> (on page 2634).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 85649 | XSI                           | The <b>ascii</b> , <b>ebcdic</b> , and <b>ibm</b> values are mutually-exclusive.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 85650 | <b>block</b>                  | Treat the input as a sequence of <newline>-terminated or end-of-file-terminated variable-length records independent of the input block boundaries. Each record shall be converted to a record with a fixed length specified by the conversion block size. Any <newline> shall be removed from the input line; <space> characters shall be appended to lines that are shorter than their conversion block size to fill the block.                                                                                                                                                                                                                                   |
| 85651 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85652 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85653 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85654 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 85655 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

|       |                |                                                                                    |
|-------|----------------|------------------------------------------------------------------------------------|
| 85656 |                | Lines that are longer than the conversion block size shall be truncated            |
| 85657 |                | to the largest number of characters that fit into that size; the number of         |
| 85658 |                | truncated lines shall be reported (see the STDERR section).                        |
| 85659 |                | The <b>block</b> and <b>unblock</b> values are mutually-exclusive.                 |
| 85660 | <b>unblock</b> | Convert fixed-length records to variable length. Read a number of bytes            |
| 85661 |                | equal to the conversion block size (or the number of bytes remaining in            |
| 85662 |                | the input, if less than the conversion block size), delete all trailing            |
| 85663 |                | <space> characters, and append a <newline>.                                        |
| 85664 | <b>lcase</b>   | Map uppercase characters specified by the <i>LC_CTYPE</i> keyword                  |
| 85665 |                | <b>tolower</b> to the corresponding lowercase character. Characters for which      |
| 85666 |                | no mapping is specified shall not be modified by this conversion.                  |
| 85667 |                | The <b>lcase</b> and <b>ucase</b> symbols are mutually-exclusive.                  |
| 85668 | <b>ucase</b>   | Map lowercase characters specified by the <i>LC_CTYPE</i> keyword                  |
| 85669 |                | <b>toupper</b> to the corresponding uppercase character. Characters for            |
| 85670 |                | which no mapping is specified shall not be modified by this conversion.            |
| 85671 | <b>swab</b>    | Swap every pair of input bytes.                                                    |
| 85672 | <b>noerror</b> | Do not stop processing on an input error. When an input error occurs, a            |
| 85673 |                | diagnostic message shall be written on standard error, followed by the             |
| 85674 |                | current input and output block counts in the same format as used at                |
| 85675 |                | completion (see the STDERR section). If the <b>sync</b> conversion is              |
| 85676 |                | specified, the missing input shall be replaced with null bytes and                 |
| 85677 |                | processed normally; otherwise, the input block shall be omitted from               |
| 85678 |                | the output.                                                                        |
| 85679 | <b>notrunc</b> | Do not truncate the output file. Preserve blocks in the output file not            |
| 85680 |                | explicitly written by this invocation of the <i>dd</i> utility. (See also the      |
| 85681 |                | preceding <b>of=file</b> operand.)                                                 |
| 85682 | <b>sync</b>    | Pad every input block to the size of the <b>ibs=</b> buffer, appending null        |
| 85683 |                | bytes. (If either <b>block</b> or <b>unblock</b> is also specified, append <space> |
| 85684 |                | characters, rather than null bytes.)                                               |

The behavior is unspecified if operands other than **conv=** are specified more than once.

For the **bs=**, **cbs=**, **ibs=**, and **obs=** operands, the application shall supply an expression specifying a size in bytes. The expression, *expr*, can be:

1. A positive decimal number
2. A positive decimal number followed by *k*, specifying multiplication by 1 024
3. A positive decimal number followed by *b*, specifying multiplication by 512
4. Two or more positive decimal numbers (with or without *k* or *b*) separated by *x*, specifying the product of the indicated values

All of the operands are processed before any input is read.

The following two tables display the octal number character values used for the **ascii** and **ebcdic** conversions (first table) and for the **ibm** conversion (second table). In both tables, the ASCII values are the row and column headers and the EBCDIC values are found at their intersections. For example, ASCII 0012 (LF) is the second row, third column, yielding 0045 in EBCDIC. The inverted tables (for EBCDIC to ASCII conversion) are not shown, but are in one-to-one

85699

85700

85701

correspondence with these tables. The differences between the two tables are highlighted by small boxes drawn around five entries.

Table 4-7 ASCII to EBCDIC Conversion

|             | 0        | 1        | 2        | 3        | 4        | 5        | 6        | 7        |
|-------------|----------|----------|----------|----------|----------|----------|----------|----------|
| <b>0000</b> | 0000 NUL | 0001 SOH | 0002 STX | 0003 ETX | 0067 EOT | 0055 ENQ | 0056 ACK | 0057 BEL |
| <b>0010</b> | 0026 BS  | 0005 HT  | 0045 LF  | 0013 VT  | 0014 FF  | 0015 CR  | 0016 SO  | 0017 SI  |
| <b>0020</b> | 0020 DLE | 0021 DC1 | 0022 DC2 | 0023 DC3 | 0074 DC4 | 0075 NAK | 0062 SYN | 0046 ETB |
| <b>0030</b> | 0030 CAN | 0031 EM  | 0077 SUB | 0047 ESC | 0034 IFS | 0035 IGS | 0036 IRS | 0037 ITB |
| <b>0040</b> | 0100 Sp  | 0132 !   | 0177 "   | 0173 #   | 0133 \$  | 0154 %   | 0120 &   | 0175 '   |
| <b>0050</b> | 0115 (   | 0135 )   | 0134 *   | 0116 +   | 0153 ,   | 0140 -   | 0113 .   | 0141 /   |
| <b>0060</b> | 0360 0   | 0361 1   | 0362 2   | 0363 3   | 0364 4   | 0365 5   | 0366 6   | 0367 7   |
| <b>0070</b> | 0370 8   | 0371 9   | 0172 :   | 0136 ;   | 0114 <   | 0176 =   | 0156 >   | 0157 ?   |
| <b>0100</b> | 0174 @   | 0301 A   | 0302 B   | 0303 C   | 0304 D   | 0305 E   | 0306 F   | 0307 G   |
| <b>0110</b> | 0310 H   | 0311 I   | 0321 J   | 0322 K   | 0323 L   | 0324 M   | 0325 N   | 0326 O   |
| <b>0120</b> | 0327 P   | 0330 Q   | 0331 R   | 0342 S   | 0343 T   | 0344 U   | 0345 V   | 0346 W   |
| <b>0130</b> | 0347 X   | 0350 Y   | 0351 Z   | 0255 [   | 0340 \   | 0275 ]   | 0232     | 0155 _   |
| <b>0140</b> | 0171 `   | 0201 a   | 0202 b   | 0203 c   | 0204 d   | 0205 e   | 0206 f   | 0207 g   |
| <b>0150</b> | 0210 h   | 0211 i   | 0221 j   | 0222 k   | 0223 l   | 0224 m   | 0225 n   | 0226 o   |
| <b>0160</b> | 0227 p   | 0230 q   | 0231 r   | 0242 s   | 0243 t   | 0244 u   | 0245 v   | 0246 w   |
| <b>0170</b> | 0247 x   | 0250 y   | 0251 z   | 0300 {   | 0117     | 0320 }   | 0137 ~   | 0007 DEL |
| <b>0200</b> | 0040 DS  | 0041 SOS | 0042 FS  | 0043 WUS | 0044 BYP | 0025 NL  | 0006 RNL | 0027 POC |
| <b>0210</b> | 0050 SA  | 0051 SFE | 0052 SM  | 0053 CSP | 0054 MFA | 0011 SPS | 0012 RPT | 0033 CU1 |
| <b>0220</b> | 0060     | 0061     | 0032 UBS | 0063 IR  | 0064 PP  | 0065 TRN | 0066 NBS | 0010 GE  |
| <b>0230</b> | 0070 SBS | 0071 IT  | 0072 RFF | 0073 CU3 | 0004 SEL | 0024 RES | 0076     | 0341     |
| <b>0240</b> | 0101     | 0102     | 0103     | 0104     | 0105     | 0106     | 0107     | 0110     |
| <b>0250</b> | 0111     | 0121     | 0122     | 0123     | 0124     | 0125     | 0126     | 0127     |
| <b>0260</b> | 0130     | 0131     | 0142     | 0143     | 0144     | 0145     | 0146     | 0147     |
| <b>0270</b> | 0150     | 0151     | 0160     | 0161     | 0162     | 0163     | 0164     | 0165     |
| <b>0300</b> | 0166     | 0167     | 0170     | 0200     | 0212     | 0213     | 0214     | 0215     |
| <b>0310</b> | 0216     | 0217     | 0220     | 0152 ¡   | 0233     | 0234     | 0235     | 0236     |
| <b>0320</b> | 0237     | 0240     | 0252     | 0253     | 0254     | 0112 ¢   | 0256     | 0257     |
| <b>0330</b> | 0260     | 0261     | 0262     | 0263     | 0264     | 0265     | 0266     | 0267     |
| <b>0340</b> | 0270     | 0271     | 0272     | 0273     | 0274     | 0241     | 0276     | 0277     |
| <b>0350</b> | 0312     | 0313     | 0314 ¢   | 0315     | 0316 ¥   | 0317     | 0332     | 0333     |
| <b>0360</b> | 0334     | 0335     | 0336     | 0337     | 0352     | 0353     | 0354 ¢   | 0355     |
| <b>0370</b> | 0356     | 0357     | 0372     | 0373     | 0374     | 0375     | 0376     | 0377 EO  |

Table 4-8 ASCII to IBM EBCDIC Conversion

|             | 0        | 1        | 2        | 3        | 4        | 5        | 6        | 7        |
|-------------|----------|----------|----------|----------|----------|----------|----------|----------|
| <b>0000</b> | 0000 NUL | 0001 SOH | 0002 STX | 0003 ETX | 0067 EOT | 0055 ENQ | 0056 ACK | 0057 BEL |
| <b>0010</b> | 0026 BS  | 0005 HT  | 0045 LF  | 0013 VT  | 0014 FF  | 0015 CR  | 0016 SO  | 0017 SI  |
| <b>0020</b> | 0020 DLE | 0021 DC1 | 0022 DC2 | 0023 DC3 | 0074 DC4 | 0075 NAK | 0062 SYN | 0046 ETB |
| <b>0030</b> | 0030 CAN | 0031 EM  | 0077 SUB | 0047 ESC | 0034 IFS | 0035 IGS | 0036 IRS | 0037 ITB |
| <b>0040</b> | 0100 Sp  | 0132 !   | 0177 "   | 0173 #   | 0133 \$  | 0154 %   | 0120 &   | 0175 '   |
| <b>0050</b> | 0115 (   | 0135 )   | 0134 *   | 0116 +   | 0153 ,   | 0140 -   | 0113 .   | 0141 /   |
| <b>0060</b> | 0360 0   | 0361 1   | 0362 2   | 0363 3   | 0364 4   | 0365 5   | 0366 6   | 0367 7   |
| <b>0070</b> | 0370 8   | 0371 9   | 0172 :   | 0136 ;   | 0114 <   | 0176 =   | 0156 >   | 0157 ?   |
| <b>0100</b> | 0174 @   | 0301 A   | 0302 B   | 0303 C   | 0304 D   | 0305 E   | 0306 F   | 0307 G   |
| <b>0110</b> | 0310 H   | 0311 I   | 0321 J   | 0322 K   | 0323 L   | 0324 M   | 0325 N   | 0326 O   |
| <b>0120</b> | 0327 P   | 0330 Q   | 0331 R   | 0342 S   | 0343 T   | 0344 U   | 0345 V   | 0346 W   |
| <b>0130</b> | 0347 X   | 0350 Y   | 0351 Z   | 0255 [   | 0340 \   | 0275 ]   | 0137 ¬   | 0155 _   |
| <b>0140</b> | 0171 `   | 0201 a   | 0202 b   | 0203 c   | 0204 d   | 0205 e   | 0206 f   | 0207 g   |
| <b>0150</b> | 0210 h   | 0211 i   | 0221 j   | 0222 k   | 0223 l   | 0224 m   | 0225 n   | 0226 o   |
| <b>0160</b> | 0227 p   | 0230 q   | 0231 r   | 0242 s   | 0243 t   | 0244 u   | 0245 v   | 0246 w   |
| <b>0170</b> | 0247 x   | 0250 y   | 0251 z   | 0300 {   | 0117     | 0320 }   | 0241 ¨   | 0007 DEL |
| <b>0200</b> | 0040 DS  | 0041 SOS | 0042 FS  | 0043 WUS | 0044 BYP | 0025 NL  | 0006 RNL | 0027 POC |
| <b>0210</b> | 0050 SA  | 0051 SFE | 0052 SM  | 0053 CSP | 0054 MFA | 0011 SPS | 0012 RPT | 0033 CU1 |
| <b>0220</b> | 0060     | 0061     | 0032 UBS | 0063 IR  | 0064 PP  | 0065 TRN | 0066 NBS | 0010 GE  |
| <b>0230</b> | 0070 SBS | 0071 IT  | 0072 RFF | 0073 CU3 | 0004 SEL | 0024 RES | 0076     | 0341     |
| <b>0240</b> | 0101     | 0102     | 0103     | 0104     | 0105     | 0106     | 0107     | 0110     |
| <b>0250</b> | 0111     | 0121     | 0122     | 0123     | 0124     | 0125     | 0126     | 0127     |
| <b>0260</b> | 0130     | 0131     | 0142     | 0143     | 0144     | 0145     | 0146     | 0147     |
| <b>0270</b> | 0150     | 0151     | 0160     | 0161     | 0162     | 0163     | 0164     | 0165     |
| <b>0300</b> | 0166     | 0167     | 0170     | 0200     | 0212     | 0213     | 0214     | 0215     |
| <b>0310</b> | 0216     | 0217     | 0220     | 0232     | 0233     | 0234     | 0235     | 0236     |
| <b>0320</b> | 0237     | 0240     | 0252     | 0253     | 0254     | 0255 [   | 0256     | 0257     |
| <b>0330</b> | 0260     | 0261     | 0262     | 0263     | 0264     | 0265     | 0266     | 0267     |
| <b>0340</b> | 0270     | 0271     | 0272     | 0273     | 0274     | 0275 ]   | 0276     | 0277     |
| <b>0350</b> | 0312     | 0313     | 0314 Œ   | 0315     | 0316 Ÿ   | 0317     | 0332     | 0333     |
| <b>0360</b> | 0334     | 0335     | 0336     | 0337     | 0352     | 0353     | 0354 ¨   | 0355     |
| <b>0370</b> | 0356     | 0357     | 0372     | 0373     | 0374     | 0375     | 0376     | 0377 EO  |

85703 **STDIN**

85704 If no **if=** operand is specified, the standard input shall be used. See the INPUT FILES section.

85705 **INPUT FILES**

85706 The input file can be any file type.

85707 **ENVIRONMENT VARIABLES**

85708 The following environment variables shall affect the execution of *dd*:

85709 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 85710 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 85711 variables used to determine the values of locale categories.)

85712 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 85713 internationalization variables.

85714 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 85715 characters (for example, single-byte as opposed to multi-byte characters in  
 85716 arguments and input files), the classification of characters as uppercase or  
 85717 lowercase, and the mapping of characters from one case to the other.

85718 **LC\_MESSAGES**

85719 Determine the locale that should be used to affect the format and contents of  
 85720 diagnostic messages written to standard error and informative messages written to  
 85721 standard output.

85722 **XSI NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

85723 **ASYNCHRONOUS EVENTS**

85724 For SIGINT, the *dd* utility shall interrupt its current processing, write status information to  
 85725 standard error, and exit as though terminated by SIGINT. It shall take the standard action for all  
 85726 other signals; see the ASYNCHRONOUS EVENTS section in [Section 1.4](#) (on page 2326).

85727 **STDOUT**

85728 If no **of=** operand is specified, the standard output shall be used. The nature of the output  
 85729 depends on the operands selected.

85730 **STDERR**

85731 On completion, *dd* shall write the number of input and output blocks to standard error. In the  
 85732 POSIX locale the following formats shall be used:

85733 "%u+%u records in\n", <number of whole input blocks>,  
 85734 <number of partial input blocks>

85735 "%u+%u records out\n", <number of whole output blocks>,  
 85736 <number of partial output blocks>

85737 A partial input block is one for which *read()* returned less than the input block size. A partial  
 85738 output block is one that was written with fewer bytes than specified by the output block size.

85739 In addition, when there is at least one truncated block, the number of truncated blocks shall be  
 85740 written to standard error. In the POSIX locale, the format shall be:

85741 "%u truncated %s\n", <number of truncated blocks>, "record" (if  
 85742 <number of truncated blocks> is one) "records" (otherwise)

85743 Diagnostic messages may also be written to standard error.



**OUTPUT FILES**

If the **of=** operand is used, the output shall be the same as described in the STDOUT section.

**EXTENDED DESCRIPTION**

None.

**EXIT STATUS**

The following exit values shall be returned:

0 The input file was copied successfully.

>0 An error occurred.

**CONSEQUENCES OF ERRORS**

If an input error is detected and the **noerror** conversion has not been specified, any partial output block shall be written to the output file, a diagnostic message shall be written, and the copy operation shall be discontinued. If some other error is detected, a diagnostic message shall be written and the copy operation shall be discontinued.

**APPLICATION USAGE**

The input and output block size can be specified to take advantage of raw physical I/O.

There are many different versions of the EBCDIC codesets. The ASCII and EBCDIC conversions specified for the *dd* utility perform conversions for the version specified by the tables.

**EXAMPLES**

The following command:

```
dd if=/dev/rmt0h of=/dev/rmt1h
```

copies from tape drive 0 to tape drive 1, using a common historical device naming convention.

The following command:

```
dd ibs=10 skip=1
```

strips the first 10 bytes from standard input.

This example reads an EBCDIC tape blocked ten 80-byte EBCDIC card images per block into the ASCII file *x*:

```
dd if=/dev/tape of=x ibs=800 cbs=80 conv=ascii,lcase
```

**RATIONALE**

The **OPTIONS** section is listed as “None” because there are no options recognized by historical *dd* utilities. Certainly, many of the operands could have been designed to use the Utility Syntax Guidelines, which would have resulted in the classic hyphenated option letters. In this version of this volume of POSIX.1-2008, *dd* retains its curious JCL-like syntax due to the large number of applications that depend on the historical implementation.

A suggested implementation technique for **conv=noerror, sync** is to zero (or <space>-fill, if **blocking** or **unblocking**) the input buffer before each read and to write the contents of the input buffer to the output even after an error. In this manner, any data transferred to the input buffer before the error was detected is preserved. Another point is that a failed read on a regular file or a disk generally does not increment the file offset, and *dd* must then seek past the block on which the error occurred; otherwise, the input error occurs repetitively. When the input is a magnetic tape, however, the tape normally has passed the block containing the error when the error is reported, and thus no seek is necessary.

The default **ibs=** and **obs=** sizes are specified as 512 bytes because there are historical (largely portable) scripts that assume these values. If they were left unspecified, unusual results could



occur if an implementation chose an odd block size.

Historical implementations of *dd* used *creat()* when processing *of=file*. This makes the **seek=** operand unusable except on special files. The **conv=notrunc** feature was added because more recent BSD-based implementations use *open()* (without *O\_TRUNC*) instead of *creat()*, but they fail to delete output file contents after the data copied.

The *w* multiplier (historically meaning *word*), is used in System V to mean 2 and in 4.2 BSD to mean 4. Since *word* is inherently non-portable, its use is not supported by this volume of POSIX.1-2008.

Standard EBCDIC does not have the characters '[' and ']'. The values used in the table are taken from a common print train that does contain them. Other than those characters, the print train values are not filled in, but appear to provide some of the motivation for the historical choice of translations reflected here.

The Standard EBCDIC table provides a 1:1 translation for all 256 bytes.

The IBM EBCDIC table does not provide such a translation. The marked cells in the tables differ in such a way that:

1. EBCDIC 0112 ('&') and 0152 (broken pipe) do not appear in the table.
2. EBCDIC 0137 ('¬') translates to/from ASCII 0236 ('^'). In the standard table, EBCDIC 0232 (no graphic) is used.
3. EBCDIC 0241 ('~') translates to/from ASCII 0176 ('~'). In the standard table, EBCDIC 0137 ('¬') is used.
4. 0255 ('[') and 0275 (']') appear twice, once in the same place as for the standard table and once in place of 0112 ('&') and 0241 ('~').

In net result:

EBCDIC 0275 (']') displaced EBCDIC 0241 ('~') in cell 0345.

That displaced EBCDIC 0137 ('¬') in cell 0176.

That displaced EBCDIC 0232 (no graphic) in cell 0136.

That replaced EBCDIC 0152 (broken pipe) in cell 0313.

EBCDIC 0255 ('[') replaced EBCDIC 0112 ('&').

This translation, however, reflects historical practice that (ASCII) '~' and '¬' were often mapped to each other, as were '[' and '&'; and ']' and (EBCDIC) '~'.

The **cbs** operand is required if any of the **ascii**, **ebcdic**, or **ibm** operands are specified. For the **ascii** operand, the input is handled as described for the **unblock** operand except that characters are converted to ASCII before the trailing <space> characters are deleted. For the **ebcdic** and **ibm** operands, the input is handled as described for the **block** operand except that the characters are converted to EBCDIC or IBM EBCDIC after the trailing <space> characters are added.

The **block** and **unblock** keywords are from historical BSD practice.

The consistent use of the word **record** in standard error messages matches most historical practice. An earlier version of System V used **block**, but this has been updated in more recent releases.

Early proposals only allowed two numbers separated by *x* to be used in a product when specifying **bs=**, **cbs=**, **ibs=**, and **obs=** sizes. This was changed to reflect the historical practice of

- 85828 allowing multiple numbers in the product as provided by Version 7 and all releases of System V  
85829 and BSD.
- 85830 A change to the **swab** conversion is required to match historical practice and is the result of IEEE  
85831 PASC Interpretations 1003.2 #03 and #04, submitted for the ISO POSIX-2: 1993 standard.
- 85832 A change to the handling of SIGINT is required to match historical practice and is the result of  
85833 IEEE PASC Interpretation 1003.2 #06 submitted for the ISO POSIX-2: 1993 standard.
- 85834 **FUTURE DIRECTIONS**
- 85835 None.
- 85836 **SEE ALSO**
- 85837 [Section 1.4](#) (on page 2326), *sed*, *tr*
- 85838 XBD [Chapter 8](#) (on page 173)
- 85839 **CHANGE HISTORY**
- 85840 First released in Issue 2.
- 85841 **Issue 5**
- 85842 The second paragraph of the **cbs=** description is reworded and marked EX.
- 85843 The FUTURE DIRECTIONS section is added.
- 85844 **Issue 6**
- 85845 Changes are made to **swab** conversion and SIGINT handling to align with the IEEE P1003.2b  
85846 draft standard.
- 85847 The normative text is reworded to avoid use of the term “must” for application requirements.
- 85848 IEEE PASC Interpretation 1003.2 #209 is applied, clarifying the interaction between *dd of=file* and  
85849 **conv=notrunc**.
- 85850 **Issue 7**
- 85851 Austin Group Interpretation 1003.1-2001 #102 is applied.
- 85852 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

85853 **NAME**85854       delta — make a delta (change) to an SCCS file (**DEVELOPMENT**)85855 **SYNOPSIS**

85856 XSI       delta [-nps] [-g list] [-m mrlist] [-r SID] [-y[comment]] file...

85857 **DESCRIPTION**85858       The *delta* utility shall be used to permanently introduce into the named SCCS files changes that  
85859       were made to the files retrieved by *get* (called the *g-files*, or generated files).85860 **OPTIONS**85861       The *delta* utility shall conform to XBD [Section 12.2](#) (on page 216), except that the *-y* option has an  
85862       optional option-argument. This optional option-argument shall not be presented as a separate  
85863       argument.

85864       The following options shall be supported:

85865       *-r SID*       Uniquely identify which delta is to be made to the SCCS file. The use of this option  
85866       shall be necessary only if two or more outstanding *get* commands for editing (*get*  
85867       *-e*) on the same SCCS file were done by the same person (login name). The SID  
85868       value specified with the *-r* option can be either the SID specified on the *get*  
85869       command line or the SID to be made as reported by the *get* utility; see *get* (on page  
85870       2812).85871       *-s*           Suppress the report to standard output of the activity associated with each *file*. See  
85872       the STDOUT section.85873       *-n*           Specify retention of the edited *g-file* (normally removed at completion of delta  
85874       processing).85875       *-g list*       Specify a *list* (see *get* for the definition of *list*) of deltas that shall be ignored when  
85876       the file is accessed at the change level (SID) created by this delta.85877       *-m mrlist*     Specify a modification request (MR) number that the application shall supply as  
85878       the reason for creating the new delta. This shall be used if the SCCS file has the *v*  
85879       flag set; see *admin*.85880       If *-m* is not used and *'-'* is not specified as a file argument, and the standard  
85881       input is a terminal, the prompt described in the STDOUT section shall be written  
85882       to standard output before the standard input is read; if the standard input is not a  
85883       terminal, no prompt shall be issued.85884       MRs in a list shall be separated by <blank> characters or escaped <newline>  
85885       characters. An unescaped <newline> shall terminate the MR list. The escape  
85886       character is <backslash>.85887       If the *v* flag has a value, it shall be taken to be the name of a program which  
85888       validates the correctness of the MR numbers. If a non-zero exit status is returned  
85889       from the MR number validation program, the *delta* utility shall terminate. (It is  
85890       assumed that the MR numbers were not all valid.)85891       *-y[comment]*   Describe the reason for making the delta. The *comment* shall be an arbitrary group  
85892       of lines that would meet the definition of a text file. Implementations shall support  
85893       *comments* from zero to 512 bytes and may support longer values. A null string  
85894       (specified as either *-y*, *-y" "*, or in response to a prompt for a comment) shall be  
85895       considered a valid *comment*.85896       If *-y* is not specified and *'-'* is not specified as a file argument, and the standard

85897 input is a terminal, the prompt described in the STDOUT section shall be written  
 85898 to standard output before the standard input is read; if the standard input is not a  
 85899 terminal, no prompt shall be issued. An unescaped <newline> shall terminate the  
 85900 comment text. The escape character is <backslash>.

85901 The `-y` option shall be required if the *file* operand is specified as `'-'`.

85902 **-p** Write (to standard output) the SCCS file differences before and after the delta is  
 85903 applied in *diff* format; see *diff*.

## 85904 OPERANDS

85905 The following operand shall be supported:

85906 *file* A pathname of an existing SCCS file or a directory. If *file* is a directory, the *delta*  
 85907 utility shall behave as though each file in the directory were specified as a named  
 85908 file, except that non-SCCS files (last component of the pathname does not begin  
 85909 with *s*.) and unreadable files shall be silently ignored.

85910 If exactly one *file* operand appears, and it is `'-'`, the standard input shall be read;  
 85911 each line of the standard input shall be taken to be the name of an SCCS file to be  
 85912 processed. Non-SCCS files and unreadable files shall be silently ignored.

## 85913 STDIN

85914 The standard input shall be a text file used only in the following cases:

- 85915 • To read an *mrlist* or a *comment* (see the `-m` and `-y` options).
- 85916 • A *file* operand shall be specified as `'-'`. In this case, the `-y` option must be used to specify  
 85917 the comment, and if the SCCS file has the *v* flag set, the `-m` option must also be used to  
 85918 specify the MR list.

## 85919 INPUT FILES

85920 Input files shall be text files whose data is to be included in the SCCS files. If the first character of  
 85921 any line of an input file is <SOH> in the POSIX locale, the results are unspecified. If this file  
 85922 contains more than 99 999 lines, the number of lines recorded in the header for this file shall be  
 85923 99 999 for this delta.

## 85924 ENVIRONMENT VARIABLES

85925 The following environment variables shall affect the execution of *delta*:

85926 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 85927 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 85928 variables used to determine the values of locale categories.)

85929 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 85930 internationalization variables.

85931 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 85932 characters (for example, single-byte as opposed to multi-byte characters in  
 85933 arguments and input files).

85934 *LC\_MESSAGES*

85935 Determine the locale that should be used to affect the format and contents of  
 85936 diagnostic messages written to standard error, and informative messages written  
 85937 to standard output.

85938 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

85939 **TZ** Determine the timezone in which the time and date are written in the SCCS file. If  
 85940 the *TZ* variable is unset or NULL, an unspecified system default timezone is used.

#### 85941 **ASYNCHRONOUS EVENTS**

85942 If SIGINT is caught, temporary files shall be cleaned up and *delta* shall exit with a non-zero exit  
 85943 code. The standard action shall be taken for all other signals; see [Section 1.4](#) (on page 2326).

#### 85944 **STDOUT**

85945 The standard output shall be used only for the following messages in the POSIX locale:

- 85946 • Prompts (see the *-m* and *-y* options) in the following formats:

85947 "MRs? "

85948 "comments? "

85949 The MR prompt, if written, shall always precede the comments prompt.

- 85950 • A report of each file's activities (unless the *-s* option is specified) in the following format:

85951 "%s\n%d inserted\n%d deleted\n%d unchanged\n", <New SID>,  
 85952 <number of lines inserted>, <number of lines deleted>,  
 85953 <number of lines unchanged>

#### 85954 **STDERR**

85955 The standard error shall be used only for diagnostic messages.

#### 85956 **OUTPUT FILES**

85957 Any SCCS files updated shall be files of an unspecified format.

#### 85958 **EXTENDED DESCRIPTION**

##### 85959 **System Date and Time**

85960 When a *delta* is added to an SCCS file, the system date and time shall be recorded for the new  
 85961 delta. If a *get* is performed using an SCCS file with a date recorded apparently in the future, the  
 85962 behavior is unspecified.

#### 85963 **EXIT STATUS**

85964 The following exit values shall be returned:

85965 0 Successful completion.

85966 >0 An error occurred.

#### 85967 **CONSEQUENCES OF ERRORS**

85968 Default.

#### 85969 **APPLICATION USAGE**

85970 Problems can arise if the system date and time have been modified (for example, put forward  
 85971 and then back again, or unsynchronized clocks across a network) and can also arise when  
 85972 different values of the *TZ* environment variable are used.

85973 Problems of a similar nature can also arise for the operation of the *get* utility, which records the  
 85974 date and time in the file body.

#### 85975 **EXAMPLES**

85976 None.

85977 **RATIONALE**

85978 None.

85979 **FUTURE DIRECTIONS**

85980 None.

85981 **SEE ALSO**85982 [Section 1.4](#) (on page 2326), *admin*, *diff*, *get*, *prs*, *rmdel*85983 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)85984 **CHANGE HISTORY**

85985 First released in Issue 2.

85986 **Issue 5**

85987 The output format description in the STDOUT section is corrected.

85988 **Issue 6**

85989 The APPLICATION USAGE section is added.

85990 The normative text is reworded to avoid use of the term “must” for application requirements.

85991 The Open Group Base Resolution bwg2001-007 is applied as follows:

- 85992 • The use of ‘–’ as a file argument is clarified.
- 85993 • The use of STDIN is added.
- 85994 • The ASYNCHRONOUS EVENTS section is updated to remove the implicit requirement
- 85995 that implementations re-signal themselves when catching a normally fatal signal.
- 85996 • New text is added to the INPUT FILES section warning that the maximum lines recorded
- 85997 in the file is 99 999.

85998 New text is added to the EXTENDED DESCRIPTION and APPLICATION USAGE sections  
85999 regarding how the system date and time may be taken into account, and the TZ environment  
86000 variable is added to the ENVIRONMENT VARIABLES section as per The Open Group Base  
86001 Resolution bwg2001-007.

86002 **Issue 7**

86003 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

86004 **NAME**

86005           df — report free disk space

86006 **SYNOPSIS**86007 XSI       df [-k] [-P] [-t] [*file...*]86008 **DESCRIPTION**

86009 XSI       The *df* utility shall write the amount of available space and file slots for file systems on which  
 86010 the invoking user has appropriate read access. File systems shall be specified by the *file*  
 86011 operands; when none are specified, information shall be written for all file systems. The format  
 86012 of the default output from *df* is unspecified, but all space figures are reported in 512-byte units,  
 86013 unless the **-k** option is specified. This output shall contain at least the file system names, amount  
 86014 XSI       of available space on each of these file systems, and, if no options other than **-t** are specified, the  
 86015 number of free file slots, or *inodes*, available; when **-t** is specified, the output shall contain the  
 86016 total allocated space as well.

86017 **OPTIONS**86018           The *df* utility shall conform to XBD Section 12.2 (on page 216).

86019           The following options shall be supported:

86020       **-k**           Use 1024-byte units, instead of the default 512-byte units, when writing space  
 86021 figures.

86022       **-P**           Produce output in the format described in the STDOUT section.

86023 XSI       **-t**           Include total allocated-space figures in the output.

86024 **OPERANDS**

86025           The following operand shall be supported:

86026       *file*       A pathname of a file within the hierarchy of the desired file system. If a file other than a FIFO, a regular file, a directory, or a special file representing the device  
 86027 XSI       containing the file system (for example, */dev/dsk/0s1*) is specified, the results are  
 86028 unspecified. If the *file* operand names a file other than a special file containing a file  
 86029 system, *df* shall write the amount of free space in the file system containing the  
 86030 specified *file* operand. Otherwise, *df* shall write the amount of free space in that  
 86031 XSI       file system.  
 86032

86033 **STDIN**

86034           Not used.

86035 **INPUT FILES**

86036           None.

86037 **ENVIRONMENT VARIABLES**86038           The following environment variables shall affect the execution of *df*:

86039       **LANG**       Provide a default value for the internationalization variables that are unset or null.  
 86040 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 86041 variables used to determine the values of locale categories.)

86042       **LC\_ALL**      If set to a non-empty string value, override the values of all the other  
 86043 internationalization variables.

86044       **LC\_CTYPE**   Determine the locale for the interpretation of sequences of bytes of text data as  
 86045 characters (for example, single-byte as opposed to multi-byte characters in  
 86046 arguments).



86047 *LC\_MESSAGES*  
 86048 Determine the locale that should be used to affect the format and contents of  
 86049 diagnostic messages written to standard error and informative messages written to  
 86050 standard output.

86051 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

86052 **ASYNCHRONOUS EVENTS**  
 86053 Default.

86054 **STDOUT**  
 86055 When both the **-k** and **-P** options are specified, the following header line shall be written (in the  
 86056 POSIX locale):

86057 "Filesystem 1024-blocks Used Available Capacity Mounted on\n"

86058 When the **-P** option is specified without the **-k** option, the following header line shall be written  
 86059 (in the POSIX locale):

86060 "Filesystem 512-blocks Used Available Capacity Mounted on\n"

86061 The implementation may adjust the spacing of the header line and the individual data lines so  
 86062 that the information is presented in orderly columns.

86063 The remaining output with **-P** shall consist of one line of information for each specified file  
 86064 system. These lines shall be formatted as follows:

86065 "%s %d %d %d %d%% %s\n", <file system name>, <total space>,  
 86066 <space used>, <space free>, <percentage used>,  
 86067 <file system root>

86068 In the following list, all quantities expressed in 512-byte units (1 024-byte when **-k** is specified)  
 86069 shall be rounded up to the next higher unit. The fields are:

86070 <file system name>  
 86071 The name of the file system, in an implementation-defined format.

86072 <total space> The total size of the file system in 512-byte units. The exact meaning of this figure  
 86073 is implementation-defined, but should include <space used>, <space free>, plus any  
 86074 space reserved by the system not normally available to a user.

86075 <space used> The total amount of space allocated to existing files in the file system, in 512-byte  
 86076 units.

86077 <space free> The total amount of space available within the file system for the creation of new  
 86078 files by unprivileged users, in 512-byte units. When this figure is less than or equal  
 86079 to zero, it shall not be possible to create any new files on the file system without  
 86080 first deleting others, unless the process has appropriate privileges. The figure  
 86081 written may be less than zero.

86082 <percentage used>  
 86083 The percentage of the normally available space that is currently allocated to all files  
 86084 on the file system. This shall be calculated using the fraction:

86085 
$$\frac{\text{<space used>}}{(\text{<space used>} + \text{<space free>})}$$
  
 86086 expressed as a percentage. This percentage may be greater than 100 if <space free>  
 86087 is less than zero. The percentage value shall be expressed as a positive integer, with  
 86088 any fractional result causing it to be rounded to the next highest integer.



86089 <file system root>

86090 The directory below which the file system hierarchy appears.

86091 XSI The output format is unspecified when **-t** is used.

## 86092 STDERR

86093 The standard error shall be used only for diagnostic messages.

## 86094 OUTPUT FILES

86095 None.

## 86096 EXTENDED DESCRIPTION

86097 None.

## 86098 EXIT STATUS

86099 The following exit values shall be returned:

86100 0 Successful completion.

86101 >0 An error occurred.

## 86102 CONSEQUENCES OF ERRORS

86103 Default.

## 86104 APPLICATION USAGE

86105 On most systems, the “name of the file system, in an implementation-defined format” is the  
86106 special file on which the file system is mounted.

86107 On large file systems, the calculation specified for percentage used can create huge rounding  
86108 errors.

## 86109 EXAMPLES

86110 1. The following example writes portable information about the **/usr** file system:

86111 `df -P /usr`

86112 2. Assuming that **/usr/src** is part of the **/usr** file system, the following produces the same  
86113 output as the previous example:

86114 `df -P /usr/src`

## 86115 RATIONALE

86116 The behavior of *df* with the **-P** option is the default action of the 4.2 BSD *df* utility. The uppercase  
86117 **-P** was selected to avoid collision with a known industry extension using **-p**.

86118 Historical *df* implementations vary considerably in their default output. It was therefore  
86119 necessary to describe the default output in a loose manner to accommodate all known historical  
86120 implementations and to add a portable option (**-P**) to provide information in a portable format.

86121 The use of 512-byte units is historical practice and maintains compatibility with *ls* and other  
86122 utilities in this volume of POSIX.1-2008. This does not mandate that the file system itself be  
86123 based on 512-byte blocks. The **-k** option was added as a compromise measure. It was agreed by  
86124 the standard developers that 512 bytes was the best default unit because of its complete  
86125 historical consistency on System V (*versus* the mixed 512/1024-byte usage on BSD systems), and  
86126 that a **-k** option to switch to 1024-byte units was a good compromise. Users who prefer the  
86127 more logical 1024-byte quantity can easily alias *df* to *df -k* without breaking many historical  
86128 scripts relying on the 512-byte units.

86129 It was suggested that *df* and the various related utilities be modified to access a *BLOCKSIZE*  
86130 environment variable to achieve consistency and user acceptance. Since this is not historical

86131 practice on any system, it is left as a possible area for system extensions and will be re-evaluated  
86132 in a future version if it is widely implemented.

86133 **FUTURE DIRECTIONS**

86134 None.

86135 **SEE ALSO**

86136 *find*

86137 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

86138 **CHANGE HISTORY**

86139 First released in Issue 2.

86140 **Issue 6**

86141 This utility is marked as part of the User Portability Utilities option.

86142 **Issue 7**

86143 Austin Group Interpretation 1003.1-2001 #099 is applied.

86144 The *df* utility is removed from the User Portability Utilities option. User Portability Utilities is  
86145 now an option for interactive utilities.

86146 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

86147 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0082 [156] is applied.

86148 **NAME**

86149 diff — compare two files

86150 **SYNOPSIS**86151 diff [-c|-e|-f|-u|-C *n*|-U *n*] [-br] *file1 file2*86152 **DESCRIPTION**

86153 The *diff* utility shall compare the contents of *file1* and *file2* and write to standard output a list of  
 86154 changes necessary to convert *file1* into *file2*. This list should be minimal. No output shall be  
 86155 produced if the files are identical.

86156 **OPTIONS**86157 The *diff* utility shall conform to XBD [Section 12.2](#) (on page 216).

86158 The following options shall be supported:

86159 **-b** Cause any amount of white space at the end of a line to be treated as a single  
 86160 <newline> (that is, the white-space characters preceding the <newline> are  
 86161 ignored) and other strings of white-space characters, not including <newline>  
 86162 characters, to compare equal.

86163 **-c** Produce output in a form that provides three lines of copied context.

86164 **-C *n*** Produce output in a form that provides *n* lines of copied context (where *n* shall be  
 86165 interpreted as a positive decimal integer).

86166 **-e** Produce output in a form suitable as input for the *ed* utility, which can then be used  
 86167 to convert *file1* into *file2*.

86168 **-f** Produce output in an alternative form, similar in format to **-e**, but not intended to  
 86169 be suitable as input for the *ed* utility, and in the opposite order.

86170 **-r** Apply *diff* recursively to files and directories of the same name when *file1* and *file2*  
 86171 are both directories.

86172 The *diff* utility shall detect infinite loops; that is, entering a previously visited  
 86173 directory that is an ancestor of the last file encountered. When it detects an infinite  
 86174 loop, *diff* shall write a diagnostic message to standard error and shall either recover  
 86175 its position in the hierarchy or terminate.

86176 **-u** Produce output in a form that provides three lines of unified context.

86177 **-U *n*** Produce output in a form that provides *n* lines of unified context (where *n* shall be  
 86178 interpreted as a non-negative decimal integer).

86179 **OPERANDS**

86180 The following operands shall be supported:

86181 *file1, file2* A pathname of a file to be compared. If either the *file1* or *file2* operand is '-', the  
 86182 standard input shall be used in its place.

86183 If both *file1* and *file2* are directories, *diff* shall not compare block special files, character special  
 86184 files, or FIFO special files to any files and shall not compare regular files to directories. Further  
 86185 details are as specified in [Diff Directory Comparison Format](#) (on page 2648). The behavior of *diff*  
 86186 on other file types is implementation-defined when found in directories.

86187 If only one of *file1* and *file2* is a directory, *diff* shall be applied to the non-directory file and the file  
 86188 contained in the directory file with a filename that is the same as the last component of the non-  
 86189 directory file.

86190 **STDIN**

86191 The standard input shall be used only if one of the *file1* or *file2* operands references standard  
86192 input. See the INPUT FILES section.

86193 **INPUT FILES**

86194 The input files may be of any type.

86195 **ENVIRONMENT VARIABLES**

86196 The following environment variables shall affect the execution of *diff*:

86197 **LANG** Provide a default value for the internationalization variables that are unset or null.  
86198 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
86199 variables used to determine the values of locale categories.)

86200 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
86201 internationalization variables.

86202 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
86203 characters (for example, single-byte as opposed to multi-byte characters in  
86204 arguments and input files).

86205 **LC\_MESSAGES**

86206 Determine the locale that should be used to affect the format and contents of  
86207 diagnostic messages written to standard error and informative messages written to  
86208 standard output.

86209 **LC\_TIME** Determine the locale for affecting the format of file timestamps written with the **-C**  
86210 and **-c** options.

86211 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

86212 **TZ** Determine the timezone used for calculating file timestamps written with a context  
86213 format. If **TZ** is unset or null, an unspecified default timezone shall be used.

86214 **ASYNCHRONOUS EVENTS**

86215 Default.

86216 **STDOUT**86217 **Diff Directory Comparison Format**

86218 If both *file1* and *file2* are directories, the following output formats shall be used.

86219 In the POSIX locale, each file that is present in only one directory shall be reported using the  
86220 following format:

86221 "Only in %s: %s\n", <directory pathname>, <filename>

86222 In the POSIX locale, subdirectories that are common to the two directories may be reported with  
86223 the following format:

86224 "Common subdirectories: %s and %s\n", <directory1 pathname>,  
86225 <directory2 pathname>

86226 For each file common to the two directories, if the two files are not to be compared: if the two  
86227 files have the same device ID and file serial number, or are both block special files that refer to  
86228 the same device, or are both character special files that refer to the same device, in the POSIX  
86229 locale the output format is unspecified. Otherwise, in the POSIX locale an unspecified format  
86230 shall be used that contains the pathnames of the two files.

86231 For each file common to the two directories, if the files are compared and are identical, no

output shall be written. If the two files differ, the following format is written:

```
"diff %s %s %s\n", <diff_options>, <filename1>, <filename2>
```

where *<diff\_options>* are the options as specified on the command line.

All directory pathnames listed in this section shall be relative to the original command line arguments. All other names of files listed in this section shall be filenames (pathname components).

### Diff Binary Output Format

In the POSIX locale, if one or both of the files being compared are not text files, it is implementation-defined whether *diff* uses the binary file output format or the other formats as specified below. The binary file output format shall contain the pathnames of two files being compared and the string "differ".

If both files being compared are text files, depending on the options specified, one of the following formats shall be used to write the differences.

### Diff Default Output Format

The default (without *-e*, *-f*, *-c*, *-C*, *-u*, or *-U* options) *diff* utility output shall contain lines of these forms:

```
"%da%d\n", <num1>, <num2>
```

```
"%da%d,%d\n", <num1>, <num2>, <num3>
```

```
"%dd%d\n", <num1>, <num2>
```

```
"%d,%dd%d\n", <num1>, <num2>, <num3>
```

```
"%dc%d\n", <num1>, <num2>
```

```
"%d,%dc%d\n", <num1>, <num2>, <num3>
```

```
"%dc%d,%d\n", <num1>, <num2>, <num3>
```

```
"%d,%dc%d,%d\n", <num1>, <num2>, <num3>, <num4>
```

These lines resemble *ed* subcommands to convert *file1* into *file2*. The line numbers before the action letters shall pertain to *file1*; those after shall pertain to *file2*. Thus, by exchanging *a* for *d* and reading the line in reverse order, one can also determine how to convert *file2* into *file1*. As in *ed*, identical pairs (where *num1*=*num2*) are abbreviated as a single number.

Following each of these lines, *diff* shall write to standard output all lines affected in the first file using the format:

```
"<Δ%s", <line>
```

and all lines affected in the second file using the format:

```
">Δ%s", <line>
```

If there are lines affected in both *file1* and *file2* (as with the *c* subcommand), the changes are separated with a line consisting of three <hyphen-minus> characters:

```
"---\n"
```

**Diff -e Output Format**

With the **-e** option, a script shall be produced that shall, when provided as input to *ed*, along with an appended **w** (write) command, convert *file1* into *file2*. Only the **a** (append), **c** (change), **d** (delete), **i** (insert), and **s** (substitute) commands of *ed* shall be used in this script. Text lines, except those consisting of the single character <period> ( ' . ' ), shall be output as they appear in the file.

**Diff -f Output Format**

With the **-f** option, an alternative format of script shall be produced. It is similar to that produced by **-e**, with the following differences:

1. It is expressed in reverse sequence; the output of **-e** orders changes from the end of the file to the beginning; the **-f** from beginning to end.
2. The command form <lines> <command-letter> used by **-e** is reversed. For example, 10c with **-e** would be c10 with **-f**.
3. The form used for ranges of line numbers is <space>-separated, rather than <comma>-separated.

**Diff -c or -C Output Format**

With the **-c** or **-C** option, the output format shall consist of affected lines along with surrounding lines of context. The affected lines shall show which ones need to be deleted or changed in *file1*, and those added from *file2*. With the **-c** option, three lines of context, if available, shall be written before and after the affected lines. With the **-C** option, the user can specify how many lines of context are written. The exact format follows.

The name and last modification time of each file shall be output in the following format:

```
***  %s %s\n", file1, <file1 timestamp>
---  %s %s\n", file2, <file2 timestamp>
```

Each <file> field shall be the pathname of the corresponding file being compared. The pathname written for standard input is unspecified.

In the POSIX locale, each <timestamp> field shall be equivalent to the output from the following command:

```
date "+%a %b %e %T %Y"
```

without the trailing <newline>, executed at the time of last modification of the corresponding file (or the current time, if the file is standard input).

Then, the following output formats shall be applied for every set of changes.

First, a line shall be written in the following format:

```
"*****\n"
```

Next, the range of lines in *file1* shall be written in the following format if the range contains two or more lines:

```
***  %d,%d *****\n", <beginning line number>, <ending line number>
```

and the following format otherwise:

```
***  %d *****\n", <ending line number>
```

The ending line number of an empty range shall be the number of the preceding line, or 0 if the

86308 range is at the start of the file.

86309 Next, the affected lines along with lines of context (unaffected lines) shall be written. Unaffected  
86310 lines shall be written in the following format:

86311 " $\Delta$ %s", <unaffected\_line>

86312 Deleted lines shall be written as:

86313 " $-\Delta$ %s", <deleted\_line>

86314 Changed lines shall be written as:

86315 "! $\Delta$ %s", <changed\_line>

86316 Next, the range of lines in *file2* shall be written in the following format if the range contains two  
86317 or more lines:

86318 "--- %d,%d ----\n", <beginning line number>, <ending line number>

86319 and the following format otherwise:

86320 "--- %d ----\n", <ending line number>

86321 Then, lines of context and changed lines shall be written as described in the previous formats.  
86322 Lines added from *file2* shall be written in the following format:

86323 "+ $\Delta$ %s", <added\_line>

#### 86324 **Diff -u or -U Output Format**

86325 The -u or -U options behave like the -c or -C options, except that the context lines are not  
86326 repeated; instead, the context, deleted, and added lines are shown together, interleaved. The  
86327 exact format follows.

86328 The name and last modification time of each file shall be output in the following format:

86329 "--- $\Delta$ %s\t%s%\s $\Delta$ %s\n", file1, <file1 timestamp>, <file1 frac>, <file1 zone> |  
86330 "+++ $\Delta$ %s\t%s%\s $\Delta$ %s\n", file2, <file2 timestamp>, <file2 frac>, <file2 zone> |

86331 Each <file> field shall be the pathname of the corresponding file being compared, or the single  
86332 character '-' if standard input is being compared. However, if the pathname contains a <tab>  
86333 or a <newline>, or if it does not consist entirely of characters taken from the portable character  
86334 set, the behavior is implementation-defined.

86335 Each <timestamp> field shall be equivalent to the output from the following command:

86336 date '+%Y-%m-%d $\Delta$ %H:%M:%S'

86337 without the trailing <newline>, executed at the time of last modification of the corresponding  
86338 file (or the current time, if the file is standard input).

86339 Each <frac> field shall be either empty, or a decimal point followed by at least one decimal digit,  
86340 indicating the fractional-seconds part (if any) of the file timestamp. The number of fractional  
86341 digits shall be at least the number needed to represent the file's timestamp without loss of  
86342 information.

86343 Each <zone> field shall be of the form "shhmm", where "shh" is a signed two-digit decimal  
86344 number in the range -24 through +25, and "mm" is an unsigned two-digit decimal number in the  
86345 range 00 through 59. It represents the timezone of the timestamp as the number of hours (hh)  
86346 and minutes (mm) east (+) or west (-) of UTC for the timestamp. If the hours and minutes are  
86347 both zero, the sign shall be '+'. However, if the timezone is not an integral number of minutes

86348 away from UTC, the <zone> field is implementation-defined.

86349 Then, the following output formats shall be applied for every set of changes.

86350 First, the range of lines in each file shall be written in the following format:

86351 "@@Δ-%sΔ+%sΔ@@", <file1 range>, <file2 range>

86352 Each <range> field shall be of the form:

86353 "%ld", <beginning line number>

86354 or: +

86355 "%ld,1", <beginning line number> +

86356 if the range contains exactly one line, and:

86357 "%ld,%ld", <beginning line number>, <number of lines>

86358 otherwise. If a range is empty, its beginning line number shall be the number of the line just

86359 before the range, or 0 if the empty range starts the file.

86360 Next, the affected lines along with lines of context shall be written. Each non-empty unaffected

86361 line shall be written in the following format:

86362 "Δ%s", <unaffected\_line>

86363 where the contents of the unaffected line shall be taken from *file1*. It is implementation-defined

86364 whether an empty unaffected line is written as an empty line or a line containing a single

86365 <space> character. This line also represents the same line of *file2*, even though *file2*'s line may

86366 contain different contents due to the **-b**. Deleted lines shall be written as:

86367 "-%s", <deleted\_line>

86368 Added lines shall be written as:

86369 "+%s", <added\_line>

86370 The order of lines written shall be the same as that of the corresponding file. A deleted line shall

86371 never be written immediately after an added line.

86372 If **-U n** is specified, the output shall contain no more than  $2n$  consecutive unaffected lines; and if

86373 the output contains an affected line and this line is adjacent to up to  $n$  consecutive unaffected

86374 lines in the corresponding file, the output shall contain these unaffected lines. **-u** shall act like

86375 **-U3**.

86376 **STDERR**

86377 The standard error shall be used only for diagnostic messages.

86378 **OUTPUT FILES**

86379 None.

86380 **EXTENDED DESCRIPTION**

86381 None.

86382 **EXIT STATUS**

86383 The following exit values shall be returned:

86384 0 No differences were found.

86385 1 Differences were found.



86386 >1 An error occurred.

## 86387 CONSEQUENCES OF ERRORS

86388 Default.

## 86389 APPLICATION USAGE

86390 If lines at the end of a file are changed and other lines are added, *diff* output may show this as a  
 86391 delete and add, as a change, or as a change and add; *diff* is not expected to know which  
 86392 happened and users should not care about the difference in output as long as it clearly shows  
 86393 the differences between the files.

## 86394 EXAMPLES

86395 If **dir1** is a directory containing a directory named **x**, **dir2** is a directory containing a directory  
 86396 named **x**, **dir1/x** and **dir2/x** both contain files named **date.out**, and **dir2/x** contains a file named **y**,  
 86397 the command:

86398 `diff -r dir1 dir2`

86399 could produce output similar to:

```
86400 Common subdirectories: dir1/x and dir2/x
86401 Only in dir2/x: y
86402 diff -r dir1/x/date.out dir2/x/date.out
86403 1c1
86404 < Mon Jul 2 13:12:16 PDT 1990
86405 ---
86406 > Tue Jun 19 21:41:39 PDT 1990
```

## 86407 RATIONALE

86408 The **-h** option was omitted because it was insufficiently specified and does not add to  
 86409 applications portability.

86410 Historical implementations employ algorithms that do not always produce a minimum list of  
 86411 differences; the current language about making every effort is the best this volume of  
 86412 POSIX.1-2008 can do, as there is no metric that could be employed to judge the quality of  
 86413 implementations against any and all file contents. The statement “This list should be minimal”  
 86414 clearly implies that implementations are not expected to provide the following output when  
 86415 comparing two 100-line files that differ in only one character on a single line:

```
86416 1,100c1,100
86417 all 100 lines from file1 preceded with "< "
86418 ---
86419 all 100 lines from file2 preceded with "> "
```

86420 The “Only in” messages required when the **-r** option is specified are not used by most historical  
 86421 implementations if the **-e** option is also specified. It is required here because it provides useful  
 86422 information that must be provided to update a target directory hierarchy to match a source  
 86423 hierarchy. The “Common subdirectories” messages are written by System V and 4.3 BSD when  
 86424 the **-r** option is specified. They are allowed here but are not required because they are reporting  
 86425 on something that is the same, not reporting a difference, and are not needed to update a target  
 86426 hierarchy.

86427 The **-c** option, which writes output in a format using lines of context, has been included. The  
 86428 format is useful for a variety of reasons, among them being much improved readability and the  
 86429 ability to understand difference changes when the target file has line numbers that differ from  
 86430 another similar, but slightly different, copy. The *patch* utility is most valuable when working  
 86431 with difference listings using a context format. The BSD version of **-c** takes an optional

argument specifying the amount of context. Rather than overloading `-c` and breaking the Utility Syntax Guidelines for *diff*, the standard developers decided to add a separate option for specifying a context diff with a specified amount of context (`-C`). Also, the format for context diffs was extended slightly in 4.3 BSD to allow multiple changes that are within context lines from each other to be merged together. The output format contains an additional four `<asterisk>` characters after the range of affected lines in the first filename. This was to provide a flag for old programs (like old versions of *patch*) that only understand the old context format. The version of context described here does not require that multiple changes within context lines be merged, but it does not prohibit it either. The extension is upwards-compatible, so any vendors that wish to retain the old version of *diff* can do so by adding the extra four `<asterisk>` characters (that is, utilities that currently use *diff* and understand the new merged format will also understand the old unmerged format, but not *vice versa*).

The `-u` and `-U` options of GNU *diff* have been included. Their output format, designed by Wayne Davison, takes up less space than `-c` and `-C` format, and in many cases is easier to read. The format's timestamps do not vary by locale, so *LC\_TIME* does not affect it. The format's line numbers are rendered with the `%ld` format, not `%d`, because the file format notation rules would allow extra `<blank>` characters to appear around the numbers.

The substitute command was added as an additional format for the `-e` option. This was added to provide implementations with a way to fix the classic "dot alone on a line" bug present in many versions of *diff*. Since many implementations have fixed this bug, the standard developers decided not to standardize broken behavior, but rather to provide the necessary tool for fixing the bug. One way to fix this bug is to output two periods whenever a lone period is needed, then terminate the append command with a period, and then use the substitute command to convert the two periods into one period.

The BSD-derived `-r` option was added to provide a mechanism for using *diff* to compare two file system trees. This behavior is useful, is standard practice on all BSD-derived systems, and is not easily reproducible with the *find* utility.

The requirement that *diff* not compare files in some circumstances, even though they have the same name, is based on the actual output of historical implementations. The specified behavior precludes the problems arising from running into FIFOs and other files that would cause *diff* to hang waiting for input with no indication to the user that *diff* was hung. An earlier version of this standard specified the output format more precisely, but in practice this requirement was widely ignored and the benefit of standardization seemed small, so it is now unspecified. In most common usage, *diff -r* should indicate differences in the file hierarchies, not the difference of contents of devices pointed to by the hierarchies.

Many early implementations of *diff* require seekable files. Since the System Interfaces volume of POSIX.1-2008 supports named pipes, the standard developers decided that such a restriction was unreasonable. Note also that the allowed filename – almost always refers to a pipe.

No directory search order is specified for *diff*. The historical ordering is, in fact, not optimal, in that it prints out all of the differences at the current level, including the statements about all common subdirectories before recursing into those subdirectories.

The message:

```
"diff %s %s %s\n", <diff_options>, <filename1>, <filename2>
```

does not vary by locale because it is the representation of a command, not an English sentence.

**86476 FUTURE DIRECTIONS**

86477 None.

**86478 SEE ALSO**

86479 *cmp, comm, ed, find*

86480 XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

**86481 CHANGE HISTORY**

86482 First released in Issue 2.

**86483 Issue 5**

86484 The FUTURE DIRECTIONS section is added.

**86485 Issue 6**

86486 The following new requirements on POSIX implementations derive from alignment with the  
86487 Single UNIX Specification:

- 86488 • The `-f` option is added.

86489 The output format for `-c` or `-C` format is changed to align with changes to the IEEE P1003.2b  
86490 draft standard resulting from IEEE PASC Interpretation 1003.2 #71.

86491 The normative text is reworded to avoid use of the term “must” for application requirements.

86492 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/20 is applied, changing the STDOUT  
86493 section. This changes the specification of *diff* `-c` so that it agrees with existing practice when  
86494 contexts contain zero lines or one line.

**86495 Issue 7**

86496 Austin Group Interpretations 1003.1-2001 #115 and #114 are applied.

86497 Austin Group Interpretation 1003.1-2001 #192 is applied, clarifying the behavior if both files are  
86498 non-text files.

86499 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

86500 SD5-XCU-ERN-103 and SD5-XCU-ERN-120 are applied, adding the `-u` option.

86501 **NAME**

86502       dirname — return the directory portion of a pathname

86503 **SYNOPSIS**86504       dirname *string*86505 **DESCRIPTION**

86506       The *string* operand shall be treated as a pathname, as defined in XBD [Section 3.271](#) (on page 76).  
 86507       The string *string* shall be converted to the name of the directory containing the filename  
 86508       corresponding to the last pathname component in *string*, performing actions equivalent to the  
 86509       following steps in order:

- 86510           1. If *string* is *//*, skip steps 2 to 5.
- 86511           2. If *string* consists entirely of <slash> characters, *string* shall be set to a single <slash>  
 86512           character. In this case, skip steps 3 to 8.
- 86513           3. If there are any trailing <slash> characters in *string*, they shall be removed.
- 86514           4. If there are no <slash> characters remaining in *string*, *string* shall be set to a single  
 86515           <period> character. In this case, skip steps 5 to 8.
- 86516           5. If there are any trailing non-<slash> characters in *string*, they shall be removed.
- 86517           6. If the remaining *string* is *//*, it is implementation-defined whether steps 7 and 8 are  
 86518           skipped or processed.
- 86519           7. If there are any trailing <slash> characters in *string*, they shall be removed.
- 86520           8. If the remaining *string* is empty, *string* shall be set to a single <slash> character.

86521       The resulting string shall be written to standard output.

86522 **OPTIONS**

86523       None.

86524 **OPERANDS**

86525       The following operand shall be supported:

86526       *string*       A string.86527 **STDIN**

86528       Not used.

86529 **INPUT FILES**

86530       None.

86531 **ENVIRONMENT VARIABLES**86532       The following environment variables shall affect the execution of *dirname*:

86533       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 86534       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 86535       variables used to determine the values of locale categories.)

86536       *LC\_ALL*     If set to a non-empty string value, override the values of all the other  
 86537       internationalization variables.

86538       *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 86539       characters (for example, single-byte as opposed to multi-byte characters in  
 86540       arguments).

86541 **LC\_MESSAGES**

86542 Determine the locale that should be used to affect the format and contents of  
 86543 diagnostic messages written to standard error.

86544 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

86545 **ASYNCHRONOUS EVENTS**

86546 Default.

86547 **STDOUT**

86548 The *dirname* utility shall write a line to the standard output in the following format:

86549 "%s\n", *<resulting string>*

86550 **STDERR**

86551 The standard error shall be used only for diagnostic messages.

86552 **OUTPUT FILES**

86553 None.

86554 **EXTENDED DESCRIPTION**

86555 None.

86556 **EXIT STATUS**

86557 The following exit values shall be returned:

86558 0 Successful completion.

86559 >0 An error occurred.

86560 **CONSEQUENCES OF ERRORS**

86561 Default.

86562 **APPLICATION USAGE**

86563 The definition of *pathname* specifies implementation-defined behavior for pathnames starting  
 86564 with two <slash> characters. Therefore, applications shall not arbitrarily add <slash> characters  
 86565 to the beginning of a pathname unless they can ensure that there are more or less than two or are  
 86566 prepared to deal with the implementation-defined consequences.

86567 **EXAMPLES**

86568 The EXAMPLES section of the *basename()* function (see XSH *basename()*) includes a table |  
 86569 showing examples of the results of processing several sample pathnames by the *basename()* and |  
 86570 *dirname()* functions and by the *basename* and *dirname* utilities.

86571 See also the examples for the *basename* utility.

86572 **RATIONALE**

86573 The behaviors of *basename* and *dirname* in this volume of POSIX.1-2008 have been coordinated so -  
 86574 that when *string* is a valid pathname:

86575 `$(basename -- "string")`

86576 would be a valid filename for the file in the directory:

86577 `$(dirname -- "string")`

86578 This would not work for the versions of these utilities in early proposals due to the way  
 86579 processing of trailing <slash> characters was specified. Consideration was given to leaving  
 86580 processing unspecified if there were trailing <slash> characters, but this cannot be done; XBD  
 86581 Section 3.271 (on page 76) allows trailing <slash> characters. The *basename* and *dirname* utilities  
 86582 have to specify consistent handling for all valid pathnames.

86583 **FUTURE DIRECTIONS**

86584 None.

86585 **SEE ALSO**86586 [Section 2.5](#) (on page 2339), *basename*86587 [XBD Section 3.271](#) (on page 76), [Chapter 8](#) (on page 173)86588 XSH *basename()*, *dirname()* +86589 **CHANGE HISTORY**

86590 First released in Issue 2.

86591 **Issue 7**86592 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0083 [192,430], XCU/TC1-2008/0084  
86593 [192], and XCU/TC1-2008/0085 [192] are applied.

**NAME**

**du** — estimate file space usage

**SYNOPSIS**

**du** [**-a** | **-s**] [**-kx**] [**-H** | **-L**] [*file*...]

**DESCRIPTION**

By default, the *du* utility shall write to standard output the size of the file space allocated to, and the size of the file space allocated to each subdirectory of, the file hierarchy rooted in each of the specified files. By default, when a symbolic link is encountered on the command line or in the file hierarchy, *du* shall count the size of the symbolic link (rather than the file referenced by the link), and shall not follow the link to another portion of the file hierarchy. The size of the file space allocated to a file of type directory shall be defined as the sum total of space allocated to all files in the file hierarchy rooted in the directory plus the space allocated to the directory itself.

When *du* cannot *stat()* files or *stat()* or read directories, it shall report an error condition and the final exit status is affected. A file that occurs multiple times under one file operand and that has a link count greater than 1 shall be counted and written for only one entry. It is implementation-defined whether a file that has a link count no greater than 1 is counted and written just once, or is counted and written for each occurrence. It is implementation-defined whether a file that occurs under one file operand is counted for other file operands. The directory entry that is selected in the report is unspecified. By default, file sizes shall be written in 512-byte units, rounded up to the next 512-byte unit.

**OPTIONS**

The *du* utility shall conform to XBD [Section 12.2](#) (on page 216).

The following options shall be supported:

**-a** In addition to the default output, report the size of each file not of type directory in the file hierarchy rooted in the specified file. The **-a** option shall not affect whether non-directories given as *file* operands are listed.

**-H** If a symbolic link is specified on the command line, *du* shall count the size of the file or file hierarchy referenced by the link.

**-k** Write the files sizes in units of 1 024 bytes, rather than the default 512-byte units.

**-L** If a symbolic link is specified on the command line or encountered during the traversal of a file hierarchy, *du* shall count the size of the file or file hierarchy referenced by the link.

**-s** Instead of the default output, report only the total sum for each of the specified files.

**-x** When evaluating file sizes, evaluate only those files that have the same device as the file specified by the *file* operand.

Specifying more than one of the mutually-exclusive options **-H** and **-L** shall not be considered an error. The last option specified shall determine the behavior of the utility.

**OPERANDS**

The following operand shall be supported:

*file* The pathname of a file whose size is to be written. If no *file* is specified, the current directory shall be used.

86636 **STDIN**

86637 Not used.

86638 **INPUT FILES**

86639 None.

86640 **ENVIRONMENT VARIABLES**86641 The following environment variables shall affect the execution of *du*:

86642 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 86643 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 86644 variables used to determine the values of locale categories.)

86645 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 86646 internationalization variables.

86647 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 86648 characters (for example, single-byte as opposed to multi-byte characters in  
 86649 arguments).

86650 *LC\_MESSAGES*

86651 Determine the locale that should be used to affect the format and contents of  
 86652 diagnostic messages written to standard error.

86653 *XSI* *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

86654 **ASYNCHRONOUS EVENTS**

86655 Default.

86656 **STDOUT**

86657 The output from *du* shall consist of the amount of space allocated to a file and the name of the  
 86658 file, in the following format:

86659 "%d %s\n", *<size>*, *<pathname>*86660 **STDERR**

86661 The standard error shall be used only for diagnostic messages.

86662 **OUTPUT FILES**

86663 None.

86664 **EXTENDED DESCRIPTION**

86665 None.

86666 **EXIT STATUS**

86667 The following exit values shall be returned:

86668 0 Successful completion.

86669 &gt;0 An error occurred.

86670 **CONSEQUENCES OF ERRORS**

86671 Default.



**APPLICATION USAGE**

None.

**EXAMPLES**

None.

**RATIONALE**

The use of 512-byte units is historical practice and maintains compatibility with *ls* and other utilities in this volume of POSIX.1-2008. This does not mandate that the file system itself be based on 512-byte blocks. The **-k** option was added as a compromise measure. It was agreed by the standard developers that 512 bytes was the best default unit because of its complete historical consistency on System V (*versus* the mixed 512/1024-byte usage on BSD systems), and that a **-k** option to switch to 1024-byte units was a good compromise. Users who prefer the 1024-byte quantity can easily alias *du* to *du -k* without breaking the many historical scripts relying on the 512-byte units.

The **-b** option was added to an early proposal to provide a resolution to the situation where System V and BSD systems give figures for file sizes in *blocks*, which is an implementation-defined concept. (In common usage, the block size is 512 bytes for System V and 1024 bytes for BSD systems.) However, **-b** was later deleted, since the default was eventually decided as 512-byte units.

Historical file systems provided no way to obtain exact figures for the space allocation given to files. There are two known areas of inaccuracies in historical file systems: cases of *indirect blocks* being used by the file system or *sparse* files yielding incorrectly high values. An indirect block is space used by the file system in the storage of the file, but that need not be counted in the space allocated to the file. A *sparse* file is one in which an *lseek()* call has been made to a position beyond the end of the file and data has subsequently been written at that point. A file system need not allocate all the intervening zero-filled blocks to such a file. It is up to the implementation to define exactly how accurate its methods are.

The **-a** and **-s** options were mutually-exclusive in the original version of *du*. The POSIX Shell and Utilities description is implied by the language in the SVID where **-s** is described as causing “only the grand total” to be reported. Some systems may produce output for **-sa**, but a Strictly Conforming POSIX Shell and Utilities Application cannot use that combination.

The **-a** and **-s** options were adopted from the SVID except that the System V behavior of not listing non-directories explicitly given as operands, unless the **-a** option is specified, was considered a bug; the BSD-based behavior (report for all operands) is mandated. The default behavior of *du* in the SVID with regard to reporting the failure to read files (it produces no messages) was considered counter-intuitive, and thus it was specified that the POSIX Shell and Utilities default behavior shall be to produce such messages. These messages can be turned off with shell redirection to achieve the System V behavior.

The **-x** option is historical practice on recent BSD systems. It has been adopted by this volume of POSIX.1-2008 because there was no other historical method of limiting the *du* search to a single file hierarchy. This limitation of the search is necessary to make it possible to obtain file space usage information about a file system on which other file systems are mounted, without having to resort to a lengthy *find* and *awk* script.

**FUTURE DIRECTIONS**

A future version of this standard may require that a file that occurs multiple times shall be counted and written for only one entry, even if the occurrences are under different file operands.

86717 **SEE ALSO**86718 *ls*86719 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)86720 XSH *fstatat()*86721 **CHANGE HISTORY**

86722 First released in Issue 2.

86723 **Issue 6**

86724 This utility is marked as part of the User Portability Utilities option.

86725 The APPLICATION USAGE section is added.

86726 The obsolescent **-r** option is removed.86727 The Open Group Corrigendum U025/3 is applied. The *du* utility is reinstated, as it had  
86728 incorrectly been marked LEGACY in Issue 5.86729 The **-H** and **-L** options for symbolic links are added as described in the IEEE P1003.2b draft  
86730 standard.86731 **Issue 7**86732 The *du* utility is moved from the User Portability Utilities option to the Base. User Portability  
86733 Utilities is now an option for interactive utilities.

86734 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

86735 **NAME**

86736 echo — write arguments to standard output

86737 **SYNOPSIS**86738 echo [*string*...]86739 **DESCRIPTION**

86740 The *echo* utility writes its arguments to standard output, followed by a <newline>. If there are  
 86741 no arguments, only the <newline> is written.

86742 **OPTIONS**

86743 The *echo* utility shall not recognize the "--" argument in the manner specified by Guideline 10  
 86744 of XBD [Section 12.2](#) (on page 216); "--" shall be recognized as a string operand.

86745 Implementations shall not support any options.

86746 **OPERANDS**

86747 The following operands shall be supported:

86748 *string* A string to be written to standard output. If the first operand is **-n**, or if any of the  
 86749 operands contain a <backslash> character, the results are implementation-defined.

86750 XSI On XSI-conformant systems, if the first operand is **-n**, it shall be treated as a string,  
 86751 not an option. The following character sequences shall be recognized on XSI-  
 86752 conformant systems within any of the arguments:

|       |              |                                                                                                                                                                                 |
|-------|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 86753 | <b>\a</b>    | Write an <alert>.                                                                                                                                                               |
| 86754 | <b>\b</b>    | Write a <backspace>.                                                                                                                                                            |
| 86755 | <b>\c</b>    | Suppress the <newline> that otherwise follows the final argument in the<br>86756 output. All characters following the ' <b>\c</b> ' in the arguments shall be<br>86757 ignored. |
| 86758 | <b>\f</b>    | Write a <form-feed>.                                                                                                                                                            |
| 86759 | <b>\n</b>    | Write a <newline>.                                                                                                                                                              |
| 86760 | <b>\r</b>    | Write a <carriage-return>.                                                                                                                                                      |
| 86761 | <b>\t</b>    | Write a <tab>.                                                                                                                                                                  |
| 86762 | <b>\v</b>    | Write a <vertical-tab>.                                                                                                                                                         |
| 86763 | <b>\\</b>    | Write a <backslash> character.                                                                                                                                                  |
| 86764 | <b>\0num</b> | Write an 8-bit value that is the zero, one, two, or three-digit octal number<br>86765 <i>num</i> .                                                                              |

86766 **STDIN**

86767 Not used.

86768 **INPUT FILES**

86769 None.

86770 **ENVIRONMENT VARIABLES**86771 The following environment variables shall affect the execution of *echo*:

86772 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 86773 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 86774 variables used to determine the values of locale categories.)

86775 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 86776 internationalization variables.

86777 XSI *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 86778 characters (for example, single-byte as opposed to multi-byte characters in  
 86779 arguments).

86780 *LC\_MESSAGES*  
 86781 Determine the locale that should be used to affect the format and contents of  
 86782 diagnostic messages written to standard error.

86783 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

86784 **ASYNCHRONOUS EVENTS**  
 86785 Default.

86786 **STDOUT**  
 86787 The *echo* utility arguments shall be separated by single <space> characters and a <newline>  
 86788 XSI character shall follow the last argument. Output transformations shall occur based on the  
 86789 escape sequences in the input. See the OPERANDS section.

86790 **STDERR**  
 86791 The standard error shall be used only for diagnostic messages.

86792 **OUTPUT FILES**  
 86793 None.

86794 **EXTENDED DESCRIPTION**  
 86795 None.

86796 **EXIT STATUS**  
 86797 The following exit values shall be returned:  
 86798 0 Successful completion.  
 86799 >0 An error occurred.

86800 **CONSEQUENCES OF ERRORS**  
 86801 Default.

86802 **APPLICATION USAGE**  
 86803 It is not possible to use *echo* portably across all POSIX systems unless both *-n* (as the first  
 86804 argument) and escape sequences are omitted.

86805 The *printf* utility can be used portably to emulate any of the traditional behaviors of the *echo*  
 86806 utility as follows (assuming that *IFS* has its standard value or is unset):

- 86807 • The historic System V *echo* and the requirements on XSI implementations in this volume of  
 86808 POSIX.1-2008 are equivalent to:

```
86809 printf "%b\n" "$*"
86810
```

- 86810 • The BSD *echo* is equivalent to:

```
86811 if [ "X$1" = "X-n" ]
86812 then
86813     shift
86814     printf "%s" "$*"
86815 else
86816     printf "%s\n" "$*"
86817 fi
```

86818 New applications are encouraged to use *printf* instead of *echo*.

#### 86819 EXAMPLES

86820 None.

#### 86821 RATIONALE

86822 The *echo* utility has not been made obsolescent because of its extremely widespread use in  
86823 historical applications. Conforming applications that wish to do prompting without <newline>  
86824 characters or that could possibly be expecting to echo a **-n**, should use the *printf* utility derived  
86825 from the Ninth Edition system.

86826 As specified, *echo* writes its arguments in the simplest of ways. The two different historical  
86827 versions of *echo* vary in fatally incompatible ways.

86828 The BSD *echo* checks the first argument for the string **-n** which causes it to suppress the  
86829 <newline> that would otherwise follow the final argument in the output.

86830 The System V *echo* does not support any options, but allows escape sequences within its  
86831 operands, as described for XSI implementations in the OPERANDS section.

86832 The *echo* utility does not support Utility Syntax Guideline 10 because historical applications  
86833 depend on *echo* to echo *all* of its arguments, except for the **-n** option in the BSD version.

#### 86834 FUTURE DIRECTIONS

86835 None.

#### 86836 SEE ALSO

86837 *printf*

86838 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

#### 86839 CHANGE HISTORY

86840 First released in Issue 2.

##### 86841 Issue 5

86842 In the OPTIONS section, the last sentence is changed to indicate that implementations “do not”  
86843 support any options; in the previous issue this said “need not”.

##### 86844 Issue 6

86845 The following new requirements on POSIX implementations derive from alignment with the  
86846 Single UNIX Specification:

- 86847 • A set of character sequences is defined as *string* operands.
- 86848 • *LC\_CTYPE* is added to the list of environment variables affecting *echo*.
- 86849 • In the OPTIONS section, implementations shall not support any options.

86850 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/21 is applied, so that the *echo* utility can  
86851 accommodate historical BSD behavior.

##### 86852 Issue 7

86853 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

86854 **NAME**

86855 ed — edit text

86856 **SYNOPSIS**86857 ed [-p *string*] [-s] [*file*]86858 **DESCRIPTION**

86859 The *ed* utility is a line-oriented text editor that uses two modes: *command mode* and *input mode*. In  
 86860 command mode the input characters shall be interpreted as commands, and in input mode they  
 86861 shall be interpreted as text. See the EXTENDED DESCRIPTION section.

86862 If an operand is '-', the results are unspecified.

86863 **OPTIONS**

86864 The *ed* utility shall conform to XBD Section 12.2 (on page 216), except for the unspecified usage  
 86865 of '-'.

86866 The following options shall be supported:

86867 -p *string* Use *string* as the prompt string when in command mode. By default, there shall be  
 86868 no prompt string.

86869 -s Suppress the writing of byte counts by **e**, **E**, **r**, and **w** commands and of the '!'  
 86870 prompt after a *!command*.

86871 **OPERANDS**

86872 The following operand shall be supported:

86873 *file* If the *file* argument is given, *ed* shall simulate an **e** command on the file named by  
 86874 the pathname, *file*, before accepting commands from the standard input.

86875 **STDIN**

86876 The standard input shall be a text file consisting of commands, as described in the EXTENDED  
 86877 DESCRIPTION section.

86878 **INPUT FILES**

86879 The input files shall be text files.

86880 **ENVIRONMENT VARIABLES**

86881 The following environment variables shall affect the execution of *ed*:

86882 *HOME* Determine the pathname of the user's home directory.

86883 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 86884 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 86885 variables used to determine the values of locale categories.)

86886 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 86887 internationalization variables.

86888 *LC\_COLLATE*

86889 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 86890 character collating elements within regular expressions.

86891 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 86892 characters (for example, single-byte as opposed to multi-byte characters in  
 86893 arguments and input files) and the behavior of character classes within regular  
 86894 expressions.

86895 **LC\_MESSAGES**

86896 Determine the locale that should be used to affect the format and contents of  
 86897 diagnostic messages written to standard error and informative messages written to  
 86898 standard output.

86899 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

86900 **ASYNCHRONOUS EVENTS**

86901 The *ed* utility shall take the standard action for all signals (see the ASYNCHRONOUS EVENTS  
 86902 section in [Section 1.4](#), on page 2326) with the following exceptions:

86903 **SIGINT** The *ed* utility shall interrupt its current activity, write the string "?\n" to standard  
 86904 output, and return to command mode (see the EXTENDED DESCRIPTION  
 86905 section).

86906 **SIGHUP** If the buffer is not empty and has changed since the last write, the *ed* utility shall  
 86907 attempt to write a copy of the buffer in a file. First, the file named **ed.hup** in the  
 86908 current directory shall be used; if that fails, the file named **ed.hup** in the directory  
 86909 named by the *HOME* environment variable shall be used. In any case, the *ed* utility  
 86910 shall exit without writing the file to the currently remembered pathname and  
 86911 without returning to command mode.

86912 **SIGQUIT** The *ed* utility shall ignore this event.

86913 **STDOUT**

86914 Various editing commands and the prompting feature (see **-p**) write to standard output, as  
 86915 described in the EXTENDED DESCRIPTION section.

86916 **STDERR**

86917 The standard error shall be used only for diagnostic messages.

86918 **OUTPUT FILES**

86919 The output files shall be text files whose formats are dependent on the editing commands given.

86920 **EXTENDED DESCRIPTION**

86921 The *ed* utility shall operate on a copy of the file it is editing; changes made to the copy shall have  
 86922 no effect on the file until a **w** (write) command is given. The copy of the text is called the *buffer*.

86923 Commands to *ed* have a simple and regular structure: zero, one, or two *addresses* followed by a  
 86924 single-character *command*, possibly followed by parameters to that command. These addresses  
 86925 specify one or more lines in the buffer. Every command that requires addresses has default  
 86926 addresses, so that the addresses very often can be omitted. If the **-p** option is specified, the  
 86927 prompt string shall be written to standard output before each command is read.

86928 In general, only one command can appear on a line. Certain commands allow text to be input.  
 86929 This text is placed in the appropriate place in the buffer. While *ed* is accepting text, it is said to be  
 86930 in *input mode*. In this mode, no commands shall be recognized; all input is merely collected.  
 86931 Input mode is terminated by entering a line consisting of two characters: a <period> ( ' . ' )  
 86932 followed by a <newline>. This line is not considered part of the input text.



## Regular Expressions in ed

The *ed* utility shall support basic regular expressions, as described in XBD [Section 9.3](#) (on page 183). Since regular expressions in *ed* are always matched against single lines (excluding the terminating <newline> characters), never against any larger section of text, there is no way for a regular expression to match a <newline>.

A null RE shall be equivalent to the last RE encountered.

Regular expressions are used in addresses to specify lines, and in some commands (for example, the **s** substitute command) to specify portions of a line to be substituted.

## Addresses in ed

Addressing in *ed* relates to the current line. Generally, the current line is the last line affected by a command. The current line number is the address of the current line. If the edit buffer is not empty, the initial value for the current line shall be the last line in the edit buffer; otherwise, zero.

Addresses shall be constructed as follows:

1. The <period> character ( ' . ' ) shall address the current line.
2. The <dollar-sign> character ( ' \$ ' ) shall address the last line of the edit buffer.
3. The positive decimal number *n* shall address the *n*th line of the edit buffer.
4. The <apostrophe>-*x* character pair ( " ' *x* " ) shall address the line marked with the mark name character *x*, which shall be a lowercase letter from the portable character set. It shall be an error if the character has not been set to mark a line or if the line that was marked is not currently present in the edit buffer.
5. A BRE enclosed by <slash> characters ( ' / ' ) shall address the first line found by searching forwards from the line following the current line toward the end of the edit buffer and stopping at the first line for which the line excluding the terminating <newline> matches the BRE. The BRE consisting of a null BRE delimited by a pair of <slash> characters shall address the next line for which the line excluding the terminating <newline> matches the last BRE encountered. In addition, the second <slash> can be omitted at the end of a command line. Within the BRE, a <backslash>-<slash> pair ( " \ / " ) shall represent a literal <slash> instead of the BRE delimiter. If necessary, the search shall wrap around to the beginning of the buffer and continue up to and including the current line, so that the entire buffer is searched.
6. A BRE enclosed by <question-mark> characters ( ' ? ' ) shall address the first line found by searching backwards from the line preceding the current line toward the beginning of the edit buffer and stopping at the first line for which the line excluding the terminating <newline> matches the BRE. The BRE consisting of a null BRE delimited by a pair of <question-mark> characters ( " ? ? " ) shall address the previous line for which the line excluding the terminating <newline> matches the last BRE encountered. In addition, the second <question-mark> can be omitted at the end of a command line. Within the BRE, a <backslash>-<question-mark> pair ( " \ ? " ) shall represent a literal <question-mark> instead of the BRE delimiter. If necessary, the search shall wrap around to the end of the buffer and continue up to and including the current line, so that the entire buffer is searched.
7. A <plus-sign> ( ' + ' ) or <hyphen-minus> character ( ' - ' ) followed by a decimal number shall address the current line plus or minus the number. A <plus-sign> or <hyphen-minus> character not followed by a decimal number shall address the current line plus or minus 1.



Addresses can be followed by zero or more address offsets, optionally <blank>-separated. Address offsets are constructed as follows:

- A <plus-sign> or <hyphen-minus> character followed by a decimal number shall add or subtract, respectively, the indicated number of lines to or from the address. A <plus-sign> or <hyphen-minus> character not followed by a decimal number shall add or subtract 1 to or from the address.
- A decimal number shall add the indicated number of lines to the address.

It shall not be an error for an intermediate address value to be less than zero or greater than the last line in the edit buffer. It shall be an error for the final address value to be less than zero or greater than the last line in the edit buffer. It shall be an error if a search for a BRE fails to find a matching line.

Commands accept zero, one, or two addresses. If more than the required number of addresses are provided to a command that requires zero addresses, it shall be an error. Otherwise, if more than the required number of addresses are provided to a command, the addresses specified first shall be evaluated and then discarded until the maximum number of valid addresses remain, for the specified command.

Addresses shall be separated from each other by a <comma> (',' ) or <semicolon> character (';'). In the case of a <semicolon> separator, the current line ('.') shall be set to the first address, and only then will the second address be calculated. This feature can be used to determine the starting line for forwards and backwards searches; see rules 5. and 6.

Addresses can be omitted on either side of the <comma> or <semicolon> separator, in which case the resulting address pairs shall be as follows:

| Specified | Resulting   |
|-----------|-------------|
| ,         | 1 , \$      |
| , addr    | 1 , addr    |
| addr ,    | addr , addr |
| ;         | . ; \$      |
| ; addr    | . ; addr    |
| addr ;    | addr ; addr |

Any <blank> characters included between addresses, address separators, or address offsets shall be ignored.

### Commands in ed

In the following list of *ed* commands, the default addresses are shown in parentheses. The number of addresses shown in the default shall be the number expected by the command. The parentheses are not part of the address; they show that the given addresses are the default.

It is generally invalid for more than one command to appear on a line. However, any command (except **e**, **E**, **f**, **q**, **Q**, **r**, **w**, and **!**) can be suffixed by the letter **l**, **n**, or **p**; in which case, except for the **l**, **n**, and **p** commands, the command shall be executed and then the new current line shall be written as described below under the **l**, **n**, and **p** commands. When an **l**, **n**, or **p** suffix is used with an **l**, **n**, or **p** command, the command shall write to standard output as described below, but it is unspecified whether the suffix writes the current line again in the requested format or whether the suffix has no effect. For example, the **pl** command (base **p** command with an **l** suffix) shall either write just the current line or write it twice—once as specified for **p** and once as specified for **l**. Also, the **g**, **G**, **v**, and **V** commands shall take a command as a parameter.

Each address component can be preceded by zero or more <blank> characters. The command

letter can be preceded by zero or more <blank> characters. If a suffix letter (**l**, **n**, or **p**) is given, the application shall ensure that it immediately follows the command.

The **e**, **E**, **f**, **r**, and **w** commands shall take an optional *file* parameter, separated from the command letter by one or more <blank> characters.

If changes have been made in the buffer since the last **w** command that wrote the entire buffer, *ed* shall warn the user if an attempt is made to destroy the editor buffer via the **e** or **q** commands. The *ed* utility shall write the string:

" ?\n"

(followed by an explanatory message if *help mode* has been enabled via the **H** command) to standard output and shall continue in command mode with the current line number unchanged. If the **e** or **q** command is repeated with no intervening command, it shall take effect.

If a terminal disconnect (see XBD Chapter 11 (on page 199), Modem Disconnect and Closing a Device Terminal), is detected:

- If accompanied by a SIGHUP signal, the *ed* utility shall operate as described in the ASYNCHRONOUS EVENTS section for a SIGHUP signal.
- If not accompanied by a SIGHUP signal, the *ed* utility shall act as if an end-of-file had been detected on standard input.

If an end-of-file is detected on standard input:

- If the *ed* utility is in input mode, *ed* shall terminate input mode and return to command mode. It is unspecified if any partially entered lines (that is, input text without a terminating <newline>) are discarded from the input text.
- If the *ed* utility is in command mode, it shall act as if a **q** command had been entered.

If the closing delimiter of an RE or of a replacement string (for example, ' / ') in a **g**, **G**, **s**, **v**, or **V** command would be the last character before a <newline>, that delimiter can be omitted, in which case the addressed line shall be written. For example, the following pairs of commands are equivalent:

```
s/s1/s2      s/s1/s2/p
g/s1         g/s1/p
?s1         ?s1?
```

If an invalid command is entered, *ed* shall write the string:

" ?\n"

(followed by an explanatory message if *help mode* has been enabled via the **H** command) to standard output and shall continue in command mode with the current line number unchanged.

## Append Command

*Synopsis:*     ( . ) a  
                 <text>  
                 .

The **a** command shall read the given text and append it after the addressed line; the current line number shall become the address of the last inserted line or, if there were none, the addressed line. Address 0 shall be valid for this command; it shall cause the appended text to be placed at the beginning of the buffer.

**Change Command**

*Synopsis:*     ( . , . ) c  
                  <text>  
                  .

The **c** command shall delete the addressed lines, then accept input text that replaces these lines; the current line shall be set to the address of the last line input; or, if there were none, at the line after the last line deleted; if the lines deleted were originally at the end of the buffer, the current line number shall be set to the address of the new last line; if no lines remain in the buffer, the current line number shall be set to zero. Address 0 shall be valid for this command; it shall be interpreted as if address 1 were specified.

**Delete Command**

*Synopsis:*     ( . , . ) d

The **d** command shall delete the addressed lines from the buffer. The address of the line after the last line deleted shall become the current line number; if the lines deleted were originally at the end of the buffer, the current line number shall be set to the address of the new last line; if no lines remain in the buffer, the current line number shall be set to zero.

**Edit Command**

*Synopsis:*     e [*file*]

The **e** command shall delete the entire contents of the buffer and then read in the file named by the pathname *file*. The current line number shall be set to the address of the last line of the buffer. If no pathname is given, the currently remembered pathname, if any, shall be used (see the **f** command). The number of bytes read shall be written to standard output, unless the **-s** option was specified, in the following format:

"%d\n", <number of bytes read>

The name *file* shall be remembered for possible use as a default pathname in subsequent **e**, **E**, **r**, and **w** commands. If *file* is replaced by '!', the rest of the line shall be taken to be a shell command line whose output is to be read. Such a shell command line shall not be remembered as the current *file*. All marks shall be discarded upon the completion of a successful **e** command. If the buffer has changed since the last time the entire buffer was written, the user shall be warned, as described previously.

**Edit Without Checking Command**

*Synopsis:*     E [*file*]

The **E** command shall possess all properties and restrictions of the **e** command except that the editor shall not check to see whether any changes have been made to the buffer since the last **w** command.

**Filename Command**

*Synopsis:*     **f** [*file*]

If *file* is given, the **f** command shall change the currently remembered pathname to *file*; whether the name is changed or not, it shall then write the (possibly new) currently remembered pathname to the standard output in the following format:

"%s\n", <pathname>

The current line number shall be unchanged.

**Global Command**

*Synopsis:*     (1,\$)g/RE/command list

In the **g** command, the first step shall be to mark every line for which the line excluding the terminating <newline> matches the given RE. Then, going sequentially from the beginning of the file to the end of the file, the given *command list* shall be executed for each marked line, with the current line number set to the address of that line. Any line modified by the *command list* shall be unmarked. When the **g** command completes, the current line number shall have the value assigned by the last command in the *command list*. If there were no matching lines, the current line number shall not be changed. A single command or the first of a list of commands shall appear on the same line as the global command. All lines of a multi-line list except the last line shall be ended with a <backslash> preceding the terminating <newline>; the **a**, **i**, and **c** commands and associated input are permitted. The ' . ' terminating input mode can be omitted if it would be the last line of the *command list*. An empty *command list* shall be equivalent to the **p** command. The use of the **g**, **G**, **v**, **V**, and **!** commands in the *command list* produces undefined results. Any character other than <space> or <newline> can be used instead of a <slash> to delimit the RE. Within the RE, the RE delimiter itself can be used as a literal character if it is preceded by a <backslash>.

**Interactive Global Command**

*Synopsis:*     (1,\$)G/RE/

In the **G** command, the first step shall be to mark every line for which the line excluding the terminating <newline> matches the given RE. Then, for every such line, that line shall be written, the current line number shall be set to the address of that line, and any one command (other than one of the **a**, **c**, **i**, **g**, **G**, **v**, and **V** commands) shall be read and executed. A <newline> shall act as a null command (causing no action to be taken on the current line); an '&' shall cause the re-execution of the most recent non-null command executed within the current invocation of **G**. Note that the commands input as part of the execution of the **G** command can address and affect any lines in the buffer. Any line modified by the command shall be unmarked. The final value of the current line number shall be the value set by the last command successfully executed. (Note that the last command successfully executed shall be the **G** command itself if a command fails or the null command is specified.) If there were no matching lines, the current line number shall not be changed. The **G** command can be terminated by a SIGINT signal. Any character other than <space> or <newline> can be used instead of a <slash> to delimit the RE and the replacement. Within the RE, the RE delimiter itself can be used as a literal character if it is preceded by a <backslash>.

**Help Command**

*Synopsis:*     h

The **h** command shall write a short message to standard output that explains the reason for the most recent '?' notification. The current line number shall be unchanged.

**Help-Mode Command**

*Synopsis:*     H

The **H** command shall cause *ed* to enter a mode in which help messages (see the **h** command) shall be written to standard output for all subsequent '?' notifications. The **H** command alternately shall turn this mode on and off; it is initially off. If the help-mode is being turned on, the **H** command also explains the previous '?' notification, if there was one. The current line number shall be unchanged.

**Insert Command**

*Synopsis:*     ( . ) i  
                  <text>  
                  .

The **i** command shall insert the given text before the addressed line; the current line is set to the last inserted line or, if there was none, to the addressed line. This command differs from the **a** command only in the placement of the input text. Address 0 shall be valid for this command; it shall be interpreted as if address 1 were specified.

**Join Command**

*Synopsis:*     ( . , . + 1 ) j

The **j** command shall join contiguous lines by removing the appropriate <newline> characters. If exactly one address is given, this command shall do nothing. If lines are joined, the current line number shall be set to the address of the joined line; otherwise, the current line number shall be unchanged.

**Mark Command**

*Synopsis:*     ( . ) kx

The **k** command shall mark the addressed line with name *x*, which the application shall ensure is a lowercase letter from the portable character set. The address " 'x' " shall then refer to this line; the current line number shall be unchanged.

**List Command**

*Synopsis:*     ( . , . ) l

The **l** command shall write to standard output the addressed lines in a visually unambiguous form. The characters listed in XBD Table 5-1 (on page 121) ('\\', '\a', '\b', '\f', '\r', '\t', '\v') shall be written as the corresponding escape sequence; the '\n' in that table is not applicable. Non-printable characters not in the table shall be written as one three-digit octal number (with a preceding <backslash> character) for each byte in the character (most significant byte first).

Long lines shall be folded, with the point of folding indicated by <newline> preceded by a <backslash>; the length at which folding occurs is unspecified, but should be appropriate for the

output device. The end of each line shall be marked with a '\$', and '\$' characters within the text shall be written with a preceding <backslash>. An l command can be appended to any other command other than e, E, f, q, Q, r, w, or !. The current line number shall be set to the address of the last line written.

### Move Command

*Synopsis:* ( , . ) *m* *address*

The **m** command shall reposition the addressed lines after the line addressed by *address*. Address 0 shall be valid for *address* and cause the addressed lines to be moved to the beginning of the buffer. It shall be an error if *address* falls within the range of moved lines. The current line number shall be set to the address of the last line moved.

### Number Command

*Synopsis:* ( , . ) *n*

The **n** command shall write to standard output the addressed lines, preceding each line by its line number and a <tab>; the current line number shall be set to the address of the last line written. The **n** command can be appended to any command other than e, E, f, q, Q, r, w, or !.

### Print Command

*Synopsis:* ( , . ) *p*

The **p** command shall write to standard output the addressed lines; the current line number shall be set to the address of the last line written. The **p** command can be appended to any command other than e, E, f, q, Q, r, w, or !.

### Prompt Command

*Synopsis:* *P*

The **P** command shall cause *ed* to prompt with an <asterisk> ('\*') (or *string*, if **-p** is specified) for all subsequent commands. The **P** command alternatively shall turn this mode on and off; it shall be initially on if the **-p** option is specified; otherwise, off. The current line number shall be unchanged.

### Quit Command

*Synopsis:* *q*

The **q** command shall cause *ed* to exit. If the buffer has changed since the last time the entire buffer was written, the user shall be warned, as described previously.

### Quit Without Checking Command

*Synopsis:* *Q*

The **Q** command shall cause *ed* to exit without checking whether changes have been made in the buffer since the last **w** command.

**Read Command**

*Synopsis:*     (\$)*r* [*file*]

The *r* command shall read in the file named by the pathname *file* and append it after the addressed line. If no *file* argument is given, the currently remembered pathname, if any, shall be used (see the *e* and *f* commands). The currently remembered pathname shall not be changed unless there is no remembered pathname. Address 0 shall be valid for *r* and shall cause the file to be read at the beginning of the buffer. If the read is successful, and *-s* was not specified, the number of bytes read shall be written to standard output in the following format:

"%d\n", <number of bytes read>

The current line number shall be set to the address of the last line read in. If *file* is replaced by '!', the rest of the line shall be taken to be a shell command line whose output is to be read. Such a shell command line shall not be remembered as the current pathname.

**Substitute Command**

*Synopsis:*     (. . .)*s*/RE/*replacement/flags*

The *s* command shall search each addressed line for an occurrence of the specified RE and replace either the first or all (non-overlapped) matched strings with the *replacement*; see the following description of the *g* suffix. It is an error if the substitution fails on every addressed line. Any character other than <space> or <newline> can be used instead of a <slash> to delimit the RE and the replacement. Within the RE, the RE delimiter itself can be used as a literal character if it is preceded by a <backslash>. The current line shall be set to the address of the last line on which a substitution occurred.

An <ampersand> ('&') appearing in the *replacement* shall be replaced by the string matching the RE on the current line. The special meaning of '&' in this context can be suppressed by preceding it by <backslash>. As a more general feature, the characters '\n', where *n* is a digit, shall be replaced by the text matched by the corresponding back-reference expression. If the corresponding back-reference expression does not match, then the characters '\n' shall be replaced by the empty string. When the character '%' is the only character in the *replacement*, the *replacement* used in the most recent substitute command shall be used as the *replacement* in the current substitute command; if there was no previous substitute command, the use of '%' in this manner shall be an error. The '%' shall lose its special meaning when it is in a replacement string of more than one character or is preceded by a <backslash>. For each <backslash> encountered in scanning *replacement* from beginning to end, the following character shall lose its special meaning (if any). It is unspecified what special meaning is given to any character other than <backslash>, '&', '%', or digits.

A line can be split by substituting a <newline> into it. The application shall ensure it escapes the <newline> in the *replacement* by preceding it by <backslash>. Such substitution cannot be done as part of a *g* or *v* command list. The current line number shall be set to the address of the last line on which a substitution is performed. If no substitution is performed, the current line number shall be unchanged. If a line is split, a substitution shall be considered to have been performed on each of the new lines for the purpose of determining the new current line number. A substitution shall be considered to have been performed even if the replacement string is identical to the string that it replaces.

The application shall ensure that the value of *flags* is zero or more of:

*count*     Substitute for the *count*th occurrence only of the RE found on each addressed line.



- 87258 **g** Globally substitute for all non-overlapping instances of the RE rather than just the first  
87259 one. If both **g** and *count* are specified, the results are unspecified.
- 87260 **l** Write to standard output the final line in which a substitution was made. The line shall  
87261 be written in the format specified for the **l** command.
- 87262 **n** Write to standard output the final line in which a substitution was made. The line shall  
87263 be written in the format specified for the **n** command.
- 87264 **p** Write to standard output the final line in which a substitution was made. The line shall  
87265 be written in the format specified for the **p** command.

## 87266 **Copy Command**

87267 *Synopsis:* ( . , . ) *taddress*

87268 The **t** command shall be equivalent to the **m** command, except that a copy of the addressed lines  
87269 shall be placed after address *address* (which can be 0); the current line number shall be set to the  
87270 address of the last line added.

## 87271 **Undo Command**

87272 *Synopsis:* **u**

87273 The **u** command shall nullify the effect of the most recent command that modified anything in  
87274 the buffer, namely the most recent **a**, **c**, **d**, **g**, **i**, **j**, **m**, **r**, **s**, **t**, **u**, **v**, **G**, or **V** command. All changes  
87275 made to the buffer by a **g**, **G**, **v**, or **V** global command shall be undone as a single change; if no  
87276 changes were made by the global command (such as with **g/RE/p**), the **u** command shall have  
87277 no effect. The current line number shall be set to the value it had immediately before the  
87278 command being undone started.

## 87279 **Global Non-Matched Command**

87280 *Synopsis:* ( 1 , \$ ) **v/RE/command list**

87281 This command shall be equivalent to the global command **g** except that the lines that are marked  
87282 during the first step shall be those for which the line excluding the terminating <newline> does  
87283 not match the RE.

## 87284 **Interactive Global Not-Matched Command**

87285 *Synopsis:* ( 1 , \$ ) **V/RE/**

87286 This command shall be equivalent to the interactive global command **G** except that the lines that  
87287 are marked during the first step shall be those for which the line excluding the terminating  
87288 <newline> does not match the RE.

## 87289 **Write Command**

87290 *Synopsis:* ( 1 , \$ ) **w** [*file*]

87291 The **w** command shall write the addressed lines into the file named by the pathname *file*. The  
87292 command shall create the file, if it does not exist, or shall replace the contents of the existing file.  
87293 The currently remembered pathname shall not be changed unless there is no remembered  
87294 pathname. If no pathname is given, the currently remembered pathname, if any, shall be used  
87295 (see the **e** and **f** commands); the current line number shall be unchanged. If the command is  
87296 successful, the number of bytes written shall be written to standard output, unless the **-s** option  
87297 was specified, in the following format:



87298 "%d\n", <number of bytes written>

87299 If *file* begins with '!', the rest of the line shall be taken to be a shell command line whose  
87300 standard input shall be the addressed lines. Such a shell command line shall not be remembered  
87301 as the current pathname. This usage of the write command with '!' shall not be considered as a  
87302 "last **w** command that wrote the entire buffer", as described previously; thus, this alone shall  
87303 not prevent the warning to the user if an attempt is made to destroy the editor buffer via the **e** or  
87304 **q** commands.

### 87305 **Line Number Command**

87306 *Synopsis:* (\$) =

87307 The line number of the addressed line shall be written to standard output in the following  
87308 format:

87309 "%d\n", <line number>

87310 The current line number shall be unchanged by this command.

### 87311 **Shell Escape Command**

87312 *Synopsis:* !*command*

87313 The remainder of the line after the '!' shall be sent to the command interpreter to be  
87314 interpreted as a shell command line. Within the text of that shell command line, the unescaped  
87315 character '%' shall be replaced with the remembered pathname; if a '!' appears as the first  
87316 character of the command, it shall be replaced with the text of the previous shell command  
87317 executed via '!'. Thus, "!!" shall repeat the previous !*command*. If any replacements of '%' or  
87318 '!' are performed, the modified line shall be written to the standard output before *command* is  
87319 executed. The ! command shall write:

87320 "!\n"

87321 to standard output upon completion, unless the -s option is specified. The current line number  
87322 shall be unchanged.

### 87323 **Null Command**

87324 *Synopsis:* (. +1)

87325 An address alone on a line shall cause the addressed line to be written. A <newline> alone shall  
87326 be equivalent to "+1p". The current line number shall be set to the address of the written line.

## 87327 **EXIT STATUS**

87328 The following exit values shall be returned:

87329 0 Successful completion without any file or command errors.

87330 >0 An error occurred.

## 87331 **CONSEQUENCES OF ERRORS**

87332 When an error in the input script is encountered, or when an error is detected that is a  
87333 consequence of the data (not) present in the file or due to an external condition such as a read or  
87334 write error:

- 87335 • If the standard input is a terminal device file, all input shall be flushed, and a new  
87336 command read.

- If the standard input is a regular file, *ed* shall terminate with a non-zero exit status.

## APPLICATION USAGE

Because of the extremely terse nature of the default error messages, the prudent script writer begins the *ed* input commands with an **H** command, so that if any errors do occur at least some clue as to the cause is made available.

In earlier versions of this standard, an obsolescent `–` option was described. This is no longer specified. Applications should use the `–s` option. Using `–` as a *file* operand now produces unspecified results. This allows implementations to continue to support the former required behavior.

## EXAMPLES

None.

## RATIONALE

The initial description of this utility was adapted from the SVID. It contains some features not found in Version 7 or BSD-derived systems. Some of the differences between the POSIX and BSD *ed* utilities include, but need not be limited to:

- The BSD `–` option does not suppress the `'!'` prompt after a `!` command.
- BSD does not support the special meanings of the `'%'` and `'!'` characters within a `!` command.
- BSD does not support the *addresses* `' ; '` and `' , '`.
- BSD allows the command/suffix pairs **pp**, **ll**, and so on, which are unspecified in this volume of POSIX.1-2008.
- BSD does not support the `'!'` character part of the **e**, **r**, or **w** commands.
- A failed **g** command in BSD sets the line number to the last line searched if there are no matches.
- BSD does not default the *command list* to the **p** command.
- BSD does not support the **G**, **h**, **H**, **n**, or **V** commands.
- On BSD, if there is no inserted text, the insert command changes the current line to the referenced line `–1`; that is, the line before the specified line.
- On BSD, the *join* command with only a single address changes the current line to that address.
- BSD does not support the **P** command; moreover, in BSD it is synonymous with the **p** command.
- BSD does not support the *undo* of the commands **j**, **m**, **r**, **s**, or **t**.
- The Version 7 *ed* command **W**, and the BSD *ed* commands **W**, **wq**, and **z** are not present in this volume of POSIX.1-2008.

The `–s` option was added to allow the functionality of the removed `–` option in a manner compatible with the Utility Syntax Guidelines.

In early proposals there was a limit, `{ED_FILE_MAX}`, that described the historical limitations of some *ed* utilities in their handling of large files; some of these have had problems with files larger than 100 000 bytes. It was this limitation that prompted much of the desire to include a *split* command in this volume of POSIX.1-2008. Since this limit was removed, this volume of POSIX.1-2008 requires that implementations document the file size limits imposed by *ed* in the

conformance document. The limit {ED\_LINE\_MAX} was also removed; therefore, the global limit {LINE\_MAX} is used for input and output lines.

The manner in which the **l** command writes non-printable characters was changed to avoid the historical backspace-overstrike method. On video display terminals, the overstrike is ambiguous because most terminals simply replace overstruck characters, making the **l** format not useful for its intended purpose of unambiguously understanding the content of the line. The historical <backslash>-escapes were also ambiguous. (The string "a\0011" could represent a line containing those six characters or a line containing the three characters 'a', a byte with a binary value of 1, and a 1.) In the format required here, a <backslash> appearing in the line is written as "\\ " so that the output is truly unambiguous. The method of marking the ends of lines was adopted from the *ex* editor and is required for any line ending in <space> characters; the '\$' is placed on all lines so that a real '\$' at the end of a line cannot be misinterpreted.

Earlier versions of this standard allowed for implementations with bytes other than eight bits, but this has been modified in this version.

The description of how a NUL is written was removed. The NUL character cannot be in text files, and this volume of POSIX.1-2008 should not dictate behavior in the case of undefined, erroneous input.

Unlike some of the other editing utilities, the filenames accepted by the **E**, **e**, **R**, and **r** commands are not patterns.

Early proposals stated that the **-p** option worked only when standard input was associated with a terminal device. This has been changed to conform to historical implementations, thereby allowing applications to interpose themselves between a user and the *ed* utility.

The form of the substitute command that uses the **n** suffix was limited in some historical documentation (where this was described incorrectly as "backreferencing"). This limit has been omitted because there is no reason why an editor processing lines of {LINE\_MAX} length should have this restriction. The command **s/x/X/2047** should be able to substitute the 2047th occurrence of 'x' on a line.

The use of printing commands with printing suffixes (such as **pn**, **lp**, and so on) was made unspecified because BSD-based systems allow this, whereas System V does not.

Some BSD-based systems exit immediately upon receipt of end-of-file if all of the lines in the file have been deleted. Since this volume of POSIX.1-2008 refers to the **q** command in this instance, such behavior is not allowed.

Some historical implementations returned exit status zero even if command errors had occurred; this is not allowed by this volume of POSIX.1-2008.

Some historical implementations contained a bug that allowed a single <period> to be entered in input mode as <backslash> <period> <newline>. This is not allowed by *ed* because there is no description of escaping any of the characters in input mode; <backslash> characters are entered into the buffer exactly as typed. The typical method of entering a single <period> has been to precede it with another character and then use the substitute command to delete that character.

It is difficult under some modes of some versions of historical operating system terminal drivers to distinguish between an end-of-file condition and terminal disconnect. POSIX.1-2008 does not require implementations to distinguish between the two situations, which permits historical implementations of the *ed* utility on historical platforms to conform. Implementations are encouraged to distinguish between the two, if possible, and take appropriate action on terminal disconnect.

Historically, *ed* accepted a zero address for the **a** and **r** commands in order to insert text at the

start of the edit buffer. When the buffer was empty the command `. =` returned zero. POSIX.1-2008 requires conformance to historical practice.

For consistency with the `a` and `r` commands and better user functionality, the `i` and `c` commands must also accept an address of 0, in which case `0i` is treated as `1i` and likewise for the `c` command.

All of the following are valid addresses:

```
+++           Three lines after the current line.
/pattern/-    One line before the next occurrence of pattern.
-2           Two lines before the current line.
3 ---- 2      Line one (note the intermediate negative address).
1 2 3         Line six.
```

Any number of addresses can be provided to commands taking addresses; for example, `"1,2,3,4,5p"` prints lines 4 and 5, because two is the greatest valid number of addresses accepted by the `print` command. This, in combination with the `<semicolon>` delimiter, permits users to create commands based on ordered patterns in the file. For example, the command `"3;/foo/;+2p"` will display the first line after line 3 that contains the pattern `foo`, plus the next two lines. Note that the address `"3;"` must still be evaluated before being discarded, because the search origin for the `"/foo/"` command depends on this.

Historically, `ed` disallowed address chains, as discussed above, consisting solely of `<comma>` or `<semicolon>` separators; for example, `","` or `";;"` were considered an error. For consistency of address specification, this restriction is removed. The following table lists some of the address forms now possible:

| Address | Addr1 | Addr2 | Status     | Comment               |
|---------|-------|-------|------------|-----------------------|
| 7,      | 7     | 7     | Historical | Valid, but erroneous. |
| 7,5,    | 5     | 5     | Historical |                       |
| 7,5,9   | 5     | 9     | Historical |                       |
| 7,9     | 7     | 9     | Historical |                       |
| 7,+     | 7     | 8     | Historical |                       |
| ,       | 1     | \$    | Historical |                       |
| ,7      | 1     | 7     | Extension  |                       |
| ,,      | \$    | \$    | Extension  |                       |
| ,i      | \$    | \$    | Extension  |                       |
| 7i      | 7     | 7     | Historical |                       |
| 7i5i    | 5     | 5     | Historical |                       |
| 7i5i9   | 5     | 9     | Historical |                       |
| 7i5,9   | 5     | 9     | Historical |                       |
| 7i\$;4  | \$    | 4     | Historical |                       |
| 7i9     | 7     | 9     | Historical |                       |
| 7i+     | 7     | 8     | Historical |                       |
| i       | .     | \$    | Historical |                       |
| i7      | .     | 7     | Extension  |                       |
| ii      | \$    | \$    | Extension  |                       |
| i,      | \$    | \$    | Extension  |                       |

Historically, `ed` accepted the `'^'` character as an address, in which case it was identical to the `<hyphen-minus>` character. POSIX.1-2008 does not require or prohibit this behavior.

**87470 FUTURE DIRECTIONS**

87471 None.

**87472 SEE ALSO**

87473 [Section 1.4](#) (on page 2326), *ex*, *sed*, *sh*, *vi*

87474 XBD [Table 5-1](#) (on page 121), [Chapter 8](#) (on page 173), [Section 9.3](#) (on page 183), [Chapter 11](#) (on  
87475 page 199), [Section 12.2](#) (on page 216)

**87476 CHANGE HISTORY**

87477 First released in Issue 2.

**87478 Issue 5**

87479 In the OPTIONS section, the meaning of `-s` and `-` is clarified.

87480 A second FUTURE DIRECTION is added.

**87481 Issue 6**

87482 The obsolescent single-minus form is removed.

87483 A second APPLICATION USAGE note is added.

87484 The Open Group Corrigendum U025/2 is applied, correcting the description of the Edit section.

87485 The *ed* utility is updated to align with the IEEE P1003.2b draft standard. This includes addition  
87486 of the treatment of the SIGQUIT signal, changes to *ed* addressing, and changes to processing  
87487 when end-of-file is detected and when terminal disconnect is detected.

87488 The normative text is reworded to avoid use of the term “must” for application requirements.

87489 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/22 is applied, adding the text: “Any line  
87490 modified by the *command list* shall be unmarked.” to the **G** command. This change corresponds  
87491 to a similar change made to the **g** command in the first version of this standard.

87492 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/7 is applied, removing text describing  
87493 behavior on systems with bytes consisting of more than eight bits.

**87494 Issue 7**

87495 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying the behavior if an operand is  
87496 ‘-’.

87497 Austin Group Interpretation 1003.1-2001 #036 is applied, clarifying the behavior for BREs.

87498 SD5-XCU-ERN-94 is applied, updating text in the EXTENDED DESCRIPTION where a terminal  
87499 disconnect is detected (in Commands in *ed*).

87500 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

87501 SD5-XCU-ERN-135 is applied, removing some RATIONALE text that is no longer applicable.

87502 **NAME**87503 `env` — set the environment for command invocation87504 **SYNOPSIS**87505 `env [-i] [name=value]... [utility [argument...]]`87506 **DESCRIPTION**87507 The *env* utility shall obtain the current environment, modify it according to its arguments, then  
87508 invoke the utility named by the *utility* operand with the modified environment.87509 Optional arguments shall be passed to *utility*.87510 If no *utility* operand is specified, the resulting environment shall be written to the standard  
87511 output, with one *name=value* pair per line.

87512 If the first argument is '-', the results are unspecified.

87513 **OPTIONS**87514 The *env* utility shall conform to XBD [Section 12.2](#) (on page 216), except for the unspecified usage  
87515 of '-'.  
87516 The following options shall be supported:

87516 The following options shall be supported:

87517 **-i** Invoke *utility* with exactly the environment specified by the arguments; the  
87518 inherited environment shall be ignored completely.87519 **OPERANDS**

87520 The following operands shall be supported:

87521 *name=value* Arguments of the form *name=value* shall modify the execution environment, and  
87522 shall be placed into the inherited environment before the *utility* is invoked.87523 *utility* The name of the utility to be invoked. If the *utility* operand names any of the  
87524 special built-in utilities in [Section 2.14](#) (on page 2374), the results are undefined.87525 *argument* A string to pass as an argument for the invoked utility.87526 **STDIN**

87527 Not used.

87528 **INPUT FILES**

87529 None.

87530 **ENVIRONMENT VARIABLES**87531 The following environment variables shall affect the execution of *env*:87532 *LANG* Provide a default value for the internationalization variables that are unset or null.  
87533 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
87534 variables used to determine the values of locale categories.)87535 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
87536 internationalization variables.87537 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
87538 characters (for example, single-byte as opposed to multi-byte characters in  
87539 arguments).87540 *LC\_MESSAGES*87541 Determine the locale that should be used to affect the format and contents of  
87542 diagnostic messages written to standard error.



|       |     |                               |                                                                                                                                                     |
|-------|-----|-------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------|
| 87543 | XSI | <b>NLSPATH</b>                | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                               |
| 87544 |     | <b>PATH</b>                   | Determine the location of the <i>utility</i> , as described in XBD <a href="#">Chapter 8</a> (on page 173).                                         |
| 87545 |     |                               | If <i>PATH</i> is specified as a <i>name=value</i> operand to <i>env</i> , the <i>value</i> given shall be used in                                  |
| 87546 |     |                               | the search for <i>utility</i> .                                                                                                                     |
| 87547 |     | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                     |
| 87548 |     |                               | Default.                                                                                                                                            |
| 87549 |     | <b>STDOUT</b>                 |                                                                                                                                                     |
| 87550 |     |                               | If no <i>utility</i> operand is specified, each <i>name=value</i> pair in the resulting environment shall be                                        |
| 87551 |     |                               | written in the form:                                                                                                                                |
| 87552 |     |                               | "%s=%s\n", <name>, <value>                                                                                                                          |
| 87553 |     |                               | If the <i>utility</i> operand is specified, the <i>env</i> utility shall not write to standard output.                                              |
| 87554 |     | <b>STDERR</b>                 |                                                                                                                                                     |
| 87555 |     |                               | The standard error shall be used only for diagnostic messages.                                                                                      |
| 87556 |     | <b>OUTPUT FILES</b>           |                                                                                                                                                     |
| 87557 |     |                               | None.                                                                                                                                               |
| 87558 |     | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                     |
| 87559 |     |                               | None.                                                                                                                                               |
| 87560 |     | <b>EXIT STATUS</b>            |                                                                                                                                                     |
| 87561 |     |                               | If <i>utility</i> is invoked, the exit status of <i>env</i> shall be the exit status of <i>utility</i> ; otherwise, the <i>env</i>                  |
| 87562 |     |                               | utility shall exit with one of the following values:                                                                                                |
| 87563 |     | 0                             | The <i>env</i> utility completed successfully.                                                                                                      |
| 87564 |     | 1–125                         | An error occurred in the <i>env</i> utility.                                                                                                        |
| 87565 |     | 126                           | The utility specified by <i>utility</i> was found but could not be invoked.                                                                         |
| 87566 |     | 127                           | The utility specified by <i>utility</i> could not be found.                                                                                         |
| 87567 |     | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                     |
| 87568 |     |                               | Default.                                                                                                                                            |
| 87569 |     | <b>APPLICATION USAGE</b>      |                                                                                                                                                     |
| 87570 |     |                               | The <i>command</i> , <i>env</i> , <i>nice</i> , <i>nohup</i> , <i>time</i> , and <i>xargs</i> utilities have been specified to use exit code 127 if |
| 87571 |     |                               | an error occurs so that applications can distinguish “failure to find a utility” from “invoked                                                      |
| 87572 |     |                               | utility exited with an error indication”. The value 127 was chosen because it is not commonly                                                       |
| 87573 |     |                               | used for other meanings; most utilities use small values for “normal error conditions” and the                                                      |
| 87574 |     |                               | values above 128 can be confused with termination due to receipt of a signal. The value 126 was                                                     |
| 87575 |     |                               | chosen in a similar manner to indicate that the utility could be found, but not invoked. Some                                                       |
| 87576 |     |                               | scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction                                                    |
| 87577 |     |                               | between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to                                                    |
| 87578 |     |                               | <i>exec</i> the utility fail with [ENOENT], and uses 126 when any attempt to <i>exec</i> the utility fails for                                      |
| 87579 |     |                               | any other reason.                                                                                                                                   |
| 87580 |     |                               | Historical implementations of the <i>env</i> utility use the <i>execvp()</i> or <i>execlp()</i> functions defined in the                            |
| 87581 |     |                               | System Interfaces volume of POSIX.1-2008 to invoke the specified utility; this provides better                                                      |
| 87582 |     |                               | performance and keeps users from having to escape characters with special meaning to the shell.                                                     |
| 87583 |     |                               | Therefore, shell functions, special built-ins, and built-ins that are only provided by the shell are                                                |
| 87584 |     |                               | not found.                                                                                                                                          |

**EXAMPLES**

The following command:

```
env -i PATH=/mybin:"$PATH" $(getconf V7_ENV) mygrep xyz myfile
```

invokes the command *mygrep* with a new *PATH* value as the only entry in its environment other than any variables required by the implementation for conformance. In this case, *PATH* is used to locate *mygrep*, which is expected to reside in */mybin*.

**RATIONALE**

As with all other utilities that invoke other utilities, this volume of POSIX.1-2008 only specifies what *env* does with standard input, standard output, standard error, input files, and output files. If a utility is executed, it is not constrained by the specification of input and output by *env*.

The *-i* option was added to allow the functionality of the removed *-* option in a manner compatible with the Utility Syntax Guidelines. It is possible to create a non-conforming environment using the *-i* option, as it may remove environment variables required by the implementation for conformance. The following will preserve these environment variables as well as preserve the *PATH* for conforming utilities:

```
IFS='
# The preceding value should be <space><tab><newline>.
# Set IFS to its default value.

set -f
# disable pathname expansion

\unalias -a
# Unset all possible aliases.
# Note that unalias is escaped to prevent an alias
# being used for unalias.
# This step is not strictly necessary, since aliases are not inherited,
# and the ENV environment variable is only used by interactive shells,
# the only way any aliases can exist in a script is if it defines them
# itself.

unset -f env getconf
# Ensure env and getconf are not user functions.

env -i $(getconf V7_ENV) PATH="$(getconf PATH)" command
```

Some have suggested that *env* is redundant since the same effect is achieved by:

```
name=value ... utility [ argument ... ]
```

The example is equivalent to *env* when an environment variable is being added to the environment of the command, but not when the environment is being set to the given value. The *env* utility also writes out the current environment if invoked without arguments. There is sufficient functionality beyond what the example provides to justify inclusion of *env*.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Section 2.14](#) (on page 2374), [Section 2.5](#) (on page 2339)

[XBD Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)



87628 **CHANGE HISTORY**

87629 First released in Issue 2.

87630 **Issue 7**87631 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying the behavior if the first  
87632 argument is '–'.87633 Austin Group Interpretation 1003.1-2001 #047 is applied, providing RATIONALE on how to use  
87634 the *env* utility to preserve a conforming environment.

87635 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

87636 The EXAMPLES section is revised to change the use of *env* –i.

## 87637 NAME

87638 ex — text editor

## 87639 SYNOPSIS

87640 UP `ex [-rR] [-s|-v] [-c command] [-t tagstring] [-w size] [file...]`

## 87641 DESCRIPTION

87642 The *ex* utility is a line-oriented text editor. There are two other modes of the editor—open and  
 87643 visual—in which screen-oriented editing is available. This is described more fully by the *ex* **open**  
 87644 and **visual** commands and in *vi*.

87645 If an operand is ‘-’, the results are unspecified.

87646 This section uses the term *edit buffer* to describe the current working text. No specific  
 87647 implementation is implied by this term. All editing changes are performed on the edit buffer,  
 87648 and no changes to it shall affect any file until an editor command writes the file.

87649 Certain terminals do not have all the capabilities necessary to support the complete *ex* definition,  
 87650 such as the full-screen editing commands (*visual mode* or *open mode*). When these commands  
 87651 cannot be supported on such terminals, this condition shall not produce an error message such  
 87652 as “not an editor command” or report a syntax error. The implementation may either accept the  
 87653 commands and produce results on the screen that are the result of an unsuccessful attempt to  
 87654 meet the requirements of this volume of POSIX.1-2008 or report an error describing the terminal-  
 87655 related deficiency.

## 87656 OPTIONS

87657 The *ex* utility shall conform to XBD [Section 12.2](#) (on page 216), except for the unspecified usage  
 87658 of ‘-’, and that ‘+’ may be recognized as an option delimiter as well as ‘-’.

87659 The following options shall be supported:

87660 **-c *command*** Specify an initial command to be executed in the first edit buffer loaded from an  
 87661 existing file (see the EXTENDED DESCRIPTION section). Implementations may  
 87662 support more than a single **-c** option. In such implementations, the specified  
 87663 commands shall be executed in the order specified on the command line.

87664 **-r** Recover the named files (see the EXTENDED DESCRIPTION section). Recovery  
 87665 information for a file shall be saved during an editor or system crash (for example,  
 87666 when the editor is terminated by a signal which the editor can catch), or after the  
 87667 use of an *ex* **preserve** command.

87668 A *crash* in this context is an unexpected failure of the system or utility that requires  
 87669 restarting the failed system or utility. A system crash implies that any utilities  
 87670 running at the time also crash. In the case of an editor or system crash, the number  
 87671 of changes to the edit buffer (since the most recent **preserve** command) that will be  
 87672 recovered is unspecified.

87673 If no *file* operands are given and the **-t** option is not specified, all other options, the  
 87674 *EXINIT* variable, and any **.exrc** files shall be ignored; a list of all recoverable files  
 87675 available to the invoking user shall be written, and the editor shall exit normally  
 87676 without further action.

87677 **-R** Set **readonly** edit option.

87678 **-s** Prepare *ex* for batch use by taking the following actions:

- 87679 • Suppress writing prompts and informational (but not diagnostic) messages.
- 87680 • Ignore the value of *TERM* and any implementation default terminal type and
- 87681 assume the terminal is a type incapable of supporting open or visual modes;
- 87682 see the **visual** command and the description of *vi*.
- 87683 • Suppress the use of the *EXINIT* environment variable and the reading of any
- 87684 **.exrc** file; see the EXTENDED DESCRIPTION section.
- 87685 • Suppress autoindentation, ignoring the value of the **autoindent** edit option.
- 87686 **-t tagstring** Edit the file containing the specified *tagstring*; see *ctags*. The tags feature
- 87687 represented by **-t tagstring** and the **tag** command is optional. It shall be provided
- 87688 on any system that also provides a conforming implementation of *ctags*; otherwise,
- 87689 the use of **-t** produces undefined results. On any system, it shall be an error to
- 87690 specify more than a single **-t** option.
- 87691 **-v** Begin in visual mode (see *vi*).
- 87692 **-w size** Set the value of the *window* editor option to *size*.

## 87693 OPERANDS

87694 The following operand shall be supported:

87695 *file* A pathname of a file to be edited.

## 87696 STDIN

87697 The standard input consists of a series of commands and input text, as described in the

87698 EXTENDED DESCRIPTION section. The implementation may limit each line of standard input

87699 to a length of {LINE\_MAX}.

87700 If the standard input is not a terminal device, it shall be as if the **-s** option had been specified.

87701 If a read from the standard input returns an error, or if the editor detects an end-of-file condition

87702 from the standard input, it shall be equivalent to a SIGHUP asynchronous event.

## 87703 INPUT FILES

87704 Input files shall be text files or files that would be text files except for an incomplete last line that

87705 is not longer than {LINE\_MAX}-1 bytes in length and contains no NUL characters. By default,

87706 any incomplete last line shall be treated as if it had a trailing <newline>. The editing of other

87707 forms of files may optionally be allowed by *ex* implementations.

87708 The **.exrc** files and source files shall be text files consisting of *ex* commands; see the EXTENDED

87709 DESCRIPTION section.

87710 By default, the editor shall read lines from the files to be edited without interpreting any of those

87711 lines as any form of editor command.

## 87712 ENVIRONMENT VARIABLES

87713 The following environment variables shall affect the execution of *ex*:

87714 **COLUMNS** Override the system-selected horizontal screen size. See XBD Chapter 8 (on page

87715 173) for valid values and results when it is unset or null.

87716 **EXINIT** Determine a list of *ex* commands that are executed on editor start-up. See the

87717 EXTENDED DESCRIPTION section for more details of the initialization phase.

87718 **HOME** Determine a pathname of a directory that shall be searched for an editor start-up

87719 file named **.exrc**; see the EXTENDED DESCRIPTION section.

|       |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 87720 | <i>LANG</i>                                                                                                                                                                                                                                                                                                                                                                   | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)                                                                                                                                                |
| 87721 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87722 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87723 | <i>LC_ALL</i>                                                                                                                                                                                                                                                                                                                                                                 | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                                                          |
| 87724 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87725 | <i>LC_COLLATE</i>                                                                                                                                                                                                                                                                                                                                                             |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87726 |                                                                                                                                                                                                                                                                                                                                                                               | Determine the locale for the behavior of ranges, equivalence classes, and multi-character collating elements within regular expressions.                                                                                                                                                                                                                                                          |
| 87727 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87728 | <i>LC_CTYPE</i>                                                                                                                                                                                                                                                                                                                                                               | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files), the behavior of character classes within regular expressions, the classification of characters as uppercase or lowercase letters, the case conversion of letters, and the detection of word boundaries. |
| 87729 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87730 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87731 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87732 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87733 | <i>LC_MESSAGES</i>                                                                                                                                                                                                                                                                                                                                                            |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87734 |                                                                                                                                                                                                                                                                                                                                                                               | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                                                                                                                                                                      |
| 87735 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87736 | <i>LINES</i>                                                                                                                                                                                                                                                                                                                                                                  | Override the system-selected vertical screen size, used as the number of lines in a screenful and the vertical screen size in visual mode. See XBD <a href="#">Chapter 8</a> (on page 173) for valid values and results when it is unset or null.                                                                                                                                                 |
| 87737 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87738 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87739 | XSI <i>NLSPATH</i>                                                                                                                                                                                                                                                                                                                                                            | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                                                             |
| 87740 | <i>PATH</i>                                                                                                                                                                                                                                                                                                                                                                   | Determine the search path for the shell command specified in the <i>ex</i> editor commands <b>!</b> , <b>shell</b> , <b>read</b> , and <b>write</b> , and the open and visual mode command <b>!</b> ; see the description of command search and execution in <a href="#">Section 2.9.1.1</a> (on page 2357).                                                                                      |
| 87741 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87742 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87743 | <i>SHELL</i>                                                                                                                                                                                                                                                                                                                                                                  | Determine the preferred command line interpreter for use as the default value of the <b>shell</b> edit option.                                                                                                                                                                                                                                                                                    |
| 87744 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87745 | <i>TERM</i>                                                                                                                                                                                                                                                                                                                                                                   | Determine the name of the terminal type. If this variable is unset or null, an unspecified default terminal type shall be used.                                                                                                                                                                                                                                                                   |
| 87746 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87747 | <b>ASYNCHRONOUS EVENTS</b>                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87748 | The following term is used in this and following sections to specify command and asynchronous event actions:                                                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87749 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87750 | <i>complete write</i>                                                                                                                                                                                                                                                                                                                                                         |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87751 | A complete write is a write of the entire contents of the edit buffer to a file of a type other than a terminal device, or the saving of the edit buffer caused by the user executing the <i>ex</i> <b>preserve</b> command. Writing the contents of the edit buffer to a temporary file that will be removed when the editor exits shall not be considered a complete write. |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87752 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87753 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87754 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87755 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87756 | The following actions shall be taken upon receipt of signals:                                                                                                                                                                                                                                                                                                                 |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87757 | <i>SIGINT</i>                                                                                                                                                                                                                                                                                                                                                                 | If the standard input is not a terminal device, <i>ex</i> shall not write the file or return to command or text input mode, and shall exit with a non-zero exit status.                                                                                                                                                                                                                           |
| 87758 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87759 |                                                                                                                                                                                                                                                                                                                                                                               | Otherwise, if executing an open or visual text input mode command, <i>ex</i> in receipt of <i>SIGINT</i> shall behave identically to its receipt of the <ESC> character.                                                                                                                                                                                                                          |
| 87760 |                                                                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                   |
| 87761 |                                                                                                                                                                                                                                                                                                                                                                               | Otherwise:                                                                                                                                                                                                                                                                                                                                                                                        |

- 87762 1. If executing an *ex* text input mode command, all input lines that have been  
87763 completely entered shall be resolved into the edit buffer, and any partially  
87764 entered line shall be discarded.
- 87765 2. If there is a currently executing command, it shall be aborted and a message  
87766 displayed. Unless otherwise specified by the *ex* or *vi* command descriptions,  
87767 it is unspecified whether any lines modified by the executing command  
87768 appear modified, or as they were before being modified by the executing  
87769 command, in the buffer.
- 87770 If the currently executing command was a motion command, its associated  
87771 command shall be discarded.
- 87772 3. If in open or visual command mode, the terminal shall be alerted.
- 87773 4. The editor shall then return to command mode.
- 87774 SIGCONT The screen shall be refreshed if in open or visual mode.
- 87775 SIGHUP If the edit buffer has been modified since the last complete write, *ex* shall attempt  
87776 to save the edit buffer so that it can be recovered later using the **-r** option or the *ex*  
87777 **recover** command. The editor shall not write the file or return to command or text  
87778 input mode, and shall terminate with a non-zero exit status.
- 87779 SIGTERM Refer to SIGHUP.
- 87780 The action taken for all other signals is unspecified.
- 87781 **STDOUT**
- 87782 The standard output shall be used only for writing prompts to the user, for informational  
87783 messages, and for writing lines from the file.
- 87784 **STDERR**
- 87785 The standard error shall be used only for diagnostic messages.
- 87786 **OUTPUT FILES**
- 87787 The output from *ex* shall be text files.
- 87788 **EXTENDED DESCRIPTION**
- 87789 Only the *ex* mode of the editor is described in this section. See *vi* for additional editing  
87790 capabilities available in *ex*.
- 87791 When an error occurs, *ex* shall write a message. If the terminal supports a standout mode (such  
87792 as inverse video), the message shall be written in standout mode. If the terminal does not  
87793 support a standout mode, and the edit option **errorbells** is set, an alert action shall precede the  
87794 error message.
- 87795 By default, *ex* shall start in command mode, which shall be indicated by a **:** prompt; see the  
87796 **prompt** command. Text input mode can be entered by the **append**, **insert**, or **change** commands;  
87797 it can be exited (and command mode re-entered) by typing a **<period>** (**' . '**) alone at the  
87798 beginning of a line.

## Initialization in ex and vi

The following symbols are used in this and following sections to specify locations in the edit buffer:

### *alternate and current pathnames*

Two pathnames, named *current* and *alternate*, are maintained by the editor. Any *ex* commands that take filenames as arguments shall set them as follows:

1. If a *file* argument is specified to the *ex* **edit**, **ex**, or **recover** commands, or if an *ex* **tag** command replaces the contents of the edit buffer.
  - a. If the command replaces the contents of the edit buffer, the current pathname shall be set to the *file* argument or the file indicated by the tag, and the alternate pathname shall be set to the previous value of the current pathname.
  - b. Otherwise, the alternate pathname shall be set to the *file* argument.
2. If a *file* argument is specified to the *ex* **next** command:
  - a. If the command replaces the contents of the edit buffer, the current pathname shall be set to the first *file* argument, and the alternate pathname shall be set to the previous value of the current pathname.
3. If a *file* argument is specified to the *ex* **file** command, the current pathname shall be set to the *file* argument, and the alternate pathname shall be set to the previous value of the current pathname.
4. If a *file* argument is specified to the *ex* **read** and **write** commands (that is, when reading or writing a file, and not to the program named by the **shell** edit option), or a *file* argument is specified to the *ex* **xit** command:
  - a. If the current pathname has no value, the current pathname shall be set to the *file* argument.
  - b. Otherwise, the alternate pathname shall be set to the *file* argument.

If the alternate pathname is set to the previous value of the current pathname when the current pathname had no previous value, then the alternate pathname shall have no value as a result.

### *current line*

The line of the edit buffer referenced by the cursor. Each command description specifies the current line after the command has been executed, as the *current line value*. When the edit buffer contains no lines, the current line shall be zero; see [Addressing in ex](#) (on page 2692).

### *current column*

The current display line column occupied by the cursor. (The columns shall be numbered beginning at 1.) Each command description specifies the current column after the command has been executed, as the *current column value*. This column is an *ideal* column that is remembered over the lifetime of the editor. The actual display line column upon which the cursor rests may be different from the current column; see the cursor positioning discussion in [Command Descriptions in vi](#) (on page 3365).

### *set to non-<blank>*

A description for a current column value, meaning that the current column shall be set to the last display line column on which is displayed any part of the first non-<blank> of the line. If the line has no non-<blank> non-<newline> characters, the current column shall be set to the last display line column on which is displayed any part of the last non-<newline>

87843 character in the line. If the line is empty, the current column shall be set to column position  
87844 1.

87845 The length of lines in the edit buffer may be limited to {LINE\_MAX} bytes. In open and visual  
87846 mode, the length of lines in the edit buffer may be limited to the number of characters that will  
87847 fit in the display. If either limit is exceeded during editing, an error message shall be written. If  
87848 either limit is exceeded by a line read in from a file, an error message shall be written and the  
87849 edit session may be terminated.

87850 If the editor stops running due to any reason other than a user command, and the edit buffer has  
87851 been modified since the last complete write, it shall be equivalent to a SIGHUP asynchronous  
87852 event. If the system crashes, it shall be equivalent to a SIGHUP asynchronous event.

87853 During initialization (before the first file is copied into the edit buffer or any user commands  
87854 from the terminal are processed) the following shall occur:

- 87855 1. If the environment variable *EXINIT* is set, the editor shall execute the *ex* commands  
87856 contained in that variable.
- 87857 2. If the *EXINIT* variable is not set, and all of the following are true:
  - 87858 a. The *HOME* environment variable is not null and not empty.
  - 87859 b. The file **.exrc** in the directory referred to by the *HOME* environment variable:
    - 87860 i. Exists
    - 87861 ii. Is owned by the same user ID as the real user ID of the process or the  
87862 process has appropriate privileges
    - 87863 iii. Is not writable by anyone other than the owner

87864 the editor shall execute the *ex* commands contained in that file.

- 87865 3. If and only if all of the following are true:
  - 87866 a. The current directory is not referred to by the *HOME* environment variable.
  - 87867 b. A command in the *EXINIT* environment variable or a command in the **.exrc** file in  
87868 the directory referred to by the *HOME* environment variable sets the editor option  
87869 **exrc**.
  - 87870 c. The **.exrc** file in the current directory:
    - 87871 i. Exists
    - 87872 ii. Is owned by the same user ID as the real user ID of the process, or by one of  
87873 a set of implementation-defined user IDs
    - 87874 iii. Is not writable by anyone other than the owner

87875 the editor shall attempt to execute the *ex* commands contained in that file.

87876 Lines in any **.exrc** file that are blank lines shall be ignored. If any **.exrc** file exists, but is not read  
87877 for ownership or permission reasons, it shall be an error.

87878 After the *EXINIT* variable and any **.exrc** files are processed, the first file specified by the user  
87879 shall be edited, as follows:

- 87880 1. If the user specified the **-t** option, the effect shall be as if the *ex tag* command was entered  
87881 with the specified argument, with the exception that if tag processing does not result in a  
87882 file to edit, the effect shall be as described in step 3. below.



2. Otherwise, if the user specified any command line *file* arguments, the effect shall be as if the *ex* **edit** command was entered with the first of those arguments as its *file* argument.
3. Otherwise, the effect shall be as if the *ex* **edit** command was entered with a nonexistent filename as its *file* argument. It is unspecified whether this action shall set the current pathname. In an implementation where this action does not set the current pathname, any editor command using the current pathname shall fail until an editor command sets the current pathname.

If the **-r** option was specified, the first time a file in the initial argument list or a file specified by the **-t** option is edited, if recovery information has previously been saved about it, that information shall be recovered and the editor shall behave as if the contents of the edit buffer have already been modified. If there are multiple instances of the file to be recovered, the one most recently saved shall be recovered, and an informational message that there are previous versions of the file that can be recovered shall be written. If no recovery information about a file is available, an informational message to this effect shall be written, and the edit shall proceed as usual.

If the **-c** option was specified, the first time a file that already exists (including a file that might not exist but for which recovery information is available, when the **-r** option is specified) replaces or initializes the contents of the edit buffer, the current line shall be set to the last line of the edit buffer, the current column shall be set to non-`<blank>`, and the *ex* commands specified with the **-c** option shall be executed. In this case, the current line and current column shall not be set as described for the command associated with the replacement or initialization of the edit buffer contents. However, if the **-t** option or a **tag** command is associated with this action, the **-c** option commands shall be executed and then the movement to the tag shall be performed.

The current argument list shall initially be set to the filenames specified by the user on the command line. If no filenames are specified by the user, the current argument list shall be empty. If the **-t** option was specified, it is unspecified whether any filename resulting from tag processing shall be prepended to the current argument list. In the case where the filename is added as a prefix to the current argument list, the current argument list reference shall be set to that filename. In the case where the filename is not added as a prefix to the current argument list, the current argument list reference shall logically be located before the first of the filenames specified on the command line (for example, a subsequent *ex* **next** command shall edit the first filename from the command line). If the **-t** option was not specified, the current argument list reference shall be to the first of the filenames on the command line.

### Addressing in *ex*

Addressing in *ex* relates to the current line and the current column; the address of a line is its 1-based line number, the address of a column is its 1-based count from the beginning of the line. Generally, the current line is the last line affected by a command. The current line number is the address of the current line. In each command description, the effect of the command on the current line number and the current column is described.

Addresses are constructed as follows:

1. The character `'.'` (period) shall address the current line.
2. The character `'$'` shall address the last line of the edit buffer.
3. The positive decimal number *n* shall address the *n*th line of the edit buffer.
4. The address `"'x"` refers to the line marked with the mark name character `'x'`, which shall be a lowercase letter from the portable character set, the backquote character, or the single-quote character. It shall be an error if the line that was marked is not currently



present in the edit buffer or the mark has not been set. Lines can be marked with the *ex mark* or *k* commands, or the *vi m* command.

5. A regular expression enclosed by <slash> characters ( '/' ) shall address the first line found by searching forwards from the line following the current line toward the end of the edit buffer and stopping at the first line for which the line excluding the terminating <newline> matches the regular expression. As stated in [Regular Expressions in ex](#) (on page 2723), an address consisting of a null regular expression delimited by <slash> characters ( " / / " ) shall address the next line for which the line excluding the terminating <newline> matches the last regular expression encountered. In addition, the second <slash> can be omitted at the end of a command line. If the **wrapscan** edit option is set, the search shall wrap around to the beginning of the edit buffer and continue up to and including the current line, so that the entire edit buffer is searched. Within the regular expression, the sequence "\/" shall represent a literal <slash> instead of the regular expression delimiter.
6. A regular expression enclosed in <question-mark> characters ( ' ? ' ) shall address the first line found by searching backwards from the line preceding the current line toward the beginning of the edit buffer and stopping at the first line for which the line excluding the terminating <newline> matches the regular expression. An address consisting of a null regular expression delimited by <question-mark> characters ( " ? ? " ) shall address the previous line for which the line excluding the terminating <newline> matches the last regular expression encountered. In addition, the second <question-mark> can be omitted at the end of a command line. If the **wrapscan** edit option is set, the search shall wrap around from the beginning of the edit buffer to the end of the edit buffer and continue up to and including the current line, so that the entire edit buffer is searched. Within the regular expression, the sequence "\?" shall represent a literal <question-mark> instead of the RE delimiter.
7. A <plus-sign> ( '+' ) or a <hyphen-minus> ( '-' ) followed by a decimal number shall address the current line plus or minus the number. A '+' or '-' not followed by a decimal number shall address the current line plus or minus 1.

Addresses can be followed by zero or more address offsets, optionally <blank>-separated. Address offsets are constructed as follows:

1. A '+' or '-' immediately followed by a decimal number shall add (subtract) the indicated number of lines to (from) the address. A '+' or '-' not followed by a decimal number shall add (subtract) 1 to (from) the address.
2. A decimal number shall add the indicated number of lines to the address.

It shall not be an error for an intermediate address value to be less than zero or greater than the last line in the edit buffer. It shall be an error for the final address value to be less than zero or greater than the last line in the edit buffer.

Commands take zero, one, or two addresses; see the descriptions of *1addr* and *2addr* in [Command Descriptions in ex](#) (on page 2699). If more than the required number of addresses are provided to a command that requires zero addresses, it shall be an error. Otherwise, if more than the required number of addresses are provided to a command, the addresses specified first shall be evaluated and then discarded until the maximum number of valid addresses remain.

Addresses shall be separated from each other by a <comma> ( ',' ) or a <semicolon> ( ';' ). If no address is specified before or after a <comma> or <semicolon> separator, it shall be as if the address of the current line was specified before or after the separator. In the case of a <semicolon> separator, the current line ( '.' ) shall be set to the first address, and only then will

the next address be calculated. This feature can be used to determine the starting line for forwards and backwards searches (see rules 5. and 6.).

A <percent-sign> ( ' % ' ) shall be equivalent to entering the two addresses " 1 , \$ " .

Any delimiting <blank> characters between addresses, address separators, or address offsets shall be discarded.

### Command Line Parsing in ex

The following symbol is used in this and following sections to describe parsing behavior:

*escape* If a character is referred to as "<backslash>-escaped" or "<control>-V-escaped", it shall mean that the character acquired or lost a special meaning by virtue of being preceded, respectively, by a <backslash> or <control>-V character. Unless otherwise specified, the escaping character shall be discarded at that time and shall not be further considered for any purpose.

Command-line parsing shall be done in the following steps. For each step, characters already evaluated shall be ignored; that is, the phrase "leading character" refers to the next character that has not yet been evaluated.

1. Leading <colon> characters shall be skipped.
  2. Leading <blank> characters shall be skipped.
  3. If the leading character is a double-quote character, the characters up to and including the next non-<backslash>-escaped <newline> shall be discarded, and any subsequent characters shall be parsed as a separate command.
  4. Leading characters that can be interpreted as addresses shall be evaluated; see [Addressing in ex](#) (on page 2692).
  5. Leading <blank> characters shall be skipped.
  6. If the next character is a <vertical-line> character or a <newline>:
    - a. If the next character is a <newline>:
      - i. If *ex* is in open or visual mode, the current line shall be set to the last address specified, if any.
      - ii. Otherwise, if the last command was terminated by a <vertical-line> character, no action shall be taken; for example, the command " | | <newline> " shall execute two implied commands, not three.
      - iii. Otherwise, step 6.b. shall apply.
    - b. Otherwise, the implied command shall be the **print** command. The last #, **p**, and **l** flags specified to any *ex* command shall be remembered and shall apply to this implied command. Executing the *ex* **number**, **print**, or **list** command shall set the remembered flags to #, nothing, and **l**, respectively, plus any other flags specified for that execution of the **number**, **print**, or **list** command.
- If *ex* is not currently performing a **global** or **v** command, and no address or count is specified, the current line shall be incremented by 1 before the command is executed. If incrementing the current line would result in an address past the last line in the edit buffer, the command shall fail, and the increment shall not happen.

- 88016 c. The <newline> or <vertical-line> character shall be discarded and any subsequent  
88017 characters shall be parsed as a separate command.
- 88018 7. The command name shall be comprised of the next character (if the character is not  
88019 alphabetic), or the next character and any subsequent alphabetic characters (if the  
88020 character is alphabetic), with the following exceptions:
- 88021 a. Commands that consist of any prefix of the characters in the command name  
88022 **delete**, followed immediately by any of the characters 'l', 'p', '+', '-', or '#'  
88023 shall be interpreted as a **delete** command, followed by a <blank>, followed by the  
88024 characters that were not part of the prefix of the **delete** command. The maximum  
88025 number of characters shall be matched to the command name **delete**; for example,  
88026 "del" shall not be treated as "de" followed by the flag l.
- 88027 b. Commands that consist of the character 'k', followed by a character that can be  
88028 used as the name of a mark, shall be equivalent to the mark command followed by  
88029 a <blank>, followed by the character that followed the 'k'.
- 88030 c. Commands that consist of the character 's', followed by characters that could be  
88031 interpreted as valid options to the s command, shall be the equivalent of the s  
88032 command, without any pattern or replacement values, followed by a <blank>,  
88033 followed by the characters after the 's'.
- 88034 8. The command name shall be matched against the possible command names, and a  
88035 command name that contains a prefix matching the characters specified by the user shall  
88036 be the executed command. In the case of commands where the characters specified by the  
88037 user could be ambiguous, the executed command shall be as follows:

|    |        |    |       |    |       |
|----|--------|----|-------|----|-------|
| a  | append | n  | next  | t  | t     |
| c  | change | p  | print | u  | undo  |
| ch | change | pr | print | un | undo  |
| e  | edit   | r  | read  | v  | v     |
| m  | move   | re | read  | w  | write |
| ma | mark   | s  | s     |    |       |

- 88044 Implementation extensions with names causing similar ambiguities shall not be checked  
88045 for a match until all possible matches for commands specified by POSIX.1-2008 have been  
88046 checked.
- 88047 9. If the command is a ! command, or if the command is a **read** command followed by zero  
88048 or more <blank> characters and a !, or if the command is a **write** command followed by  
88049 one or more <blank> characters and a !, the rest of the command shall include all  
88050 characters up to a non-<backslash>-escaped <newline>. The <newline> shall be  
88051 discarded and any subsequent characters shall be parsed as a separate *ex* command.
- 88052 10. Otherwise, if the command is an **edit**, **ex**, or **next** command, or a **visual** command while  
88053 in open or visual mode, the next part of the command shall be parsed as follows:
- 88054 a. Any '!' character immediately following the command shall be skipped and be  
88055 part of the command.
- 88056 b. Any leading <blank> characters shall be skipped and be part of the command.
- 88057 c. If the next character is a '+', characters up to the first non-<backslash>-escaped  
88058 <newline> or non-<backslash>-escaped <blank> shall be skipped and be part of  
88059 the command.

- 88060 d. The rest of the command shall be determined by the steps specified in paragraph  
88061 12.
- 88062 11. Otherwise, if the command is a **global**, **open**, **s**, or **v** command, the next part of the  
88063 command shall be parsed as follows:
- 88064 a. Any leading <blank> characters shall be skipped and be part of the command.
- 88065 b. If the next character is not an alphanumeric, double-quote, <newline>,  
88066 <backslash>, or <vertical-line> character:
- 88067 i. The next character shall be used as a command delimiter.
- 88068 ii. If the command is a **global**, **open**, or **v** command, characters up to the first  
88069 non-<backslash>-escaped <newline>, or first non-<backslash>-escaped  
88070 delimiter character, shall be skipped and be part of the command.
- 88071 iii. If the command is an **s** command, characters up to the first  
88072 non-<backslash>-escaped <newline>, or second non-<backslash>-escaped  
88073 delimiter character, shall be skipped and be part of the command.
- 88074 c. If the command is a **global** or **v** command, characters up to the first  
88075 non-<backslash>-escaped <newline> shall be skipped and be part of the  
88076 command.
- 88077 d. Otherwise, the rest of the command shall be determined by the steps specified in  
88078 paragraph 12.
- 88079 12. Otherwise:
- 88080 a. If the command was a **map**, **unmap**, **abbreviate**, or **unabbreviate** command,  
88081 characters up to the first non-<control>-V-escaped <newline>, <vertical-line>, or  
88082 double-quote character shall be skipped and be part of the command.
- 88083 b. Otherwise, characters up to the first non-<backslash>-escaped <newline>,  
88084 <vertical-line>, or double-quote character shall be skipped and be part of the  
88085 command.
- 88086 c. If the command was an **append**, **change**, or **insert** command, and the step 12.b.  
88087 ended at a <vertical-line> character, any subsequent characters, up to the next  
88088 non-<backslash>-escaped <newline> shall be used as input text to the command.
- 88089 d. If the command was ended by a double-quote character, all subsequent characters,  
88090 up to the next non-<backslash>-escaped <newline>, shall be discarded.
- 88091 e. The terminating <newline> or <vertical-line> character shall be discarded and any  
88092 subsequent characters shall be parsed as a separate *ex* command.

88093 Command arguments shall be parsed as described by the Synopsis and Description of each  
88094 individual *ex* command. This parsing shall not be <blank>-sensitive, except for the **!** argument,  
88095 which must follow the command name without intervening <blank> characters, and where it  
88096 would otherwise be ambiguous. For example, *count* and *flag* arguments need not be  
88097 <blank>-separated because "d22p" is not ambiguous, but *file* arguments to the *ex* **next**  
88098 command must be separated by one or more <blank> characters. Any <blank> in command  
88099 arguments for the **abbreviate**, **unabbreviate**, **map**, and **unmap** commands can be  
88100 <control>-V-escaped, in which case the <blank> shall not be used as an argument delimiter. Any  
88101 <blank> in the command argument for any other command can be <backslash>-escaped, in  
88102 which case that <blank> shall not be used as an argument delimiter.

88103 Within command arguments for the **abbreviate**, **unabbreviate**, **map**, and **unmap** commands,

any character can be <control>-V-escaped. All such escaped characters shall be treated literally and shall have no special meaning. Within command arguments for all other *ex* commands that are not regular expressions or replacement strings, any character that would otherwise have a special meaning can be <backslash>-escaped. Escaped characters shall be treated literally, without special meaning as shell expansion characters or '!', '%', and '#' expansion characters. See [Regular Expressions in ex](#) (on page 2723) and [Replacement Strings in ex](#) (on page 2724) for descriptions of command arguments that are regular expressions or replacement strings.

Non-<backslash>-escaped '%' characters appearing in *file* arguments to any *ex* command shall be replaced by the current pathname; unescaped '#' characters shall be replaced by the alternate pathname. It shall be an error if '%' or '#' characters appear unescaped in an argument and their corresponding values are not set.

Non-<backslash>-escaped '!' characters in the arguments to either the *ex* ! command or the open and visual mode ! command, or in the arguments to the *ex* **read** command, where the first non-<blank> after the command name is a '!' character, or in the arguments to the *ex* **write** command where the command name is followed by one or more <blank> characters and the first non-<blank> after the command name is a '!' character, shall be replaced with the arguments to the last of those three commands as they appeared after all unescaped '%', '#', and '!' characters were replaced. It shall be an error if '!' characters appear unescaped in one of these commands and there has been no previous execution of one of these commands.

If an error occurs during the parsing or execution of an *ex* command:

- An informational message to this effect shall be written. Execution of the *ex* command shall stop, and the cursor (for example, the current line and column) shall not be further modified.
- If the *ex* command resulted from a map expansion, all characters from that map expansion shall be discarded, except as otherwise specified by the **map** command.
- Otherwise, if the *ex* command resulted from the processing of an *EXINIT* environment variable, a **.exrc** file, a **:source** command, a **-c** option, or a **+command** specified to an *ex* **edit**, **ex**, **next**, or **visual** command, no further commands from the source of the commands shall be executed.
- Otherwise, if the *ex* command resulted from the execution of a buffer or a **global** or **v** command, no further commands caused by the execution of the buffer or the **global** or **v** command shall be executed.
- Otherwise, if the *ex* command was not terminated by a <newline>, all characters up to and including the next non-<backslash>-escaped <newline> shall be discarded.

### Input Editing in ex

The following symbol is used in this and the following sections to specify command actions:

*word* In the POSIX locale, a word consists of a maximal sequence of letters, digits, and underscores, delimited at both ends by characters other than letters, digits, or underscores, or by the beginning or end of a line or the edit buffer.

When accepting input characters from the user, in either *ex* command mode or *ex* text input mode, *ex* shall enable canonical mode input processing, as defined in the System Interfaces volume of POSIX.1-2008.

If in *ex* text input mode:

1. If the **number** edit option is set, *ex* shall prompt for input using the line number that would be assigned to the line if it is entered, in the format specified for the *ex* **number** command.
2. If the **autoindent** edit option is set, *ex* shall prompt for input using **autoindent** characters, as described by the **autoindent** edit option. **autoindent** characters shall follow the line number, if any.

If in *ex* command mode:

1. If the **prompt** edit option is set, input shall be prompted for using a single ' : ' character; otherwise, there shall be no prompt.

The input characters in the following sections shall have the following effects on the input line.

## Scroll

*Synopsis:*      eof

See the description of the *stty* eof character in *stty*.

If in *ex* command mode:

If the eof character is the first character entered on the line, the line shall be evaluated as if it contained two characters: a <control>-D and a <newline>.

Otherwise, the eof character shall have no special meaning.

If in *ex* text input mode:

If the cursor follows an **autoindent** character, the **autoindent** characters in the line shall be modified so that a part of the next text input character will be displayed on the first column in the line after the previous **shiftwidth** edit option column boundary, and the user shall be prompted again for input for the same line.

Otherwise, if the cursor follows a ' 0 ', which follows an **autoindent** character, and the ' 0 ' was the previous text input character, the ' 0 ' and all **autoindent** characters in the line shall be discarded, and the user shall be prompted again for input for the same line.

Otherwise, if the cursor follows a ' ^ ', which follows an **autoindent** character, and the ' ^ ' was the previous text input character, the ' ^ ' and all **autoindent** characters in the line shall be discarded, and the user shall be prompted again for input for the same line. In addition, the **autoindent** level for the next input line shall be derived from the same line from which the **autoindent** level for the current input line was derived.

Otherwise, if there are no **autoindent** or text input characters in the line, the eof character shall be discarded.

Otherwise, the eof character shall have no special meaning.



**<newline>**

*Synopsis:*     <newline>  
                  <control>-J

If in *ex* command mode:

Cause the command line to be parsed; <control>-J shall be mapped to the <newline> for this purpose.

If in *ex* text input mode:

Terminate the current line. If there are no characters other than **autoindent** characters on the line, all characters on the line shall be discarded.

Prompt for text input on a new line after the current line. If the **autoindent** edit option is set, an appropriate number of **autoindent** characters shall be added as a prefix to the line as described by the *ex* **autoindent** edit option.

**<backslash>**

*Synopsis:*     <backslash>

Allow the entry of a subsequent <newline> or <control>-J as a literal character, removing any special meaning that it may have to the editor during text input mode. The <backslash> character shall be retained and evaluated when the command line is parsed, or retained and included when the input text becomes part of the edit buffer.

**<control>-V**

*Synopsis:*     <control>-V

Allow the entry of any subsequent character as a literal character, removing any special meaning that it may have to the editor during text input mode. The <control>-V character shall be discarded before the command line is parsed or the input text becomes part of the edit buffer.

If the “literal next” functionality is performed by the underlying system, it is implementation-defined whether a character other than <control>-V performs this function.

**<control>-W**

*Synopsis:*     <control>-W

Discard the <control>-W, and the word previous to it in the input line, including any <blank> characters following the word and preceding the <control>-W. If the “word erase” functionality is performed by the underlying system, it is implementation-defined whether a character other than <control>-W performs this function.

**Command Descriptions in ex**

The following symbols are used in this section to represent command modifiers. Some of these modifiers can be omitted, in which case the specified defaults shall be used.

*laddr*       A single line address, given in any of the forms described in [Addressing in ex](#) (on page 2692); the default shall be the current line ( ' . ' ), unless otherwise specified.

If the line address is zero, it shall be an error, unless otherwise specified in the following command descriptions.

|       |               |                                                                                                                                |
|-------|---------------|--------------------------------------------------------------------------------------------------------------------------------|
| 88219 |               | If the edit buffer is empty, and the address is specified with a command other than                                            |
| 88220 |               | <b>=, append, insert, open, put, read,</b> or <b>visual</b> , or the address is not zero, it shall be                          |
| 88221 |               | an error.                                                                                                                      |
| 88222 | <i>2addr</i>  | Two addresses specifying an inclusive range of lines. If no addresses are specified,                                           |
| 88223 |               | the default for <i>2addr</i> shall be the current line only (" . , . "), unless otherwise                                      |
| 88224 |               | specified in the following command descriptions. If one address is specified, <i>2addr</i>                                     |
| 88225 |               | shall specify that line only, unless otherwise specified in the following command                                              |
| 88226 |               | descriptions.                                                                                                                  |
| 88227 |               | It shall be an error if the first address is greater than the second address.                                                  |
| 88228 |               | If the edit buffer is empty, and the two addresses are specified with a command                                                |
| 88229 |               | other than the <b>!, write, wq,</b> or <b>xit</b> commands, or either address is not zero, it shall                            |
| 88230 |               | be an error.                                                                                                                   |
| 88231 | <i>count</i>  | A positive decimal number. If <i>count</i> is specified, it shall be equivalent to specifying                                  |
| 88232 |               | an additional address to the command, unless otherwise specified by the following                                              |
| 88233 |               | command descriptions. The additional address shall be equal to the last address                                                |
| 88234 |               | specified to the command (either explicitly or by default) plus <i>count</i> –1.                                               |
| 88235 |               | If this would result in an address greater than the last line of the edit buffer, it shall                                     |
| 88236 |               | be corrected to equal the last line of the edit buffer.                                                                        |
| 88237 | <i>flags</i>  | One or more of the characters <b>'+', '-'</b> , <b>'#'</b> , <b>'p'</b> , or <b>'l'</b> (ell). The flag characters             |
| 88238 |               | can be <blank>-separated, and in any order or combination. The characters <b>'#'</b> ,                                         |
| 88239 |               | <b>'p'</b> , and <b>'l'</b> shall cause lines to be written in the format specified by the <b>print</b>                        |
| 88240 |               | command with the specified <i>flags</i> .                                                                                      |
| 88241 |               | The lines to be written are as follows:                                                                                        |
| 88242 |               | 1. All edit buffer lines written during the execution of the <i>ex</i> <b>&amp;</b> , <b>~</b> , <b>list</b> , <b>number</b> , |
| 88243 |               | <b>open</b> , <b>print</b> , <b>s</b> , <b>visual</b> , and <b>z</b> commands shall be written as specified by <i>flags</i> .  |
| 88244 |               | 2. After the completion of an <i>ex</i> command with a flag as an argument, the                                                |
| 88245 |               | current line shall be written as specified by <i>flags</i> , unless the current line was                                       |
| 88246 |               | the last line written by the command.                                                                                          |
| 88247 |               | The characters <b>'+'</b> and <b>'-'</b> cause the value of the current line after the execution                               |
| 88248 |               | of the <i>ex</i> command to be adjusted by the offset address as described in <a href="#">Addressing</a>                       |
| 88249 |               | <a href="#">in ex</a> (on page 2692). This adjustment shall occur before the current line is written                           |
| 88250 |               | as described in 2. above.                                                                                                      |
| 88251 |               | The default for <i>flags</i> shall be none.                                                                                    |
| 88252 | <i>buffer</i> | One of a number of named areas for holding text. The named buffers are specified                                               |
| 88253 |               | by the alphanumeric characters of the POSIX locale. There shall also be one                                                    |
| 88254 |               | "unnamed" buffer. When no buffer is specified for editor commands that use a                                                   |
| 88255 |               | buffer, the unnamed buffer shall be used. Commands that store text into buffers                                                |
| 88256 |               | shall store the text as it was before the command took effect, and shall store text                                            |
| 88257 |               | occurring earlier in the file before text occurring later in the file, regardless of how                                       |
| 88258 |               | the text region was specified. Commands that store text into buffers shall store the                                           |
| 88259 |               | text into the unnamed buffer as well as any specified buffer.                                                                  |
| 88260 |               | In <i>ex</i> commands, buffer names are specified as the name by itself. In open or visual                                     |
| 88261 |               | mode commands the name is preceded by a double-quote ( <b>' '</b> ) character.                                                 |
| 88262 |               | If the specified buffer name is an uppercase character, and the buffer contents are                                            |
| 88263 |               | to be modified, the buffer shall be appended to rather than being overwritten. If                                              |



the buffer is not being modified, specifying the buffer name in lowercase and uppercase shall have identical results.

There shall also be buffers named by the numbers 1 through 9. In open and visual mode, if a region of text including characters from more than a single line is being modified by the *vi* **c** or **d** commands, the motion character associated with the **c** or **d** commands specifies that the buffer text shall be in line mode, or the commands **%**, **'**, **/**, **?**, **(**, **)**, **N**, **n**, **{**, or **}** are used to define a region of text for the **c** or **d** commands, the contents of buffers 1 through 8 shall be moved into the buffer named by the next numerically greater value, the contents of buffer 9 shall be discarded, and the region of text shall be copied into buffer 1. This shall be in addition to copying the text into a user-specified buffer or unnamed buffer, or both. Numeric buffers can be specified as a source buffer for open and visual mode commands; however, specifying a numeric buffer as the write target of an open or visual mode command shall have unspecified results.

The text of each buffer shall have the characteristic of being in either line or character mode. Appending text to a non-empty buffer shall set the mode to match the characteristic of the text being appended. Appending text to a buffer shall cause the creation of at least one additional line in the buffer. All text stored into buffers by *ex* commands shall be in line mode. The *ex* commands that use buffers as the source of text specify individually how buffers of different modes are handled. Each open or visual mode command that uses buffers for any purpose specifies individually the mode of the text stored into the buffer and how buffers of different modes are handled.

*file* Command text used to derive a pathname. The default shall be the current pathname, as defined previously, in which case, if no current pathname has yet been established it shall be an error, except where specifically noted in the individual command descriptions that follow. If the command text contains any of the characters **'**, **~**, **'**, **{**, **'**, **[**, **'**, **\***, **'**, **?**, **'**, **\$**, **'**, **"**, backquote, single-quote, and **<backslash>**, it shall be subjected to the process of "shell expansions", as described below; if more than a single pathname results and the command expects only one, it shall be an error.

The process of shell expansions in the editor shall be done as follows. The *ex* utility shall pass two arguments to the program named by the shell edit option; the first shall be **-c**, and the second shall be the string "echo" and the command text as a single argument. The standard output and standard error of that command shall replace the command text.

**!** A character that can be appended to the command name to modify its operation, as detailed in the individual command descriptions. With the exception of the *ex* **read**, **write**, and **!** commands, the **'** character shall only act as a modifier if there are no **<blank>** characters between it and the command name.

*remembered search direction*

The *vi* commands **N** and **n** begin searching in a forwards or backwards direction in the edit buffer based on a remembered search direction, which is initially unset, and is set by the *ex* **global**, **v**, **s**, and **tag** commands, and the *vi* **/** and **?** commands.

**Abbreviate**

*Synopsis:*     ab[*breviate*][*lhs rhs*]

If *lhs* and *rhs* are not specified, write the current list of abbreviations and do nothing more.

Implementations may restrict the set of characters accepted in *lhs* or *rhs*, except that printable characters and <blank> characters shall not be restricted. Additional restrictions shall be implementation-defined.

In both *lhs* and *rhs*, any character may be escaped with a <control>-V, in which case the character shall not be used to delimit *lhs* from *rhs*, and the escaping <control>-V shall be discarded.

In open and visual text input mode, if a non-word or <ESC> character that is not escaped by a <control>-V character is entered after a word character, a check shall be made for a set of characters matching *lhs*, in the text input entered during this command. If it is found, the effect shall be as if *rhs* was entered instead of *lhs*.

The set of characters that are checked is defined as follows:

1. If there are no characters inserted before the word and non-word or <ESC> characters that triggered the check, the set of characters shall consist of the word character.
2. If the character inserted before the word and non-word or <ESC> characters that triggered the check is a word character, the set of characters shall consist of the characters inserted immediately before the triggering characters that are word characters, plus the triggering word character.
3. If the character inserted before the word and non-word or <ESC> characters that triggered the check is not a word character, the set of characters shall consist of the characters that were inserted before the triggering characters that are neither <blank> characters nor word characters, plus the triggering word character.

It is unspecified whether the *lhs* argument entered for the *ex* **abbreviate** and **unabbreviate** commands is replaced in this fashion. Regardless of whether or not the replacement occurs, the effect of the command shall be as if the replacement had not occurred.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Append**

*Synopsis:*     [*laddr*] a[*ppend*][!]

Enter *ex* text input mode; the input text shall be placed after the specified line. If line zero is specified, the text shall be placed at the beginning of the edit buffer.

This command shall be affected by the **number** and **autoindent** edit options; following the command name with '!' shall cause the **autoindent** edit option setting to be toggled for the duration of this command only.

*Current line:* Set to the last input line; if no lines were input, set to the specified line, or to the first line of the edit buffer if a line of zero was specified, or zero if the edit buffer is empty.

*Current column:* Set to non-<blank>.

**Arguments**

*Synopsis:*     ar[gs]

Write the current argument list, with the current argument-list entry, if any, between ' [ ' and ' ] ' characters.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Change**

*Synopsis:*     [2addr] c[hange][!][count]

Enter *ex* text input mode; the input text shall replace the specified lines. The specified lines shall be copied into the unnamed buffer, which shall become a line mode buffer.

This command shall be affected by the **number** and **autoindent** edit options; following the command name with ' ! ' shall cause the **autoindent** edit option setting to be toggled for the duration of this command only.

*Current line:* Set to the last input line; if no lines were input, set to the line before the first address, or to the first line of the edit buffer if there are no lines preceding the first address, or to zero if the edit buffer is empty.

*Current column:* Set to non-<blank>.

**Change Directory**

*Synopsis:*     chd[ir][!][directory]  
              cd[!][directory]

Change the current working directory to *directory*.

If no *directory* argument is specified, and the *HOME* environment variable is set to a non-null and non-empty value, *directory* shall default to the value named in the *HOME* environment variable. If the *HOME* environment variable is empty or is undefined, the default value of *directory* is implementation-defined.

If no ' ! ' is appended to the command name, and the edit buffer has been modified since the last complete write, and the current pathname does not begin with a ' / ', it shall be an error.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Copy**

*Synopsis:*     [2addr] co[py] laddr [flags]  
              [2addr] t laddr [flags]

Copy the specified lines after the specified destination line; line zero specifies that the lines shall be placed at the beginning of the edit buffer.

*Current line:* Set to the last line copied.

*Current column:* Set to non-<blank>.

**Delete**

*Synopsis:*     `[2addr] d[delete][buffer][count][flags]`

Delete the specified lines into a buffer (defaulting to the unnamed buffer), which shall become a line-mode buffer.

Flags can immediately follow the command name; see [Command Line Parsing in ex](#) (on page 2694).

*Current line:* Set to the line following the deleted lines, or to the last line in the edit buffer if that line is past the end of the edit buffer, or to zero if the edit buffer is empty.

*Current column:* Set to non-<blank>.

**Edit**

*Synopsis:*     `e[dit][!][+command][file]`  
               `ex[!][+command][file]`

If no `'!'` is appended to the command name, and the edit buffer has been modified since the last complete write, it shall be an error.

If *file* is specified, replace the current contents of the edit buffer with the current contents of *file*, and set the current pathname to *file*. If *file* is not specified, replace the current contents of the edit buffer with the current contents of the file named by the current pathname. If for any reason the current contents of the file cannot be accessed, the edit buffer shall be empty.

The `+command` option shall be <blank>-delimited; <blank> characters within the `+command` can be escaped by preceding them with a <backslash> character. The `+command` shall be interpreted as an *ex* command immediately after the contents of the edit buffer have been replaced and the current line and column have been set.

If the edit buffer is empty:

*Current line:* Set to 0.

*Current column:* Set to 1.

Otherwise, if executed while in *ex* command mode or if the `+command` argument is specified:

*Current line:* Set to the last line of the edit buffer.

*Current column:* Set to non-<blank>.

Otherwise, if *file* is omitted or results in the current pathname:

*Current line:* Set to the first line of the edit buffer.

*Current column:* Set to non-<blank>.

Otherwise, if *file* is the same as the last file edited, the line and column shall be set as follows; if the file was previously edited, the line and column may be set as follows:

*Current line:* Set to the last value held when that file was last edited. If this value is not a valid line in the new edit buffer, set to the first line of the edit buffer.

*Current column:* If the current line was set to the last value held when the file was last edited, set to the last value held when the file was last edited. Otherwise, or if the last value is not a valid column in the new edit buffer, set to non-<blank>.

Otherwise:

*Current line*: Set to the first line of the edit buffer.

*Current column*: Set to non-<blank>.

## File

*Synopsis:*     f[file][file]

If a *file* argument is specified, the alternate pathname shall be set to the current pathname, and the current pathname shall be set to *file*.

Write an informational message. If the file has a current pathname, it shall be included in this message; otherwise, the message shall indicate that there is no current pathname. If the edit buffer contains lines, the current line number and the number of lines in the edit buffer shall be included in this message; otherwise, the message shall indicate that the edit buffer is empty. If the edit buffer has been modified since the last complete write, this fact shall be included in this message. If the **readonly** edit option is set, this fact shall be included in this message. The message may contain other unspecified information.

*Current line*: Unchanged.

*Current column*: Unchanged.

## Global

*Synopsis:*     [2addr] g[lobal] /pattern/ [commands]  
                  [2addr] v /pattern/ [commands]

The optional '!' character after the **global** command shall be the same as executing the **v** command.

If *pattern* is empty (for example, " / ") or not specified, the last regular expression used in the editor command shall be used as the *pattern*. The *pattern* can be delimited by <slash> characters (shown in the Synopsis), as well as any non-alphanumeric or non-<blank> other than <backslash>, <vertical-line>, <newline>, or double-quote.

If no lines are specified, the lines shall default to the entire file.

The **global** and **v** commands are logically two-pass operations. First, mark the lines within the specified lines for which the line excluding the terminating <newline> matches (**global**) or does not match (**v** or **global!**) the specified pattern. Second, execute the *ex* commands given by *commands*, with the current line ( ' . ' ) set to each marked line. If an error occurs during this process, or the contents of the edit buffer are replaced (for example, by the *ex* :**edit** command) an error message shall be written and no more commands resulting from the execution of this command shall be processed.

Multiple *ex* commands can be specified by entering multiple commands on a single line using a <vertical-line> to delimit them, or one per line, by escaping each <newline> with a <backslash>.

If no commands are specified:

1. If in *ex* command mode, it shall be as if the **print** command were specified.
2. Otherwise, no command shall be executed.

For the **append**, **change**, and **insert** commands, the input text shall be included as part of the command, and the terminating <period> can be omitted if the command ends the list of commands. The **open** and **visual** commands can be specified as one of the commands, in which

case each marked line shall cause the editor to enter open or visual mode. If open or visual mode is exited using the *vi* **Q** command, the current line shall be set to the next marked line, and open or visual mode reentered, until the list of marked lines is exhausted.

The **global**, **v**, and **undo** commands cannot be used in *commands*. Marked lines may be deleted by commands executed for lines occurring earlier in the file than the marked lines. In this case, no commands shall be executed for the deleted lines.

If the remembered search direction is not set, the **global** and **v** commands shall set it to forward.

The **autoprint** and **autoindent** edit options shall be inhibited for the duration of the **g** or **v** command.

*Current line*: If no commands executed, set to the last marked line. Otherwise, as specified for the executed *ex* commands.

*Current column*: If no commands are executed, set to non-<blank>; otherwise, as specified for the individual *ex* commands.

## Insert

*Synopsis*:     [*laddr*] i[nsert][!]

Enter *ex* text input mode; the input text shall be placed before the specified line. If the line is zero or 1, the text shall be placed at the beginning of the edit buffer.

This command shall be affected by the **number** and **autoindent** edit options; following the command name with '!' shall cause the **autoindent** edit option setting to be toggled for the duration of this command only.

*Current line*: Set to the last input line; if no lines were input, set to the line before the specified line, or to the first line of the edit buffer if there are no lines preceding the specified line, or zero if the edit buffer is empty.

*Current column*: Set to non-<blank>.

## Join

*Synopsis*:     [*2addr*] j[oin][!][*count*][*flags*]

If *count* is specified:

If no address was specified, the **join** command shall behave as if *2addr* were the current line and the current line plus *count* (*.*, *.* + *count*).

If one address was specified, the **join** command shall behave as if *2addr* were the specified address and the specified address plus *count* (*addr*, *addr* + *count*).

If two addresses were specified, the **join** command shall behave as if an additional address, equal to the last address plus *count* -1 (*addr1*, *addr2*, *addr2* + *count* -1), was specified.

If this would result in a second address greater than the last line of the edit buffer, it shall be corrected to be equal to the last line of the edit buffer.

If no *count* is specified:

88498 If no address was specified, the **join** command shall behave as if *2addr* were the current  
 88499 line and the next line (*.*, *.* +1).

88500 If one address was specified, the **join** command shall behave as if *2addr* were the specified  
 88501 address and the next line (*addr*, *addr* +1).

88502 Join the text from the specified lines together into a single line, which shall replace the specified  
 88503 lines.

88504 If a '!' character is appended to the command name, the **join** shall be without modification of  
 88505 any line, independent of the current locale.

88506 Otherwise, in the POSIX locale, set the current line to the first of the specified lines, and then, for  
 88507 each subsequent line, proceed as follows:

- 88508 1. Discard leading <space> characters from the line to be joined.
- 88509 2. If the line to be joined is now empty, delete it, and skip steps 3 through 5.
- 88510 3. If the current line ends in a <blank>, or the first character of the line to be joined is a ' ) ' character, join the lines without further modification.
- 88511 3. If the current line ends in a <blank>, or the first character of the line to be joined is a ' ) ' character, join the lines without further modification.
- 88512 4. If the last character of the current line is a ' . ' , join the lines with two <space> characters  
 88513 between them.
- 88514 5. Otherwise, join the lines with a single <space> between them.

88515 *Current line*: Set to the first line specified.

88516 *Current column*: Set to non-<blank>.

## 88517 List

88518 *Synopsis*: `[2addr] l[ist][count][flags]`

88519 This command shall be equivalent to the *ex* command:

88520 `[2addr] p[rint][count] l[flags]`

88521 See [Print](#) (on page 2711).

## 88522 Map

88523 *Synopsis*: `map[!][lhs rhs]`

88524 If *lhs* and *rhs* are not specified:

- 88525 1. If '!' is specified, write the current list of text input mode maps.
- 88526 2. Otherwise, write the current list of command mode maps.
- 88527 3. Do nothing more.

88528 Implementations may restrict the set of characters accepted in *lhs* or *rhs*, except that printable  
 88529 characters and <blank> characters shall not be restricted. Additional restrictions shall be  
 88530 implementation-defined. In both *lhs* and *rhs*, any character can be escaped with a <control>-V, in  
 88531 which case the character shall not be used to delimit *lhs* from *rhs*, and the escaping <control>-V  
 88532 shall be discarded.

88533 If the character '!' is appended to the **map** command name, the mapping shall be effective  
 88534 during open or visual text input mode rather than **open** or **visual** command mode. This allows  
 88535 *lhs* to have two different **map** definitions at the same time: one for command mode and one for



88536 text input mode.

88537 For command mode mappings:

88538 When the *lhs* is entered as any part of a *vi* command in open or visual mode (but not as  
88539 part of the arguments to the command), the action shall be as if the corresponding *rhs* had  
88540 been entered.

88541 If any character in the command, other than the first, is escaped using a <control>-V  
88542 character, that character shall not be part of a match to an *lhs*.

88543 It is unspecified whether implementations shall support **map** commands where the *lhs* is  
88544 more than a single character in length, where the first character of the *lhs* is printable.

88545 If *lhs* contains more than one character and the first character is '#', followed by a  
88546 sequence of digits corresponding to a numbered function key, then when this function key  
88547 is typed it shall be mapped to *rhs*. Characters other than digits following a '#' character  
88548 also represent the function key named by the characters in the *lhs* following the '#' and  
88549 may be mapped to *rhs*. It is unspecified how function keys are named or what function  
88550 keys are supported.

88551 For text input mode mappings:

88552 When the *lhs* is entered as any part of text entered in open or visual text input modes, the  
88553 action shall be as if the corresponding *rhs* had been entered.

88554 If any character in the input text is escaped using a <control>-V character, that character  
88555 shall not be part of a match to an *lhs*.

88556 It is unspecified whether the *lhs* text entered for subsequent **map** or **unmap** commands is  
88557 replaced with the *rhs* text for the purposes of the screen display; regardless of whether or  
88558 not the display appears as if the corresponding *rhs* text was entered, the effect of the  
88559 command shall be as if the *lhs* text was entered.

88560 If only part of the *lhs* is entered, it is unspecified how long the editor will wait for additional,  
88561 possibly matching characters before treating the already entered characters as not matching the  
88562 *lhs*.

88563 The *rhs* characters shall themselves be subject to remapping, unless otherwise specified by the  
88564 **remap** edit option, except that if the characters in *lhs* occur as prefix characters in *rhs*, those  
88565 characters shall not be remapped.

88566 On block-mode terminals, the mapping need not occur immediately (for example, it may occur  
88567 after the terminal transmits a group of characters to the system), but it shall achieve the same  
88568 results as if it occurred immediately.

88569 *Current line*: Unchanged.

88570 *Current column*: Unchanged.



**Mark**

*Synopsis:*     [*laddr*] ma[*rk*] *character*  
                   [*laddr*] *k character*

Implementations shall support *character* values of a single lowercase letter of the POSIX locale and the backquote and single-quote characters; support of other characters is implementation-defined.

If executing the *vi m* command, set the specified mark to the current line and 1-based numbered character referenced by the current column, if any; otherwise, column position 1.

Otherwise, set the specified mark to the specified line and 1-based numbered first non-<blank> non-<newline> in the line, if any; otherwise, the last non-<newline> in the line, if any; otherwise, column position 1.

The mark shall remain associated with the line until the mark is reset or the line is deleted. If a deleted line is restored by a subsequent **undo** command, any marks previously associated with the line, which have not been reset, shall be restored as well. Any use of a mark not associated with a current line in the edit buffer shall be an error.

The marks ' and ' shall be set as described previously, immediately before the following events occur in the editor:

1. The use of ' \$ ' as an *ex* address
2. The use of a positive decimal number as an *ex* address
3. The use of a search command as an *ex* address
4. The use of a mark reference as an *ex* address
5. The use of the following open and visual mode commands: <control>-], %, (, ), [, ], {, }
6. The use of the following open and visual mode commands: ', **G**, **H**, **L**, **M**, **z** if the current line will change as a result of the command
7. The use of the open and visual mode commands: /, ?, **N**, ', **n** if the current line or column will change as a result of the command
8. The use of the *ex* mode commands: **z**, **undo**, **global**, **v**

For rules 1., 2., 3., and 4., the ' and ' marks shall not be set if the *ex* command is parsed as specified by rule 6.a. in [Command Line Parsing in ex](#) (on page 2694).

For rules 5., 6., and 7., the ' and ' marks shall not be set if the commands are used as motion commands in open and visual mode.

For rules 1., 2., 3., 4., 5., 6., 7., and 8., the ' and ' marks shall not be set if the command fails.

The ' and ' marks shall be set as described previously, each time the contents of the edit buffer are replaced (including the editing of the initial buffer), if in open or visual mode, or if in **ex** mode and the edit buffer is not empty, before any commands or movements (including commands or movements specified by the **-c** or **-t** options or the **+command** argument) are executed on the edit buffer. If in open or visual mode, the marks shall be set as if executing the *vi m* command; otherwise, as if executing the *ex mark* command.

When changing from **ex** mode to open or visual mode, if the ' and ' marks are not already set, the ' and ' marks shall be set as described previously.

*Current line:* Unchanged.

88612 *Current column*: Unchanged.

## 88613 **Move**

88614 *Synopsis*:     `[2addr] m[ove] 1addr [flags]`

88615 Move the specified lines after the specified destination line. A destination of line zero specifies  
88616 that the lines shall be placed at the beginning of the edit buffer. It shall be an error if the  
88617 destination line is within the range of lines to be moved.

88618 *Current line*: Set to the last of the moved lines.

88619 *Current column*: Set to non-<blank>.

## 88620 **Next**

88621 *Synopsis*:     `n[ext][!][+command][file ...]`

88622 If no '!' is appended to the command name, and the edit buffer has been modified since the  
88623 last complete write, it shall be an error, unless the file is successfully written as specified by the  
88624 **autowrite** option.

88625 If one or more files is specified:

- 88626     1. Set the argument list to the specified filenames.
- 88627     2. Set the current argument list reference to be the first entry in the argument list.
- 88628     3. Set the current pathname to the first filename specified.

88629 Otherwise:

- 88630     1. It shall be an error if there are no more filenames in the argument list after the filename  
88631       currently referenced.
- 88632     2. Set the current pathname and the current argument list reference to the filename after the  
88633       filename currently referenced in the argument list.

88634 Replace the contents of the edit buffer with the contents of the file named by the current  
88635 pathname. If for any reason the contents of the file cannot be accessed, the edit buffer shall be  
88636 empty.

88637 This command shall be affected by the **autowrite** and **writeany** edit options.

88638 The *+command* option shall be <blank>-delimited; <blank> characters can be escaped by  
88639 preceding them with a <backslash> character. The *+command* shall be interpreted as an *ex*  
88640 command immediately after the contents of the edit buffer have been replaced and the current  
88641 line and column have been set.

88642 *Current line*: Set as described for the **edit** command.

88643 *Current column*: Set as described for the **edit** command.

**Number**

*Synopsis:*    [2addr] nu[mber][count][flags]  
               [2addr] #[count][flags]

These commands shall be equivalent to the *ex* command:

[2addr] p[rint][count] #[flags]

See [Print](#).

**Open**

*Synopsis:*    [1addr] o[pen] /pattern/ [flags]

This command need not be supported on block-mode terminals or terminals with insufficient capabilities. If standard input, standard output, or standard error are not terminal devices, the results are unspecified.

Enter open mode.

The trailing delimiter can be omitted from *pattern* at the end of the command line. If *pattern* is empty (for example, " / / ") or not specified, the last regular expression used in the editor shall be used as the pattern. The pattern can be delimited by <slash> characters (shown in the Synopsis), as well as any alphanumeric, or non-<blank> other than <backslash>, <vertical-line>, <newline>, or double-quote.

*Current line:* Set to the specified line.

*Current column:* Set to non-<blank>.

**Preserve**

*Synopsis:*    pre[serve]

Save the edit buffer in a form that can later be recovered by using the **-r** option or by using the *ex* **recover** command. After the file has been preserved, a mail message shall be sent to the user. This message shall be readable by invoking the *mailx* utility. The message shall contain the name of the file, the time of preservation, and an *ex* command that could be used to recover the file. Additional information may be included in the mail message.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Print**

*Synopsis:*    [2addr] p[rint][count][flags]

Write the addressed lines. The behavior is unspecified if the number of columns on the display is less than the number of columns required to write any single character in the lines being written.

Non-printable characters, except for the <tab>, shall be written as implementation-defined multi-character sequences.

If the # flag is specified or the **number** edit option is set, each line shall be preceded by its line number in the following format:

"%6dΔΔ", <line number>

If the l flag is specified or the **list** edit option is set:

1. The characters listed in XBD Table 5-1 (on page 121) shall be written as the corresponding escape sequence.
2. Non-printable characters not in XBD Table 5-1 (on page 121) shall be written as one three-digit octal number (with a preceding <backslash>) for each byte in the character (most significant byte first).
3. The end of each line shall be marked with a '\$', and literal '\$' characters within the line shall be written with a preceding <backslash>.

Long lines shall be folded; the length at which folding occurs is unspecified, but should be appropriate for the output terminal, considering the number of columns of the terminal.

If a line is folded, and the **l** flag is not specified and the **list** edit option is not set, it is unspecified whether a multi-column character at the folding position is separated; it shall not be discarded.

*Current line*: Set to the last written line.

*Current column*: Unchanged if the current line is unchanged; otherwise, set to non-<blank>.

## Put

*Synopsis*:     [*laddr*] pu[t][*buffer*]

Append text from the specified buffer (by default, the unnamed buffer) to the specified line; line zero specifies that the text shall be placed at the beginning of the edit buffer. Each portion of a line in the buffer shall become a new line in the edit buffer, regardless of the mode of the buffer.

*Current line*: Set to the last line entered into the edit buffer.

*Current column*: Set to non-<blank>.

## Quit

*Synopsis*:     q[uit][!]

If no '!' is appended to the command name:

1. If the edit buffer has been modified since the last complete write, it shall be an error.
2. If there are filenames in the argument list after the filename currently referenced, and the last command was not a **quit**, **wq**, **xit**, or **ZZ** (see [Exit](#), on page 3399) command, it shall be an error.

Otherwise, terminate the editing session.

## Read

*Synopsis*:     [*laddr*] r[ead][!][*file*]

If '!' is not the first non-<blank> to follow the command name, a copy of the specified file shall be appended into the edit buffer after the specified line; line zero specifies that the copy shall be placed at the beginning of the edit buffer. The number of lines and bytes read shall be written. If no *file* is named, the current pathname shall be the default. If there is no current pathname, then *file* shall become the current pathname. If there is no current pathname or *file* operand, it shall be an error. Specifying a *file* that is not of type regular shall have unspecified results.

Otherwise, if *file* is preceded by '!', the rest of the line after the '!' shall have '%', '#', and '!' characters expanded as described in [Command Line Parsing in ex](#) (on page 2694).

The *ex* utility shall then pass two arguments to the program named by the shell edit option; the

first shall be **-c** and the second shall be the expanded arguments to the **read** command as a single argument. The standard input of the program shall be set to the standard input of the *ex* program when it was invoked. The standard error and standard output of the program shall be appended into the edit buffer after the specified line.

Each line in the copied file or program output (as delimited by <newline> characters or the end of the file or output if it is not immediately preceded by a <newline>), shall be a separate line in the edit buffer. Any occurrences of <carriage-return> and <newline> pairs in the output shall be treated as single <newline> characters.

The special meaning of the '!' following the **read** command can be overridden by escaping it with a <backslash> character.

*Current line*: If no lines are added to the edit buffer, unchanged. Otherwise, if in open or visual mode, set to the first line entered into the edit buffer. Otherwise, set to the last line entered into the edit buffer.

*Current column*: Set to non-<blank>.

## Recover

*Synopsis:*     **rec[over][!]** *file*

If no '!' is appended to the command name, and the edit buffer has been modified since the last complete write, it shall be an error.

If no *file* operand is specified, then the current pathname shall be used. If there is no current pathname or *file* operand, it shall be an error.

If no recovery information has previously been saved about *file*, the **recover** command shall behave identically to the **edit** command, and an informational message to this effect shall be written.

Otherwise, set the current pathname to *file*, and replace the current contents of the edit buffer with the recovered contents of *file*. If there are multiple instances of the file to be recovered, the one most recently saved shall be recovered, and an informational message that there are previous versions of the file that can be recovered shall be written. The editor shall behave as if the contents of the edit buffer have already been modified.

*Current file*: Set as described for the **edit** command.

*Current column*: Set as described for the **edit** command.

## Rewind

*Synopsis:*     **rew[ind][!]**

If no '!' is appended to the command name, and the edit buffer has been modified since the last complete write, it shall be an error, unless the file is successfully written as specified by the **autowrite** option.

If the argument list is empty, it shall be an error.

The current argument list reference and the current pathname shall be set to the first filename in the argument list.

Replace the contents of the edit buffer with the contents of the file named by the current pathname. If for any reason the contents of the file cannot be accessed, the edit buffer shall be empty.

This command shall be affected by the **autowrite** and **writeln** edit options.

*Current line*: Set as described for the **edit** command.

*Current column*: Set as described for the **edit** command.

## Set

*Synopsis*:     `se[t][option[=value]] ...][nooption ...][option? ...][all]`

When no arguments are specified, write the value of the **term** edit option and those options whose values have been changed from the default settings; when the argument *all* is specified, write all of the option values.

Giving an option name followed by the character '?' shall cause the current value of that option to be written. The '?' can be separated from the option name by zero or more <blank> characters. The '?' shall be necessary only for Boolean valued options. Boolean options can be given values by the form **set option** to turn them on or **set nooption** to turn them off; string and numeric options can be assigned by the form **set option=value**. Any <blank> characters in strings can be included as is by preceding each <blank> with an escaping <backslash>. More than one option can be set or listed by a single set command by specifying multiple arguments, each separated from the next by one or more <blank> characters.

See [Edit Options in ex](#) (on page 2724) for details about specific options.

*Current line*: Unchanged.

*Current column*: Unchanged.

## Shell

*Synopsis*:     `sh[ell]`

Invoke the program named in the **shell** edit option with the single argument **-i** (interactive mode). Editing shall be resumed when the program exits.

*Current line*: Unchanged.

*Current column*: Unchanged.

## Source

*Synopsis*:     `so[urce] file`

Read and execute *ex* commands from *file*. Lines in the file that are blank lines shall be ignored.

*Current line*: As specified for the individual *ex* commands.

*Current column*: As specified for the individual *ex* commands.

## Substitute

*Synopsis*:     `[2addr] s[ubstitute][/pattern/repl][options][count][flags]`

`[2addr] &[options][count][flags]`

`[2addr] ~[options][count][flags]`

Replace the first instance of the pattern *pattern* by the string *repl* on each specified line. (See [Regular Expressions in ex](#) (on page 2723) and [Replacement Strings in ex](#) (on page 2724).) Any non-alphabetic, non-<blank> delimiter other than <backslash>, '&|', <newline>, or double-quote can be used instead of '/'. <backslash> characters can be used to escape delimiters, <backslash> characters, and other special characters.

88801 The trailing delimiter can be omitted from *pattern* or from *repl* at the end of the command line. If  
88802 both *pattern* and *repl* are not specified or are empty (for example, `"/ /"`), the last `s` command  
88803 shall be repeated. If only *pattern* is not specified or is empty, the last regular expression used in  
88804 the editor shall be used as the pattern. If only *repl* is not specified or is empty, the pattern shall be  
88805 replaced by nothing. If the entire replacement pattern is `' % '`, the last replacement pattern to an  
88806 `s` command shall be used.

88807 Entering a <carriage-return> in *repl* (which requires an escaping <backslash> in *ex* mode and an  
88808 escaping <control>-V in open or *vi* mode) shall split the line at that point, creating a new line in  
88809 the edit buffer. The <carriage-return> shall be discarded.

88810 If *options* includes the letter `'g'` (**global**), all non-overlapping instances of the pattern in the line  
88811 shall be replaced.

88812 If *options* includes the letter `'c'` (**confirm**), then before each substitution the line shall be written;  
88813 the written line shall reflect all previous substitutions. On the following line, <space> characters  
88814 shall be written beneath the characters from the line that are before the *pattern* to be replaced,  
88815 and `'^'` characters written beneath the characters included in the *pattern* to be replaced. The *ex*  
88816 utility shall then wait for a response from the user. An affirmative response shall cause the  
88817 substitution to be done, while any other input shall not make the substitution. An affirmative  
88818 response shall consist of a line with the affirmative response (as defined by the current locale) at  
88819 the beginning of the line. This line shall be subject to editing in the same way as the *ex* command  
88820 line.

88821 If interrupted (see the ASYNCHRONOUS EVENTS section), any modifications confirmed by the  
88822 user shall be preserved in the edit buffer after the interrupt.

88823 If the remembered search direction is not set, the `s` command shall set it to forward.

88824 In the second Synopsis, the `&` command shall repeat the previous substitution, as if the `&`  
88825 command were replaced by:

88826 `s/pattern/repl/`

88827 where *pattern* and *repl* are as specified in the previous `s`, `&`, or `~` command.

88828 In the third Synopsis, the `~` command shall repeat the previous substitution, as if the `'~'` were  
88829 replaced by:

88830 `s/pattern/repl/`

88831 where *pattern* shall be the last regular expression specified to the editor, and *repl* shall be from  
88832 the previous substitution (including `&` and `~`) command.

88833 These commands shall be affected by the `LC_MESSAGES` environment variable.

88834 *Current line*: Set to the last line in which a substitution occurred, or, unchanged if no substitution  
88835 occurred.

88836 *Current column*: Set to non-<blank>.



**Suspend**

*Synopsis:*     su[suspend][!]  
                 st[op][!]

Allow control to return to the invoking process; *ex* shall suspend itself as if it had received the SIGTSTP signal. The suspension shall occur only if job control is enabled in the invoking shell (see the description of *set -m*).

These commands shall be affected by the **autowrite** and **writeany** edit options.

The current **susp** character (see *stty*) shall be equivalent to the **suspend** command.

**Tag**

*Synopsis:*     ta[g][!] tagstring

The results are unspecified if the format of a tags file is not as specified by the *ctags* utility (see *ctags*) description.

The **tag** command shall search for *tagstring* in the tag files referred to by the **tag** edit option, in the order they are specified, until a reference to *tagstring* is found. Files shall be searched from beginning to end. If no reference is found, it shall be an error and an error message to this effect shall be written. If the reference is not found, or if an error occurs while processing a file referred to in the **tag** edit option, it shall be an error, and an error message shall be written at the first occurrence of such an error.

Otherwise, if the tags file contained a pattern, the pattern shall be treated as a regular expression used in the editor; for example, for the purposes of the **s** command.

If the *tagstring* is in a file with a different name than the current pathname, set the current pathname to the name of that file, and replace the contents of the edit buffer with the contents of that file. In this case, if no '!' is appended to the command name, and the edit buffer has been modified since the last complete write, it shall be an error, unless the file is successfully written as specified by the **autowrite** option.

This command shall be affected by the **autowrite**, **tag**, **taglength**, and **writeany** edit options.

*Current line:* If the tags file contained a line number, set to that line number. If the line number is larger than the last line in the edit buffer, an error message shall be written and the current line shall be set as specified for the **edit** command.

If the tags file contained a pattern, set to the first occurrence of the pattern in the file. If no matching pattern is found, an error message shall be written and the current line shall be set as specified for the **edit** command.

*Current column:* If the tags file contained a line-number reference and that line-number was not larger than the last line in the edit buffer, or if the tags file contained a pattern and that pattern was found, set to non-<blank>. Otherwise, set as specified for the **edit** command.



**Unabbreviate**

*Synopsis:*     una[bbrev] lhs

If *lhs* is not an entry in the current list of abbreviations (see [Abbreviate](#), on page 2702), it shall be an error. Otherwise, delete *lhs* from the list of abbreviations.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Undo**

*Synopsis:*     u[ndo]

Reverse the changes made by the last command that modified the contents of the edit buffer, including **undo**. For this purpose, the **global**, **v**, **open**, and **visual** commands, and commands resulting from buffer executions and mapped character expansions, are considered single commands.

If no action that can be undone preceded the **undo** command, it shall be an error.

If the **undo** command restores lines that were marked, the mark shall also be restored unless it was reset subsequent to the deletion of the lines.

*Current line:*

1. If lines are added or changed in the file, set to the first line added or changed.
2. Set to the line before the first line deleted, if it exists.
3. Set to 1 if the edit buffer is not empty.
4. Set to zero.

*Current column:* Set to non-<blank>.

**Unmap**

*Synopsis:*     unm[ap][!] lhs

If '!' is appended to the command name, and if *lhs* is not an entry in the list of text input mode map definitions, it shall be an error. Otherwise, delete *lhs* from the list of text input mode map definitions.

If no '!' is appended to the command name, and if *lhs* is not an entry in the list of command mode map definitions, it shall be an error. Otherwise, delete *lhs* from the list of command mode map definitions.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Version**

*Synopsis:*     `ve[rsion]`

Write a message containing version information for the editor. The format of the message is unspecified.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Visual**

*Synopsis:*     `[laddr] vi[sual][type][count][flags]`

If *ex* is currently in open or visual mode, the Synopsis and behavior of the visual command shall be the same as the **edit** command, as specified by **Edit** (on page 2704).

Otherwise, this command need not be supported on block-mode terminals or terminals with insufficient capabilities. If standard input, standard output, or standard error are not terminal devices, the results are unspecified.

If *count* is specified, the value of the **window** edit option shall be set to *count* (as described in **window**, on page 2731). If the '^' type character was also specified, the **window** edit option shall be set before being used by the type character.

Enter visual mode. If *type* is not specified, it shall be as if a *type* of '+' was specified. The *type* shall cause the following effects:

- + Place the beginning of the specified line at the top of the display.
- Place the end of the specified line at the bottom of the display.
- . Place the beginning of the specified line in the middle of the display.
- ^ If the specified line is less than or equal to the value of the **window** edit option, set the line to 1; otherwise, decrement the line by the value of the **window** edit option minus 1. Place the beginning of this line as close to the bottom of the displayed lines as possible, while still displaying the value of the **window** edit option number of lines.

*Current line:* Set to the specified line.

*Current column:* Set to non-<blank>.

**Write**

*Synopsis:*     `[2addr] w[rite][!][>][file]`  
                   `[2addr] w[rite][!][file]`  
                   `[2addr] wq[!][>][file]`

If no lines are specified, the lines shall default to the entire file.

The command **wq** shall be equivalent to a **write** command followed by a **quit** command; **wq!** shall be equivalent to **write!** followed by **quit**. In both cases, if the **write** command fails, the **quit** shall not be attempted.

If the command name is not followed by one or more <blank> characters, or *file* is not preceded by a '!' character, the **write** shall be to a file.

1. If the >> argument is specified, and the file already exists, the lines shall be appended to the file instead of replacing its contents. If the >> argument is specified, and the file does not already exist, it is unspecified whether the write shall proceed as if the >> argument

- 88943 had not been specified or if the **write** shall fail.
- 88944 2. If the **readonly** edit option is set (see **readonly**, on page 2728), the **write** shall fail.
- 88945 3. If *file* is specified, and is not the current pathname, and the file exists, the **write** shall fail.
- 88946 4. If *file* is not specified, the current pathname shall be used. If there is no current pathname,
- 88947 the **write** command shall fail.
- 88948 5. If the current pathname is used, and the current pathname has been changed by the **file**
- 88949 or **read** commands, and the file exists, the **write** shall fail. If the **write** is successful,
- 88950 subsequent **writes** shall not fail for this reason (unless the current pathname is changed
- 88951 again).
- 88952 6. If the whole edit buffer is not being written, and the file to be written exists, the **write**
- 88953 shall fail.

88954 For rules 1., 2., 3., and 5., the **write** can be forced by appending the character '**!**' to the

88955 command name.

88956 For rules 2., 3., and 5., the **write** can be forced by setting the **writeany** edit option.

88957 Additional, implementation-defined tests may cause the **write** to fail.

88958 If the edit buffer is empty, a file without any contents shall be written.

88959 An informational message shall be written noting the number of lines and bytes written.

88960 Otherwise, if the command is followed by one or more <blank> characters, and the file is

88961 preceded by '**!**', the rest of the line after the '**!**' shall have '%', '#', and '**!**' characters

88962 expanded as described in **Command Line Parsing in ex** (on page 2694).

88963 The *ex* utility shall then pass two arguments to the program named by the **shell** edit option; the

88964 first shall be **-c** and the second shall be the expanded arguments to the **write** command as a

88965 single argument. The specified lines shall be written to the standard input of the command. The

88966 standard error and standard output of the program, if any, shall be written as described for the

88967 **print** command. If the last character in that output is not a <newline>, a <newline> shall be

88968 written at the end of the output.

88969 The special meaning of the '**!**' following the **write** command can be overridden by escaping it

88970 with a <backslash> character.

88971 *Current line*: Unchanged.

88972 *Current column*: Unchanged.

## 88973 **Write and Exit**

88974 *Synopsis:*     [2addr] x[it][!][file]

88975 If the edit buffer has not been modified since the last complete **write**, **xit** shall be equivalent to

88976 the **quit** command, or if a '**!**' is appended to the command name, to **quit!**.

88977 Otherwise, **xit** shall be equivalent to the **wq** command, or if a '**!**' is appended to the command

88978 name, to **wq!**.

88979 *Current line*: Unchanged.

88980 *Current column*: Unchanged.

**Yank**

*Synopsis:*     `[2addr] ya[nk][buffer][count]`

Copy the specified lines to the specified buffer (by default, the unnamed buffer), which shall become a line-mode buffer.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Adjust Window**

*Synopsis:*     `[1addr] z[!][type ...][count][flags]`

If no line is specified, the current line shall be the default; if *type* is omitted as well, the current line value shall first be incremented by 1. If incrementing the current line would cause it to be greater than the last line in the edit buffer, it shall be an error.

If there are <blank> characters between the *type* argument and the preceding *z* command name or optional '!' character, it shall be an error.

If *count* is specified, the value of the **window** edit option shall be set to *count* (as described in **window**, on page 2731). If *count* is omitted, it shall default to 2 times the value of the **scroll** edit option, or if ! was specified, the number of lines in the display minus 1.

If *type* is omitted, then *count* lines starting with the specified line shall be written. Otherwise, *count* lines starting with the line specified by the *type* argument shall be written.

The *type* argument shall change the lines to be written. The possible values of *type* are as follows:

- The specified line shall be decremented by the following value:

$((\text{number of ``-'' characters}) \times \text{count}) - 1$

If the calculation would result in a number less than 1, it shall be an error. Write lines from the edit buffer, starting at the new value of line, until *count* lines or the last line in the edit buffer has been written.

- + The specified line shall be incremented by the following value:

$((\text{number of ``+'' characters}) - 1) \times \text{count} + 1$

If the calculation would result in a number greater than the last line in the edit buffer, it shall be an error. Write lines from the edit buffer, starting at the new value of line, until *count* lines or the last line in the edit buffer has been written.

**=,.** If more than a single ' .' or '=' is specified, it shall be an error. The following steps shall be taken:

1. If *count* is zero, nothing shall be written.
2. Write as many of the *N* lines before the current line in the edit buffer as exist. If *count* or '!' was specified, *N* shall be:

$(\text{count} - 1) / 2$

Otherwise, *N* shall be:

$(\text{count} - 3) / 2$

If *N* is a number less than 3, no lines shall be written.

- 89019 3. If '=' was specified as the type character, write a line consisting of the smaller of the  
89020 number of columns in the display divided by two, or 40 '-' characters.
- 89021 4. Write the current line.
- 89022 5. Repeat step 3.
- 89023 6. Write as many of the *N* lines after the current line in the edit buffer as exist. *N* shall  
89024 be defined as in step 2. If *N* is a number less than 3, no lines shall be written. If *count*  
89025 is less than 3, no lines shall be written.

89026 ^ The specified line shall be decremented by the following value:

89027  $((\text{number of ``^`` characters}) + 1) \times \text{count} - 1$

89028 If the calculation would result in a number less than 1, it shall be an error. Write lines from  
89029 the edit buffer, starting at the new value of line, until *count* lines or the last line in the edit  
89030 buffer has been written.

89031 *Current line*: Set to the last line written, unless the type is =, in which case, set to the specified  
89032 line.

89033 *Current column*: Set to non-<blank>.

## 89034 Escape

89035 *Synopsis:*       ! *command*  
89036               [*addr*]! *command*

89037 The contents of the line after the '!' shall have '%', '#', and '!' characters expanded as  
89038 described in [Command Line Parsing in ex](#) (on page 2694). If the expansion causes the text of the  
89039 line to change, it shall be redisplayed, preceded by a single '!' character.

89040 The *ex* utility shall execute the program named by the **shell** edit option. It shall pass two  
89041 arguments to the program; the first shall be -c, and the second shall be the expanded arguments  
89042 to the ! command as a single argument.

89043 If no lines are specified, the standard input, standard output, and standard error of the program  
89044 shall be set to the standard input, standard output, and standard error of the *ex* program when it  
89045 was invoked. In addition, a warning message shall be written if the edit buffer has been  
89046 modified since the last complete write, and the **warn** edit option is set.

89047 If lines are specified, they shall be passed to the program as standard input, and the standard  
89048 output and standard error of the program shall replace those lines in the edit buffer. Each line in  
89049 the program output (as delimited by <newline> characters or the end of the output if it is not  
89050 immediately preceded by a <newline>), shall be a separate line in the edit buffer. Any  
89051 occurrences of <carriage-return> and <newline> pairs in the output shall be treated as single  
89052 <newline> characters. The specified lines shall be copied into the unnamed buffer before they  
89053 are replaced, and the unnamed buffer shall become a line-mode buffer.

89054 If in *ex* mode, a single '!' character shall be written when the program completes.

89055 This command shall be affected by the **shell** and **warn** edit options. If no lines are specified, this  
89056 command shall be affected by the **autowrite** and **writany** edit options. If lines are specified, this  
89057 command shall be affected by the **autoprint** edit option.

Current line:

1. If no lines are specified, unchanged.
2. Otherwise, set to the last line read in, if any lines are read in.
3. Otherwise, set to the line before the first line of the lines specified, if that line exists.
4. Otherwise, set to the first line of the edit buffer if the edit buffer is not empty.
5. Otherwise, set to zero.

Current column: If no lines are specified, unchanged. Otherwise, set to non-<blank>.

### Shift Left

Synopsis: `[2addr] <[< ...][count][flags]`

Shift the specified lines to the start of the line; the number of column positions to be shifted shall be the number of command characters times the value of the **shiftwidth** edit option. Only leading <blank> characters shall be deleted or changed into other <blank> characters in shifting; other characters shall not be affected.

Lines to be shifted shall be copied into the unnamed buffer, which shall become a line-mode buffer.

This command shall be affected by the **autoprint** edit option.

Current line: Set to the last line in the lines specified.

Current column: Set to non-<blank>.

### Shift Right

Synopsis: `[2addr] >[> ...][count][flags]`

Shift the specified lines away from the start of the line; the number of column positions to be shifted shall be the number of command characters times the value of the **shiftwidth** edit option. The shift shall be accomplished by adding <blank> characters as a prefix to the line or changing leading <blank> characters into other <blank> characters. Empty lines shall not be changed.

Lines to be shifted shall be copied into the unnamed buffer, which shall become a line-mode buffer.

This command shall be affected by the **autoprint** edit option.

Current line: Set to the last line in the lines specified.

Current column: Set to non-<blank>.

### <control>-D

Synopsis: `<control>-D`

Write the next *n* lines, where *n* is the minimum of the values of the **scroll** edit option and the number of lines after the current line in the edit buffer. If the current line is the last line of the edit buffer it shall be an error.

Current line: Set to the last line written.

Current column: Set to non-<blank>.

**Write Line Number**

*Synopsis:*     `[laddr] = [flags]`

If *line* is not specified, it shall default to the last line in the edit buffer. Write the line number of the specified line.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Execute**

*Synopsis:*     `[2addr] @ buffer`

`[2addr] * buffer`

If no buffer is specified or is specified as '@' or '\*', the last buffer executed shall be used. If no previous buffer has been executed, it shall be an error.

For each line specified by the addresses, set the current line ('.') to the specified line, and execute the contents of the named *buffer* (as they were at the time the @ command was executed) as *ex* commands. For each line of a line-mode buffer, and all but the last line of a character-mode buffer, the *ex* command parser shall behave as if the line was terminated by a <newline>.

If an error occurs during this process, or a line specified by the addresses does not exist when the current line would be set to it, or more than a single line was specified by the addresses, and the contents of the edit buffer are replaced (for example, by the *ex* :edit command) an error message shall be written, and no more commands resulting from the execution of this command shall be processed.

*Current line:* As specified for the individual *ex* commands.

*Current column:* As specified for the individual *ex* commands.

**Regular Expressions in ex**

The *ex* utility shall support regular expressions that are a superset of the basic regular expressions described in XBD [Section 9.3](#) (on page 183). A null regular expression (" / ") shall be equivalent to the last regular expression encountered.

Regular expressions can be used in addresses to specify lines and, in some commands (for example, the **substitute** command), to specify portions of a line to be substituted.

The following constructs can be used to enhance the basic regular expressions:

\< Match the beginning of a *word*. (See the definition of *word* at the beginning of [Command Descriptions in ex](#) (on page 2699).)

\> Match the end of a *word*.

~ Match the replacement part of the last **substitute** command. The <tilde> ('~') character can be escaped in a regular expression to become a normal character with no special meaning. The <backslash> shall be discarded.

When the editor option **magic** is not set, the only characters with special meanings shall be '^' at the beginning of a pattern, '\$' at the end of a pattern, and <backslash>. The characters '.', '\*', '[', and '~' shall be treated as ordinary characters unless preceded by a <backslash>; when preceded by a <backslash> they shall regain their special meaning, or in the case of <backslash>, be handled as a single <backslash>. <backslash> characters used to escape other characters shall be discarded.



## Replacement Strings in ex

The character '&' ('&' if the editor option **magic** is not set) in the replacement string shall stand for the text matched by the pattern to be replaced. The character '~' ('\~' if **magic** is not set) shall be replaced by the replacement part of the previous **substitute** command. The sequence '\n', where *n* is an integer, shall be replaced by the text matched by the corresponding back-reference expression. If the corresponding back-reference expression does not match, then the characters '\n' shall be replaced by the empty string.

The strings '\l', '\u', '\L', and '\U' can be used to modify the case of elements in the replacement string (using the '&' or "\"digit) notation. The string '\l' ('\u') shall cause the character that follows to be converted to lowercase (uppercase). The string '\L' ('\U') shall cause all characters subsequent to it to be converted to lowercase (uppercase) as they are inserted by the substitution until the string '\e' or '\E', or the end of the replacement string, is encountered.

Otherwise, any character following a <backslash> shall be treated as that literal character, and the escaping <backslash> shall be discarded.

An example of case conversion with the **s** command is as follows:

```
:p
The cat sat on the mat.
:s/\<.at\>/\u&/gp
The Cat Sat on the Mat.
:s/S\(.*\)M/S\U\1\eM/p
The Cat SAT ON THE Mat.
```

## Edit Options in ex

The *ex* utility has a number of options that modify its behavior. These options have default settings, which can be changed using the **set** command.

Options are Boolean unless otherwise specified.

### autoindent, ai

[Default *unset*]

If **autoindent** is set, each line in input mode shall be indented (using first as many <tab> characters as possible, as determined by the editor option **tabstop**, and then using <space> characters) to align with another line, as follows:

1. If in open or visual mode and the text input is part of a line-oriented command (see the EXTENDED DESCRIPTION in *vi*), align to the first column.
2. Otherwise, if in open or visual mode, indentation for each line shall be set as follows:
  - a. If a line was previously inserted as part of this command, it shall be set to the indentation of the last inserted line by default, or as otherwise specified for the <control>-D character in *Input Mode Commands* in *vi* (on page 3399).
  - b. Otherwise, it shall be set to the indentation of the previous current line, if any; otherwise, to the first column.
3. For the *ex* **a**, **i**, and **c** commands, indentation for each line shall be set as follows:



- 89176 a. If a line was previously inserted as part of this command, it shall be set to the
- 89177 indentation of the last inserted line by default, or as otherwise specified for the *eof*
- 89178 character in **Scroll** (on page 2698).
- 89179 b. Otherwise, if the command is the *ex a* command, it shall be set to the line
- 89180 appended after, if any; otherwise to the first column.
- 89181 c. Otherwise, if the command is the *ex i* command, it shall be set to the line inserted
- 89182 before, if any; otherwise to the first column.
- 89183 d. Otherwise, if the command is the *ex c* command, it shall be set to the indentation of
- 89184 the line replaced.

#### 89185 **autoprint, ap**

89186 [Default *set*]

89187 If **autoprint** is set, the current line shall be written after each *ex* command that modifies the  
 89188 contents of the current edit buffer, and after each **tag** command for which the tag search pattern  
 89189 was found or tag line number was valid, unless:

- 89190 1. The command was executed while in open or visual mode.
- 89191 2. The command was executed as part of a **global** or **v** command or **@** buffer execution.
- 89192 3. The command was the form of the **read** command that reads a file into the edit buffer.
- 89193 4. The command was the **append**, **change**, or **insert** command.
- 89194 5. The command was not terminated by a <newline>.
- 89195 6. The current line shall be written by a flag specified to the command; for example, **delete #**  
 89196 shall write the current line as specified for the flag modifier to the **delete** command, and  
 89197 not as specified by the **autoprint** edit option.

#### 89198 **autowrite, aw**

89199 [Default *unset*]

89200 If **autowrite** is set, and the edit buffer has been modified since it was last completely written to  
 89201 any file, the contents of the edit buffer shall be written as if the *ex write* command had been  
 89202 specified without arguments, before each command affected by the **autowrite** edit option is  
 89203 executed. Appending the character '!' to the command name of any of the *ex* commands  
 89204 except '!' shall prevent the write. If the write fails, it shall be an error and the command shall  
 89205 not be executed.

#### 89206 **beautify, bf**

89207 XSI [Default *unset*]

89208 If **beautify** is set, all non-printable characters, other than <tab>, <newline>, and <form-feed>  
 89209 characters, shall be discarded from text read in from files.

**89210 directory, dir**

89211 [Default *implementation-defined*]

89212 The value of this option specifies the directory in which the editor buffer is to be placed. If this  
89213 directory is not writable by the user, the editor shall quit.

**89214 edcompatible, ed**

89215 [Default *unset*]

89216 Causes the presence of **g** and **c** suffixes on substitute commands to be remembered, and toggled  
89217 by repeating the suffixes.

**89218 errorbells, eb**

89219 [Default *unset*]

89220 If the editor is in *ex* mode, and the terminal does not support a standout mode (such as inverse  
89221 video), and **errorbells** is set, error messages shall be preceded by alerting the terminal.

**89222 exrc**

89223 [Default *unset*]

89224 If **exrc** is set, *ex* shall access any **.exrc** file in the current directory, as described in [Initialization in](#)  
89225 [ex and vi](#) (on page 2690). If **exrc** is not set, *ex* shall ignore any **.exrc** file in the current directory  
89226 during initialization, unless the current directory is that named by the *HOME* environment  
89227 variable.

**89228 ignorecase, ic**

89229 [Default *unset*]

89230 If **ignorecase** is set, characters that have uppercase and lowercase representations shall have  
89231 those representations considered as equivalent for purposes of regular expression comparison.

89232 The **ignorecase** edit option shall affect all remembered regular expressions; for example,  
89233 unsetting the **ignorecase** edit option shall cause a subsequent *vi n* command to search for the  
89234 last basic regular expression in a case-sensitive fashion.

**89235 list**

89236 [Default *unset*]

89237 If **list** is set, edit buffer lines written while in *ex* command mode shall be written as specified for  
89238 the **print** command with the **l** flag specified. In open or visual mode, each edit buffer line shall  
89239 be displayed as specified for the *ex print* command with the **l** flag specified. In open or visual  
89240 text input mode, when the cursor does not rest on any character in the line, it shall rest on the  
89241 ' \$ ' marking the end of the line.

**magic**

[Default *set*]

If **magic** is set, modify the interpretation of characters in regular expressions and substitution replacement strings (see [Regular Expressions in ex](#) (on page 2723) and [Replacement Strings in ex](#), on page 2724).

**mesg**

[Default *set*]

If **mesg** is set, the permission for others to use the **write** or **talk** commands to write to the terminal shall be turned on while in open or visual mode. The shell-level command **mesg n** shall take precedence over any setting of the *ex* **mesg** option; that is, if **mesg y** was issued before the editor started (or in a shell escape), such as:

```
:!mesg y
```

the **mesg** option in *ex* shall suppress incoming messages, but the **mesg** option shall not enable incoming messages if **mesg n** was issued.

**number, nu**

[Default *unset*]

If **number** is set, edit buffer lines written while in *ex* command mode shall be written with line numbers, in the format specified by the **print** command with the **#** flag specified. In *ex* text input mode, each line shall be preceded by the line number it will have in the file.

In open or visual mode, each edit buffer line shall be displayed with a preceding line number, in the format specified by the *ex* **print** command with the **#** flag specified. This line number shall not be considered part of the line for the purposes of evaluating the current column; that is, column position 1 shall be the first column position after the format specified by the **print** command.

**paragraphs, para**

[Default in the POSIX locale `IPLPPPQPP LIpplpipbp`]

The **paragraphs** edit option shall define additional paragraph boundaries for the open and visual mode commands. The **paragraphs** edit option can be set to a character string consisting of zero or more character pairs. It shall be an error to set it to an odd number of characters.

**prompt**

[Default *set*]

If **prompt** is set, *ex* command mode input shall be prompted for with a <colon> (':'); when unset, no prompt shall be written.

**readonly**[Default *see text*]

If the **readonly** edit option is set, read-only mode shall be enabled (see **Write**, on page 2718). The **readonly** edit option shall be initialized to set if either of the following conditions are true:

- The command-line option **-R** was specified.
- Performing actions equivalent to the *access()* function called with the following arguments indicates that the file lacks write permission:
  1. The current pathname is used as the *path* argument.
  2. The constant **W\_OK** is used as the *amode* argument.

The **readonly** edit option may be initialized to set for other, implementation-defined reasons. The **readonly** edit option shall not be initialized to unset based on any special privileges of the user or process. The **readonly** edit option shall be reinitialized each time that the contents of the edit buffer are replaced (for example, by an **edit** or **next** command) unless the user has explicitly set it, in which case it shall remain set until the user explicitly unsets it. Once unset, it shall again be reinitialized each time that the contents of the edit buffer are replaced.

**redraw**[Default *unset*]

The editor simulates an intelligent terminal on a dumb terminal. (Since this is likely to require a large amount of output to the terminal, it is useful only at high transmission speeds.)

**remap**[Default *set*]

If **remap** is set, map translation shall allow for maps defined in terms of other maps; translation shall continue until a final product is obtained. If unset, only a one-step translation shall be done.

**report**

[Default 5]

The value of this **report** edit option specifies what number of lines being added, copied, deleted, or modified in the edit buffer will cause an informational message to be written to the user. The following conditions shall cause an informational message. The message shall contain the number of lines added, copied, deleted, or modified, but is otherwise unspecified.

- An *ex* or *vi* editor command, other than **open**, **undo**, or **visual**, that modifies at least the value of the **report** edit option number of lines, and which is not part of an *ex* **global** or **v** command, or *ex* or *vi* buffer execution, shall cause an informational message to be written.
- An *ex* **yank** or *vi* **y** or **Y** command, that copies at least the value of the **report** edit option plus 1 number of lines, and which is not part of an *ex* **global** or **v** command, or *ex* or *vi* buffer execution, shall cause an informational message to be written.
- An *ex* **global**, **v**, **open**, **undo**, or **visual** command or *ex* or *vi* buffer execution, that adds or deletes a total of at least the value of the **report** edit option number of lines, and which is not part of an *ex* **global** or **v** command, or *ex* or *vi* buffer execution, shall cause an informational message to be written. (For example, if 3 lines were added and 8 lines deleted during an *ex* **visual** command, 5 would be the number compared against the

89316 **report** edit option after the command completed.)

89317 **scroll, scr**

89318 [Default (number of lines in the display -1)/2]

89319 The value of the **scroll** edit option shall determine the number of lines scrolled by the *ex*  
89320 <control>-D and **z** commands. For the *vi* <control>-D and <control>-U commands, it shall be the  
89321 initial number of lines to scroll when no previous <control>-D or <control>-U command has  
89322 been executed.

89323 **sections**

89324 [Default in the POSIX locale `NHSHH HUnhsh`]

89325 The **sections** edit option shall define additional section boundaries for the open and visual mode  
89326 commands. The **sections** edit option can be set to a character string consisting of zero or more  
89327 character pairs; it shall be an error to set it to an odd number of characters.

89328 **shell, sh**

89329 [Default from the environment variable `SHELL`]

89330 The value of this option shall be a string. The default shall be taken from the `SHELL`  
89331 environment variable. If the `SHELL` environment variable is null or empty, the *sh* (see *sh*) utility  
89332 shall be the default.

89333 **shiftwidth, sw**

89334 [Default 8]

89335 The value of this option shall give the width in columns of an indentation level used during  
89336 autoindentation and by the shift commands (< and >).

89337 **showmatch, sm**

89338 [Default *unset*]

89339 The functionality described for the **showmatch** edit option need not be supported on block-  
89340 mode terminals or terminals with insufficient capabilities.

89341 If **showmatch** is set, in open or visual mode, when a ' ) ' or ' } ' is typed, if the matching ' ( ' or  
89342 ' { ' is currently visible on the display, the matching ' ( ' or ' { ' shall be flagged moving the  
89343 cursor to its location for an unspecified amount of time.

89344 **showmode**

89345 [Default *unset*]

89346 If **showmode** is set, in open or visual mode, the current mode that the editor is in shall be  
89347 displayed on the last line of the display. Command mode and text input mode shall be  
89348 differentiated; other unspecified modes and implementation-defined information may be  
89349 displayed.

**slowopen**

[Default *unset*]

If **slowopen** is set during open and visual text input modes, the editor shall not update portions of the display other than those display line columns that display the characters entered by the user (see [Input Mode Commands in vi](#), on page 3399).

**tabstop, ts**

[Default 8]

The value of this edit option shall specify the column boundary used by a <tab> in the display (see [autoprint, ap](#) (on page 2725) and [Input Mode Commands in vi](#), on page 3399).

**taglength, tl**

[Default zero]

The value of this edit option shall specify the maximum number of characters that are considered significant in the user-specified tag name and in the tag name from the tags file. If the value is zero, all characters in both tag names shall be significant.

**tags**

[Default *see text*]

The value of this edit option shall be a string of <blank>-delimited pathnames of files used by the **tag** command. The default value is unspecified.

**term**

[Default from the environment variable *TERM*]

The value of this edit option shall be a string. The default shall be taken from the *TERM* variable in the environment. If the *TERM* environment variable is empty or null, the default is unspecified. The editor shall use the value of this edit option to determine the type of the display device.

The results are unspecified if the user changes the value of the term edit option after editor initialization.

**terse**

[Default *unset*]

If **terse** is set, error messages may be less verbose. However, except for this caveat, error messages are unspecified. Furthermore, not all error messages need change for different settings of this option.

**warn**

[Default *set*]

If **warn** is set, and the contents of the edit buffer have been modified since they were last completely written, the editor shall write a warning message before certain ! commands (see [Escape](#), on page 2721).

**window**

[Default *see text*]

A value used in open and visual mode, by the <control>-B and <control>-F commands, and, in visual mode, to specify the number of lines displayed when the screen is repainted.

If the **-w** command-line option is not specified, the default value shall be set to the value of the *LINES* environment variable. If the *LINES* environment variable is empty or null, the default shall be the number of lines in the display minus 1.

Setting the **window** edit option to zero or to a value greater than the number of lines in the display minus 1 (either explicitly or based on the **-w** option or the *LINES* environment variable) shall cause the **window** edit option to be set to the number of lines in the display minus 1.

The baud rate of the terminal line may change the default in an implementation-defined manner.

**wrapmargin, wm**

[Default 0]

If the value of this edit option is zero, it shall have no effect.

If not in the POSIX locale, the effect of this edit option is implementation-defined.

Otherwise, it shall specify a number of columns from the ending margin of the terminal.

During open and visual text input modes, for each character for which any part of the character is displayed in a column that is less than **wrapmargin** columns from the ending margin of the display line, the editor shall behave as follows:

1. If the character triggering this event is a <blank>, it, and all immediately preceding <blank> characters on the current line entered during the execution of the current text input command, shall be discarded, and the editor shall behave as if the user had entered a single <newline> instead. In addition, if the next user-entered character is a <space>, it shall be discarded as well.
2. Otherwise, if there are one or more <blank> characters on the current line immediately preceding the last group of inserted non-<blank> characters which was entered during the execution of the current text input command, the <blank> characters shall be replaced as if the user had entered a single <newline> instead.

If the **autoindent** edit option is set, and the events described in 1. or 2. are performed, any <blank> characters at or after the cursor in the current line shall be discarded.

The ending margin shall be determined by the system or overridden by the user, as described for *COLUMNS* in the ENVIRONMENT VARIABLES section and XBD [Chapter 8](#) (on page 173).

89418 **wrapsan, ws**

89419 [Default *set*]

89420 If **wrapsan** is set, searches (the *ex* / or ? addresses, or open and visual mode /, ?, N, and n  
89421 commands) shall wrap around the beginning or end of the edit buffer; when unset, searches  
89422 shall stop at the beginning or end of the edit buffer.

89423 **writeany, wa**

89424 [Default *unset*]

89425 If **writeany** is set, some of the checks performed when executing the *ex* **write** commands shall be  
89426 inhibited, as described in editor option **autowrite**.

## 89427 EXIT STATUS

89428 The following exit values shall be returned:

89429 0 Successful completion.

89430 >0 An error occurred.

## 89431 CONSEQUENCES OF ERRORS

89432 When any error is encountered and the standard input is not a terminal device file, *ex* shall not  
89433 write the file or return to command or text input mode, and shall terminate with a non-zero exit  
89434 status.

89435 Otherwise, when an unrecoverable error is encountered, it shall be equivalent to a SIGHUP  
89436 asynchronous event.

89437 Otherwise, when an error is encountered, the editor shall behave as specified in [Command Line](#)  
89438 [Parsing in ex](#) (on page 2694).

## 89439 APPLICATION USAGE

89440 If a SIGSEGV signal is received while *ex* is saving a file, the file might not be successfully saved.

89441 The **next** command can accept more than one file, so usage such as:

89442 `next `ls [abc]*``

89443 is valid; it would not be valid for the **edit** or **read** commands, for example, because they expect  
89444 only one file and unspecified results occur.

## 89445 EXAMPLES

89446 None.

## 89447 RATIONALE

89448 The *ex/vi* specification is based on the historical practice found in the 4 BSD and System V  
89449 implementations of *ex* and *vi*.

89450 A *restricted editor* (both the historical *red* utility and modifications to *ex*) were considered and  
89451 rejected for inclusion. Neither option provided the level of security that users might expect.

89452 It is recognized that *ex* visual mode and related features would be difficult, if not impossible, to  
89453 implement satisfactorily on a block-mode terminal, or a terminal without any form of cursor  
89454 addressing; thus, it is not a mandatory requirement that such features should work on all  
89455 terminals. It is the intention, however, that an *ex* implementation should provide the full set of  
89456 capabilities on all terminals capable of supporting them.



## Options

The `-c` replacement for `+command` was inspired by the `-e` option of `sed`. Historically, all such commands (see `edit` and `next` as well) were executed from the last line of the edit buffer. This meant, for example, that `+/pattern` would fail unless the `wrapsan` option was set. POSIX.1-2008 requires conformance to historical practice. The `+command` option is no longer specified by POSIX.1-2008 but may be present in some implementations. Historically, some implementations restricted the `ex` commands that could be listed as part of the command line arguments. For consistency, POSIX.1-2008 does not permit these restrictions.

In historical implementations of the editor, the `-R` option (and the `readonly` edit option) only prevented overwriting of files; appending to files was still permitted, mapping loosely into the `cs`h `noclobber` variable. Some implementations, however, have not followed this semantic, and `readonly` does not permit appending either. POSIX.1-2008 follows the latter practice, believing that it is a more obvious and intuitive meaning of `readonly`.

The `-s` option suppresses all interactive user feedback and is useful for editing scripts in batch jobs. The list of specific effects is historical practice. The terminal type “incapable of supporting open and visual modes” has historically been named “dumb”.

The `-t` option was required because the `ctags` utility appears in POSIX.1-2008 and the option is available in all historical implementations of `ex`.

Historically, the `ex` and `vi` utilities accepted a `-x` option, which did encryption based on the algorithm found in the historical `crypt` utility. The `-x` option for encryption, and the associated `crypt` utility, were omitted because the algorithm used was not specifiable and the export control laws of some nations make it difficult to export cryptographic technology. In addition, it did not historically provide the level of security that users might expect.

## Standard Input

An end-of-file condition is not equivalent to an end-of-file character. A common end-of-file character, `<control>-D`, is historically an `ex` command.

There was no maximum line length in historical implementations of `ex`. Specifically, as it was parsed in chunks, the addresses had a different maximum length than the filenames. Further, the maximum line buffer size was declared as `BUFSIZ`, which was different lengths on different systems. This version selected the value of `{LINE_MAX}` to impose a reasonable restriction on portable usage of `ex` and to aid test suite writers in their development of realistic tests that exercise this limit.

## Input Files

It was an explicit decision by the standard developers that a `<newline>` be added to any file lacking one. It was believed that this feature of `ex` and `vi` was relied on by users in order to make text files lacking a trailing `<newline>` more portable. It is recognized that this will require a user-specified option or extension for implementations that permit `ex` and `vi` to edit files of type other than text if such files are not otherwise identified by the system. It was agreed that the ability to edit files of arbitrary type can be useful, but it was not considered necessary to mandate that an `ex` or `vi` implementation be required to handle files other than text files.

The paragraph in the INPUT FILES section, “By default, ...”, is intended to close a long-standing security problem in `ex` and `vi`; that of the “modeline” or “modelines” edit option. This feature allows any line in the first or last five lines of the file containing the strings `"ex:"` or `"vi:"` (and, apparently, `"ei:"` or `"vx:"`) to be a line containing editor commands, and `ex` interprets all the text up to the next `':'` or `<newline>` as a command. Consider the

consequences, for example, of an unsuspecting user using *ex* or *vi* as the editor when replying to a mail message in which a line such as:

```
ex: ! rm -rf :
```

appeared in the signature lines. The standard developers believed strongly that an editor should not by default interpret any lines of a file. Vendors are strongly urged to delete this feature from their implementations of *ex* and *vi*.

## Asynchronous Events

The intention of the phrase “complete write” is that the entire edit buffer be written to stable storage. The note regarding temporary files is intended for implementations that use temporary files to back edit buffers unnamed by the user.

Historically, SIGQUIT was ignored by *ex*, but was the equivalent of the **Q** command in visual mode; that is, it exited visual mode and entered *ex* mode. POSIX.1-2008 permits, but does not require, this behavior. Historically, SIGINT was often used by *vi* users to terminate text input mode (<control>-C is often easier to enter than <ESC>). Some implementations of *vi* alerted the terminal on this event, and some did not. POSIX.1-2008 requires that SIGINT behave identically to <ESC>, and that the terminal not be alerted.

Historically, suspending the *ex* editor during text input mode was similar to SIGINT, as completed lines were retained, but any partial line discarded, and the editor returned to command mode. POSIX.1-2008 is silent on this issue; implementations are encouraged to follow historical practice, where possible.

Historically, the *vi* editor did not treat SIGTSTP as an asynchronous event, and it was therefore impossible to suspend the editor in visual text input mode. There are two major reasons for this. The first is that SIGTSTP is a broadcast signal on UNIX systems, and the chain of events where the shell *execs* an application that then *execs vi* usually caused confusion for the terminal state if SIGTSTP was delivered to the process group in the default manner. The second was that most implementations of the UNIX *curses* package did not handle SIGTSTP safely, and the receipt of SIGTSTP at the wrong time would cause them to crash. POSIX.1-2008 is silent on this issue; implementations are encouraged to treat suspension as an asynchronous event if possible.

Historically, modifications to the edit buffer made before SIGINT interrupted an operation were retained; that is, anywhere from zero to all of the lines to be modified might have been modified by the time the SIGINT arrived. These changes were not discarded by the arrival of SIGINT. POSIX.1-2008 permits this behavior, noting that the **undo** command is required to be able to undo these partially completed commands.

The action taken for signals other than SIGINT, SIGCONT, SIGHUP, and SIGTERM is unspecified because some implementations attempt to save the edit buffer in a useful state when other signals are received.

## Standard Error

For *ex/vi*, diagnostic messages are those messages reported as a result of a failed attempt to invoke *ex* or *vi*, such as invalid options or insufficient resources, or an abnormal termination condition. Diagnostic messages should not be confused with the error messages generated by inappropriate or illegal user commands.

## Initialization in *ex* and *vi*

If an *ex* command (other than **cd**, **chdir**, or **source**) has a filename argument, one or both of the alternate and current pathnames will be set. Informally, they are set as follows:

1. If the *ex* command is one that replaces the contents of the edit buffer, and it succeeds, the current pathname will be set to the filename argument (the first filename argument in the case of the **next** command) and the alternate pathname will be set to the previous current pathname, if there was one.
2. In the case of the file read/write forms of the **read** and **write** commands, if there is no current pathname, the current pathname will be set to the filename argument.
3. Otherwise, the alternate pathname will be set to the filename argument.

For example, **:edit foo** and **:recover foo**, when successful, set the current pathname, and, if there was a previous current pathname, the alternate pathname. The commands **:write**, **!command**, and **:edit** set neither the current or alternate pathnames. If the **:edit foo** command were to fail for some reason, the alternate pathname would be set. The **read** and **write** commands set the alternate pathname to their *file* argument, unless the current pathname is not set, in which case they set the current pathname to their *file* arguments. The alternate pathname was not historically set by the **:source** command. POSIX.1-2008 requires conformance to historical practice. Implementations adding commands that take filenames as arguments are encouraged to set the alternate pathname as described here.

Historically, *ex* and *vi* read the **.exrc** file in the *\$HOME* directory twice, if the editor was executed in the *\$HOME* directory. POSIX.1-2008 prohibits this behavior.

Historically, the 4 BSD *ex* and *vi* read the *\$HOME* and local **.exrc** files if they were owned by the real ID of the user, or the **sourceany** option was set, regardless of other considerations. This was a security problem because it is possible to put normal UNIX system commands inside a **.exrc** file. POSIX.1-2008 does not specify the **sourceany** option, and historical implementations are encouraged to delete it.

The **.exrc** files must be owned by the real ID of the user, and not writable by anyone other than the owner. The appropriate privileges exception is intended to permit users to acquire special privileges, but continue to use the **.exrc** files in their home directories.

System V Release 3.2 and later *vi* implementations added the option **[no]exrc**. The behavior is that local **.exrc** files are read-only if the **exrc** option is set. The default for the **exrc** option was off, so by default, local **.exrc** files were not read. The problem this was intended to solve was that System V permitted users to give away files, so there is no possible ownership or writeability test to ensure that the file is safe. This is still a security problem on systems where users can give away files, but there is nothing additional that POSIX.1-2008 can do. The implementation-defined exception is intended to permit groups to have local **.exrc** files that are shared by users, by creating pseudo-users to own the shared files.

POSIX.1-2008 does not mention system-wide *ex* and *vi* start-up files. While they exist in several implementations of *ex* and *vi*, they are not present in any implementations considered historical practice by POSIX.1-2008. Implementations that have such files should use them only if they are owned by the real user ID or an appropriate user (for example, root on UNIX systems) and if they are not writable by any user other than their owner. System-wide start-up files should be read before the *EXINIT* variable, *\$HOME/.exrc*, or local **.exrc** files are evaluated.

Historically, any *ex* command could be entered in the *EXINIT* variable or the **.exrc** file, although ones requiring that the edit buffer already contain lines of text generally caused historical implementations of the editor to drop **core**. POSIX.1-2008 requires that any *ex* command be

permitted in the *EXINIT* variable and *.exrc* files, for simplicity of specification and consistency, although many of them will obviously fail under many circumstances.

The initialization of the contents of the edit buffer uses the phrase “the effect shall be” with regard to various *ex* commands. The intent of this phrase is that edit buffer contents loaded during the initialization phase not be lost; that is, loading the edit buffer should fail if the *.exrc* file read in the contents of a file and did not subsequently write the edit buffer. An additional intent of this phrase is to specify that the initial current line and column is set as specified for the individual *ex* commands.

Historically, the *-t* option behaved as if the tag search were a *+command*; that is, it was executed from the last line of the file specified by the tag. This resulted in the search failing if the pattern was a forward search pattern and the *wraps* edit option was not set. POSIX.1-2008 does not permit this behavior, requiring that the search for the tag pattern be performed on the entire file, and, if not found, that the current line be set to a more reasonable location in the file.

Historically, the empty edit buffer presented for editing when a file was not specified by the user was unnamed. This is permitted by POSIX.1-2008; however, implementations are encouraged to provide users a temporary filename for this buffer because it permits them the use of *ex* commands that use the current pathname during temporary edit sessions.

Historically, the file specified using the *-t* option was not part of the current argument list. This practice is permitted by POSIX.1-2008; however, implementations are encouraged to include its name in the current argument list for consistency.

Historically, the *-c* command was generally not executed until a file that already exists was edited. POSIX.1-2008 requires conformance to this historical practice. Commands that could cause the *-c* command to be executed include the *ex* commands **edit**, **next**, **recover**, **rewind**, and **tag**, and the *vi* commands *<control>-^* and *<control>-]*. Historically, reading a file into an edit buffer did not cause the *-c* command to be executed (even though it might set the current pathname) with the exception that it did cause the *-c* command to be executed if: the editor was in *ex* mode, the edit buffer had no current pathname, the edit buffer was empty, and no read commands had yet been attempted. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

Historically, the *-r* option was the same as a normal edit session if there was no recovery information available for the file. This allowed users to enter:

```
vi -r *.c
```

and recover whatever files were recoverable. In some implementations, recovery was attempted only on the first file named, and the file was not entered into the argument list; in others, recovery was attempted for each file named. In addition, some historical implementations ignored *-r* if *-t* was specified or did not support command line *file* arguments with the *-t* option. For consistency and simplicity of specification, POSIX.1-2008 disallows these special cases, and requires that recovery be attempted the first time each file is edited.

Historically, *vi* initialized the ‘ and ’ marks, but *ex* did not. This meant that if the first command in *ex* mode was **visual** or if an *ex* command was executed first (for example, *vi +10 file*), *vi* was entered without the marks being initialized. Because the standard developers believed the marks to be generally useful, and for consistency and simplicity of specification, POSIX.1-2008 requires that they always be initialized if in open or visual mode, or if in *ex* mode and the edit buffer is not empty. Not initializing it in *ex* mode if the edit buffer is empty is historical practice; however, it has always been possible to set (and use) marks in empty edit buffers in open and visual mode edit sessions.

## Addressing

Historically, *ex* and *vi* accepted the additional addressing forms '`\/'` and '`\?'`'. They were equivalent to '`/'`' and '`??'`', respectively. They are not required by POSIX.1-2008, mostly because nobody can remember whether they ever did anything different historically.

Historically, *ex* and *vi* permitted an address of zero for several commands, and permitted the % address in empty files for others. For consistency, POSIX.1-2008 requires support for the former in the few commands where it makes sense, and disallows it otherwise. In addition, because POSIX.1-2008 requires that % be logically equivalent to '`1,$`', it is also supported where it makes sense and disallowed otherwise.

Historically, the % address could not be followed by further addresses. For consistency and simplicity of specification, POSIX.1-2008 requires that additional addresses be supported.

All of the following are valid *addresses*:

- `+++` Three lines after the current line.
- `/re/-` One line before the next occurrence of *re*.
- `-2` Two lines before the current line.
- `3 ---- 2` Line one (note intermediate negative address).
- `1 2 3` Line six.

Any number of addresses can be provided to commands taking addresses; for example, '`1,2,3,4,5p`' prints lines 4 and 5, because two is the greatest valid number of addresses accepted by the **print** command. This, in combination with the <semicolon> delimiter, permits users to create commands based on ordered patterns in the file. For example, the command **3;/foo/;+2print** will display the first line after line 3 that contains the pattern *foo*, plus the next two lines. Note that the address **3;** must be evaluated before being discarded because the search origin for the **/foo/** command depends on this.

Historically, values could be added to addresses by including them after one or more <blank> characters; for example, **3 - 5p** wrote the seventh line of the file, and **/foo/ 5** was the same as **/foo/+5**. However, only absolute values could be added; for example, **5 /foo/** was an error. POSIX.1-2008 requires conformance to historical practice. Address offsets are separately specified from addresses because they could historically be provided to visual mode search commands.

Historically, any missing addresses defaulted to the current line. This was true for leading and trailing <comma>-delimited addresses, and for trailing <semicolon>-delimited addresses. For consistency, POSIX.1-2008 requires it for leading <semicolon> addresses as well.

Historically, *ex* and *vi* accepted the '`^`' character as both an address and as a flag offset for commands. In both cases it was identical to the '`-`' character. POSIX.1-2008 does not require or prohibit this behavior.

Historically, the enhancements to basic regular expressions could be used in addressing; for example, '`~'`', '`\<`', and '`\>`'. POSIX.1-2008 requires conformance to historical practice; that is, that regular expression usage be consistent, and that regular expression enhancements be supported wherever regular expressions are used.



## Command Line Parsing in ex

Historical *ex* command parsing was even more complex than that described here. POSIX.1-2008 requires the subset of the command parsing that the standard developers believed was documented and that users could reasonably be expected to use in a portable fashion, and that was historically consistent between implementations. (The discarded functionality is obscure, at best.) Historical implementations will require changes in order to comply with POSIX.1-2008; however, users are not expected to notice any of these changes. Most of the complexity in *ex* parsing is to handle three special termination cases:

1. The **!**, **global**, **v**, and the filter versions of the **read** and **write** commands are delimited by `<newline>` characters (they can contain `<vertical-line>` characters that are usually shell pipes).
2. The **ex**, **edit**, **next**, and **visual** in open and visual mode commands all take *ex* commands, optionally containing `<vertical-line>` characters, as their first arguments.
3. The **s** command takes a regular expression as its first argument, and uses the delimiting characters to delimit the command.

Historically, `<vertical-line>` characters in the `+command` argument of the **ex**, **edit**, **next**, **vi**, and **visual** commands, and in the *pattern* and *replacement* parts of the **s** command, did not delimit the command, and in the filter cases for **read** and **write**, and the **!**, **global**, and **v** commands, they did not delimit the command at all. For example, the following commands are all valid:

```
:edit +25 | s/abc/ABC/ file.c
:s/ | /PIPE/
:read !spell % | columnate
:global/pattern/p | l
:s/a/b/ | s/c/d | set
```

Historically, empty or `<blank>` filled lines in **.exrc** files and **sourced** files (as well as *EXINIT* variables and *ex* command scripts) were treated as default commands; that is, **print** commands. POSIX.1-2008 specifically requires that they be ignored when encountered in **.exrc** and **sourced** files to eliminate a common source of new user error.

Historically, *ex* commands with multiple adjacent (or `<blank>`-separated) vertical lines were handled oddly when executed from *ex* mode. For example, the command `l l l <carriage-return>`, when the cursor was on line 1, displayed lines 2, 3, and 5 of the file. In addition, the command `l` would only display the line after the next line, instead of the next two lines. The former worked more logically when executed from *vi* mode, and displayed lines 2, 3, and 4. POSIX.1-2008 requires the *vi* behavior; that is, a single default command and line number increment for each command separator, and trailing `<newline>` characters after `<vertical-line>` separators are discarded.

Historically, *ex* permitted a single extra `<colon>` as a leading command character; for example, **:g/pattern/:p** was a valid command. POSIX.1-2008 generalizes this to require that any number of leading `<colon>` characters be stripped.

Historically, any prefix of the **delete** command could be followed without intervening `<blank>` characters by a flag character because in the command **d p**, *p* is interpreted as the buffer *p*. POSIX.1-2008 requires conformance to historical practice.

Historically, the **k** command could be followed by the mark name without intervening `<blank>` characters. POSIX.1-2008 requires conformance to historical practice.

Historically, the **s** command could be immediately followed by flag and option characters; for example, **s/e/E/l s|sgc3p** was a valid command. However, flag characters could not stand alone;

for example, the commands **sp** and **s l** would fail, while the command **sgp** and **s gl** would succeed. (Obviously, the '#' flag character was used as a delimiter character if it followed the command.) Another issue was that option characters had to precede flag characters even when the command was fully specified; for example, the command **s/e/E/pg** would fail, while the command **s/e/E/gp** would succeed. POSIX.1-2008 requires conformance to historical practice.

Historically, the first command name that had a prefix matching the input from the user was the executed command; for example, **ve**, **ver**, and **vers** all executed the **version** command. Commands were in a specific order, however, so that **a** matched **append**, not **abbreviate**. POSIX.1-2008 requires conformance to historical practice. The restriction on command search order for implementations with extensions is to avoid the addition of commands such that the historical prefixes would fail to work portably.

Historical implementations of *ex* and *vi* did not correctly handle multiple *ex* commands, separated by <vertical-line> characters, that entered or exited visual mode or the editor. Because implementations of *vi* exist that do not exhibit this failure mode, POSIX.1-2008 does not permit it.

The requirement that alphabetic command names consist of all following alphabetic characters up to the next non-alphabetic character means that alphabetic command names must be separated from their arguments by one or more non-alphabetic characters, normally a <blank> or '!' character, except as specified for the exceptions, the **delete**, **k**, and **s** commands.

Historically, the repeated execution of the *ex* default **print** commands (<control>-D, *eof*, <newline>, <carriage-return>) erased any prompting character and displayed the next lines without scrolling the terminal; that is, immediately below any previously displayed lines. This provided a cleaner presentation of the lines in the file for the user. POSIX.1-2008 does not require this behavior because it may be impossible in some situations; however, implementations are strongly encouraged to provide this semantic if possible.

Historically, it was possible to change files in the middle of a command, and have the rest of the command executed in the new file; for example:

```
:edit +25 file.c | s/abc/ABC/ | 1
```

was a valid command, and the substitution was attempted in the newly edited file. POSIX.1-2008 requires conformance to historical practice. The following commands are examples that exercise the *ex* parser:

```
echo 'foo | bar' > file1; echo 'foo/bar' > file2;
```

```
vi
```

```
:edit +1 | s/|/PIPE/ | w file1 | e file2 | 1 | s/\/SLASH/ | wq
```

Historically, there was no protection in editor implementations to avoid *ex* **global**, **v**, **@**, or **\*** commands changing edit buffers during execution of their associated commands. Because this would almost invariably result in catastrophic failure of the editor, and implementations exist that do exhibit these problems, POSIX.1-2008 requires that changing the edit buffer during a **global** or **v** command, or during a **@** or **\*** command for which there will be more than a single execution, be an error. Implementations supporting multiple edit buffers simultaneously are strongly encouraged to apply the same semantics to switching between buffers as well.

The *ex* command quoting required by POSIX.1-2008 is a superset of the quoting in historical implementations of the editor. For example, it was not historically possible to escape a <blank> in a filename; for example, **:edit foo\\\ bar** would report that too many filenames had been entered for the edit command, and there was no method of escaping a <blank> in the first argument of an **edit**, **ex**, **next**, or **visual** command at all. POSIX.1-2008 extends historical practice, requiring that quoting behavior be made consistent across all *ex* commands, except for

the **map**, **unmap**, **abbreviate**, and **unabbreviate** commands, which historically used <control>-V instead of <backslash> characters for quoting. For those four commands, POSIX.1-2008 requires conformance to historical practice.

Backslash quoting in *ex* is non-intuitive. <backslash>-escapes are ignored unless they escape a special character; for example, when performing *file* argument expansion, the string "\\%" is equivalent to '\%', not "\<current\_pathname>". This can be confusing for users because <backslash> is usually one of the characters that causes shell expansion to be performed, and therefore shell quoting rules must be taken into consideration. Generally, quoting characters are only considered if they escape a special character, and a quoting character must be provided for each layer of parsing for which the character is special. As another example, only a single <backslash> is necessary for the '\1' sequence in substitute replacement patterns, because the character '1' is not special to any parsing layer above it.

<control>-V quoting in *ex* is slightly different from backslash quoting. In the four commands where <control>-V quoting applies (**abbreviate**, **unabbreviate**, **map**, and **unmap**), any character may be escaped by a <control>-V whether it would have a special meaning or not. POSIX.1-2008 requires conformance to historical practice.

Historical implementations of the editor did not require delimiters within character classes to be escaped; for example, the command :s/[/] on the string "xxx/yyy" would delete the '/' from the string. POSIX.1-2008 disallows this historical practice for consistency and because it places a large burden on implementations by requiring that knowledge of regular expressions be built into the editor parser.

Historically, quoting <newline> characters in *ex* commands was handled inconsistently. In most cases, the <newline> character always terminated the command, regardless of any preceding escape character, because <backslash> characters did not escape <newline> characters for most *ex* commands. However, some *ex* commands (for example, **s**, **map**, and **abbreviation**) permitted <newline> characters to be escaped (although in the case of **map** and **abbreviation**, <control>-V characters escaped them instead of <backslash> characters). This was true in not only the command line, but also **.exrc** and **sourced** files. For example, the command:

```
map = foo<control-V><newline>bar
```

would succeed, although it was sometimes difficult to get the <control>-V and the inserted <newline> passed to the *ex* parser. For consistency and simplicity of specification, POSIX.1-2008 requires that it be possible to escape <newline> characters in *ex* commands at all times, using <backslash> characters for most *ex* commands, and using <control>-V characters for the **map** and **abbreviation** commands. For example, the command **print<newline>list** is required to be parsed as the single command **print<newline>list**. While this differs from historical practice, POSIX.1-2008 developers believed it unlikely that any script or user depended on the historical behavior.

Historically, an error in a command specified using the **-c** option did not cause the rest of the **-c** commands to be discarded. POSIX.1-2008 disallows this for consistency with mapped keys, the **@**, **global**, **source**, and **v** commands, the *EXINIT* environment variable, and the **.exrc** files.



## Input Editing in *ex*

One of the common uses of the historical *ex* editor is over slow network connections. Editors that run in canonical mode can require far less traffic to and from, and far less processing on, the host machine, as well as more easily supporting block-mode terminals. For these reasons, POSIX.1-2008 requires that *ex* be implemented using canonical mode input processing, as was done historically.

POSIX.1-2008 does not require the historical 4 BSD input editing characters “word erase” or “literal next”. For this reason, it is unspecified how they are handled by *ex*, although they must have the required effect. Implementations that resolve them after the line has been ended using a <newline> or <control>-M character, and implementations that rely on the underlying system terminal support for this processing, are both conforming. Implementations are strongly urged to use the underlying system functionality, if at all possible, for compatibility with other system text input interfaces.

Historically, when the *eof* character was used to decrement the **autoindent** level, the cursor moved to display the new end of the **autoindent** characters, but did not move the cursor to a new line, nor did it erase the <control>-D character from the line. POSIX.1-2008 does not specify that the cursor remain on the same line or that the rest of the line is erased; however, implementations are strongly encouraged to provide the best possible user interface; that is, the cursor should remain on the same line, and any <control>-D character on the line should be erased.

POSIX.1-2008 does not require the historical 4 BSD input editing character “reprint”, traditionally <control>-R, which redisplayed the current input from the user. For this reason, and because the functionality cannot be implemented after the line has been terminated by the user, POSIX.1-2008 makes no requirements about this functionality. Implementations are strongly urged to make this historical functionality available, if possible.

Historically, <control>-Q did not perform a literal next function in *ex*, as it did in *vi*. POSIX.1-2008 requires conformance to historical practice to avoid breaking historical *ex* scripts and **.exrc** files.

## *eof*

Whether the *eof* character immediately modifies the **autoindent** characters in the prompt is left unspecified so that implementations can conform in the presence of systems that do not support this functionality. Implementations are encouraged to modify the line and redisplay it immediately, if possible.

The specification of the handling of the *eof* character differs from historical practice only in that *eof* characters are not discarded if they follow normal characters in the text input. Historically, they were always discarded.

## Command Descriptions in *ex*

Historically, several commands (for example, **global**, **v**, **visual**, **s**, **write**, **wq**, **yank**, **!**, **<**, **>**, **&**, and **~**) were executable in empty files (that is, the default address(es) were 0), or permitted explicit addresses of 0 (for example, 0 was a valid address, or 0,0 was a valid range). Addresses of 0, or command execution in an empty file, make sense only for commands that add new text to the edit buffer or write commands (because users may wish to write empty files). POSIX.1-2008 requires this behavior for such commands and disallows it otherwise, for consistency and simplicity of specification.

A count to an *ex* command has been historically corrected to be no greater than the last line in a

file; for example, in a five-line file, the command **1,6print** would fail, but the command **1print300** would succeed. POSIX.1-2008 requires conformance to historical practice.

Historically, the use of flags in *ex* commands could be obscure. General historical practice was as described by POSIX.1-2008, but there were some special cases. For instance, the **list**, **number**, and **print** commands ignored trailing address offsets; for example, **3p +++#** would display line 3, and 3 would be the current line after the execution of the command. The **open** and **visual** commands ignored both the trailing offsets and the trailing flags. Also, flags specified to the **open** and **visual** commands interacted badly with the **list** edit option, and setting and then unsetting it during the open/visual session would cause *vi* to stop displaying lines in the specified format. For consistency and simplicity of specification, POSIX.1-2008 does not permit any of these exceptions to the general rule.

POSIX.1-2008 uses the word *copy* in several places when discussing buffers. This is not intended to imply implementation.

Historically, *ex* users could not specify numeric buffers because of the ambiguity this would cause; for example, in the command **3 delete 2**, it is unclear whether 2 is a buffer name or a *count*. POSIX.1-2008 requires conformance to historical practice by default, but does not preclude extensions.

Historically, the contents of the unnamed buffer were frequently discarded after commands that did not explicitly affect it; for example, when using the **edit** command to switch files. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

The *ex* utility did not historically have access to the numeric buffers, and, furthermore, deleting lines in *ex* did not modify their contents. For example, if, after doing a delete in *vi*, the user switched to *ex*, did another delete, and then switched back to *vi*, the contents of the numeric buffers would not have changed. POSIX.1-2008 requires conformance to historical practice. Numeric buffers are described in the *ex* utility in order to confine the description of buffers to a single location in POSIX.1-2008.

The metacharacters that trigger shell expansion in *file* arguments match historical practice, as does the method for doing shell expansion. Implementations wishing to provide users with the flexibility to alter the set of metacharacters are encouraged to provide a **shellmeta** string edit option.

Historically, *ex* commands executed from *vi* refreshed the screen when it did not strictly need to do so; for example, **!date > /dev/null** does not require a screen refresh because the output of the UNIX *date* command requires only a single line of the screen. POSIX.1-2008 requires that the screen be refreshed if it has been overwritten, but makes no requirements as to how an implementation should make that determination. Implementations may prompt and refresh the screen regardless.

## Abbreviate

Historical practice was that characters that were entered as part of an abbreviation replacement were subject to **map** expansions, the **showmatch** edit option, further abbreviation expansions, and so on; that is, they were logically pushed onto the terminal input queue, and were not a simple replacement. POSIX.1-2008 requires conformance to historical practice. Historical practice was that whenever a non-word character (that had not been escaped by a <control>-V) was entered after a word character, *vi* would check for abbreviations. The check was based on the type of the character entered before the word character of the word/non-word pair that triggered the check. The word character of the word/non-word pair that triggered the check and all characters entered before the trigger pair that were of that type were included in the check, with the exception of <blank> characters, which always delimited the abbreviation.

This means that, for the abbreviation to work, the *lhs* must end with a word character, there can be no transitions from word to non-word characters (or *vice versa*) other than between the last and next-to-last characters in the *lhs*, and there can be no <blank> characters in the *lhs*. In addition, because of the historical quoting rules, it was impossible to enter a literal <control>-V in the *lhs*. POSIX.1-2008 requires conformance to historical practice. Historical implementations did not inform users when abbreviations that could never be used were entered; implementations are strongly encouraged to do so.

For example, the following abbreviations will work:

```
:ab (p REPLACE
:ab p REPLACE
:ab ((p REPLACE
```

The following abbreviations will not work:

```
:ab ( REPLACE
:ab (pp REPLACE
```

Historical practice is that words on the *vi* colon command line were subject to abbreviation expansion, including the arguments to the **abbrev** (and more interestingly) the **unabbrev** command. Because there are implementations that do not do abbreviation expansion for the first argument to those commands, this is permitted, but not required, by POSIX.1-2008. However, the following sequence:

```
:ab foo bar
:ab foo baz
```

resulted in the addition of an abbreviation of "baz" for the string "bar" in historical *ex/vi*, and the sequence:

```
:ab foo1 bar
:ab foo2 bar
:unabbreviate foo2
```

deleted the abbreviation "foo1", not "foo2". These behaviors are not permitted by POSIX.1-2008 because they clearly violate the expectations of the user.

It was historical practice that <control>-V, not <backslash>, characters be interpreted as escaping subsequent characters in the **abbreviate** command. POSIX.1-2008 requires conformance to historical practice; however, it should be noted that an abbreviation containing a <blank> will never work.

## Append

Historically, any text following a <vertical-line> command separator after an **append**, **change**, or **insert** command became part of the insert text. For example, in the command:

```
:g/pattern/append|stuff1
```

a line containing the text "stuff1" would be appended to each line matching pattern. It was also historically valid to enter:

```
:append|stuff1
stuff2
.
```

and the text on the *ex* command line would be appended along with the text inserted after it. There was an historical bug, however, that the user had to enter two terminating lines (the ' . '

lines) to terminate text input mode in this case. POSIX.1-2008 requires conformance to historical practice, but disallows the historical need for multiple terminating lines.

### Change

See the RATIONALE for the **append** command. Historical practice for cursor positioning after the change command when no text is input, is as described in POSIX.1-2008. However, one System V implementation is known to have been modified such that the cursor is positioned on the first address specified, and not on the line before the first address. POSIX.1-2008 disallows this modification for consistency.

Historically, the **change** command did not support buffer arguments, although some implementations allow the specification of an optional buffer. This behavior is neither required nor disallowed by POSIX.1-2008.

### Change Directory

A common extension in *ex* implementations is to use the elements of a **cdpath** edit option as prefix directories for *path* arguments to **chdir** that are relative pathnames and that do not have ' . ' or " . . " as their first component. Elements in the **cdpath** edit option are <colon>-separated. The initial value of the **cdpath** edit option is the value of the shell *CDPATH* environment variable. This feature was not included in POSIX.1-2008 because it does not exist in any of the implementations considered historical practice.

### Copy

Historical implementations of *ex* permitted copies to lines inside of the specified range; for example, **:2,5copy3** was a valid command. POSIX.1-2008 requires conformance to historical practice.

### Delete

POSIX.1-2008 requires support for the historical parsing of a **delete** command followed by flags, without any intervening <blank> characters. For example:

**1dp** Deletes the first line and prints the line that was second.

**1delep** As for **1dp**.

**1d** Deletes the first line, saving it in buffer *p*.

**1d p1l** (Pee-one-ell.) Deletes the first line, saving it in buffer *p*, and listing the line that was second.

### Edit

Historically, any *ex* command could be entered as a *+command* argument to the **edit** command, although some (for example, **insert** and **append**) were known to confuse historical implementations. For consistency and simplicity of specification, POSIX.1-2008 requires that any command be supported as an argument to the **edit** command.

Historically, the command argument was executed with the current line set to the last line of the file, regardless of whether the **edit** command was executed from visual mode or not. POSIX.1-2008 requires conformance to historical practice.

Historically, the *+command* specified to the **edit** and **next** commands was delimited by the first <blank>, and there was no way to quote them. For consistency, POSIX.1-2008 requires that the usual *ex* backslash quoting be provided.

Historically, specifying the *+command* argument to the edit command required a filename to be specified as well; for example, **:edit +100** would always fail. For consistency and simplicity of specification, POSIX.1-2008 does not permit this usage to fail for that reason.

Historically, only the cursor position of the last file edited was remembered by the editor. POSIX.1-2008 requires that this be supported; however, implementations are permitted to remember and restore the cursor position for any file previously edited.

## File

Historical versions of the *ex* editor **file** command displayed a current line and number of lines in the edit buffer of 0 when the file was empty, while the *vi* <control>-G command displayed a current line and number of lines in the edit buffer of 1 in the same situation. POSIX.1-2008 does not permit this discrepancy, instead requiring that a message be displayed indicating that the file is empty.

## Global

The two-pass operation of the **global** and **v** commands is not intended to imply implementation, only the required result of the operation.

The current line and column are set as specified for the individual *ex* commands. This requirement is cumulative; that is, the current line and column must track across all the commands executed by the **global** or **v** commands.

## Insert

See the RATIONALE for the **append** command.

Historically, **insert** could not be used with an address of zero; that is, not when the edit buffer was empty. POSIX.1-2008 requires that this command behave consistently with the **append** command.

## Join

The action of the **join** command in relation to the special characters is only defined for the POSIX locale because the correct amount of white space after a period varies; in Japanese none is required, in French only a single space, and so on.

## List

The historical output of the **list** command was potentially ambiguous. The standard developers believed correcting this to be more important than adhering to historical practice, and POSIX.1-2008 requires unambiguous output.

## Map

Historically, command mode maps only applied to command names; for example, if the character 'x' was mapped to 'y', the command **fx** searched for the 'x' character, not the 'y' character. POSIX.1-2008 requires this behavior. Historically, entering <control>-V as the first character of a *vi* command was an error. Several implementations have extended the semantics of *vi* such that <control>-V means that the subsequent command character is not mapped. This is permitted, but not required, by POSIX.1-2008. Regardless, using <control>-V to escape the second or later character in a sequence of characters that might match a **map** command, or any character in text input mode, is historical practice, and stops the entered keys from matching a map. POSIX.1-2008 requires conformance to historical practice.



Historical implementations permitted digits to be used as a **map** command *lhs*, but then ignored the map. POSIX.1-2008 requires that the mapped digits not be ignored.

The historical implementation of the **map** command did not permit **map** commands that were more than a single character in length if the first character was printable. This behavior is permitted, but not required, by POSIX.1-2008.

Historically, mapped characters were remapped unless the **remap** edit option was not set, or the prefix of the mapped characters matched the mapping characters; for example, in the **map**:

```
:map ab abcd
```

the characters "ab" were used as is and were not remapped, but the characters "cd" were mapped if appropriate. This can cause infinite loops in the *vi* mapping mechanisms. POSIX.1-2008 requires conformance to historical practice, and that such loops be interruptible.

Text input maps had the same problems with expanding the *lhs* for the **ex map!** and **unmap!** command as did the **ex abbreviate** and **unabbreviate** commands. See the RATIONALE for the **ex abbreviate** command. POSIX.1-2008 requires similar modification of some historical practice for the **map** and **unmap** commands, as described for the **abbreviate** and **unabbreviate** commands.

Historically, **maps** that were subsets of other **maps** behaved differently depending on the order in which they were defined. For example:

```
:map! ab      short
:map! abc     long
```

would always translate the characters "ab" to "short", regardless of how fast the characters "abc" were entered. If the entry order was reversed:

```
:map! abc     long
:map! ab      short
```

the characters "ab" would cause the editor to pause, waiting for the completing 'c' character, and the characters might never be mapped to "short". For consistency and simplicity of specification, POSIX.1-2008 requires that the shortest match be used at all times.

The length of time the editor spends waiting for the characters to complete the *lhs* is unspecified because the timing capabilities of systems are often inexact and variable, and it may depend on other factors such as the speed of the connection. The time should be long enough for the user to be able to complete the sequence, but not long enough for the user to have to wait. Some implementations of *vi* have added a **keytime** option, which permits users to set the number of 0,1 seconds the editor waits for the completing characters. Because mapped terminal function and cursor keys tend to start with an <ESC> character, and <ESC> is the key ending *vi* text input mode, **maps** starting with <ESC> characters are generally exempted from this timeout period, or, at least timed out differently.

## Mark

Historically, users were able to set the "previous context" marks explicitly. In addition, the **ex** commands **"** and **"** and the *vi* commands **"**, **"**, and **"** all referred to the same mark. In addition, the previous context marks were not set if the command, with which the address setting the mark was associated, failed. POSIX.1-2008 requires conformance to historical practice. Historically, if marked lines were deleted, the mark was also deleted, but would reappear if the change was undone. POSIX.1-2008 requires conformance to historical practice.

The description of the special events that set the ' and ' marks matches historical practice. For example, historically the command **/a/** did not set the ' and ' marks, but the command

90069        **/a/,/b/delete** did.

## 90070        **Next**

90071        Historically, any *ex* command could be entered as a *+command* argument to the **next** command,  
90072        although some (for example, **insert** and **append**) were known to confuse historical  
90073        implementations. POSIX.1-2008 requires that any command be permitted and that it behave as  
90074        specified. The **next** command can accept more than one file, so usage such as:

90075        `next `ls [abc] ``

90076        is valid; it need not be valid for the **edit** or **read** commands, for example, because they expect  
90077        only one filename.

90078        Historically, the **next** command behaved differently from the **:rewind** command in that it  
90079        ignored the force flag if the **autowrite** flag was set. For consistency, POSIX.1-2008 does not  
90080        permit this behavior.

90081        Historically, the **next** command positioned the cursor as if the file had never been edited before,  
90082        regardless. POSIX.1-2008 does not permit this behavior, for consistency with the **edit** command.

90083        Implementations wanting to provide a counterpart to the **next** command that edited the  
90084        previous file have used the command **prev[ious]**, which takes no *file* argument. POSIX.1-2008  
90085        does not require this command.

## 90086        **Open**

90087        Historically, the **open** command would fail if the **open** edit option was not set. POSIX.1-2008  
90088        does not mention the **open** edit option and does not require this behavior. Some historical  
90089        implementations do not permit entering open mode from open or visual mode, only from *ex*  
90090        mode. For consistency, POSIX.1-2008 does not permit this behavior.

90091        Historically, entering open mode from the command line (that is, *vi +open*) resulted in  
90092        anomalous behaviors; for example, the *ex* file and *set* commands, and the *vi* command  
90093        <control>-G did not work. For consistency, POSIX.1-2008 does not permit this behavior.

90094        Historically, the **open** command only permitted ' / ' characters to be used as the search pattern  
90095        delimiter. For consistency, POSIX.1-2008 requires that the search delimiters used by the **s**, **global**,  
90096        and **v** commands be accepted as well.

## 90097        **Preserve**

90098        The **preserve** command does not historically cause the file to be considered unmodified for the  
90099        purposes of future commands that may exit the editor. POSIX.1-2008 requires conformance to  
90100        historical practice.

90101        Historical documentation stated that mail was not sent to the user when preserve was executed;  
90102        however, historical implementations did send mail in this case. POSIX.1-2008 requires  
90103        conformance to the historical implementations.

## Print

The writing of NUL by the **print** command is not specified as a special case because the standard developers did not want to require *ex* to support NUL characters. Historically, characters were displayed using the ARPA standard mappings, which are as follows:

1. Printable characters are left alone.
2. Control characters less than \177 are represented as '^' followed by the character offset from the '@' character in the ASCII map; for example, \007 is represented as '^G'.
3. \177 is represented as '^?' followed by '? '.

The display of characters having their eighth bit set was less standard. Existing implementations use hex (0x00), octal (\000), and a meta-bit display. (The latter displayed bytes that had their eighth bit set as the two characters "M-" followed by the seven-bit display as described above.) The latter probably has the best claim to historical practice because it was used for the **-v** option of 4 BSD and 4 BSD-derived versions of the *cat* utility since 1980.

No specific display format is required by POSIX.1-2008.

Explicit dependence on the ASCII character set has been avoided where possible, hence the use of the phrase an "implementation-defined multi-character sequence" for the display of non-printable characters in preference to the historical usage of, for instance, "^I" for the <tab>. Implementations are encouraged to conform to historical practice in the absence of any strong reason to diverge.

Historically, all *ex* commands beginning with the letter 'p' could be entered using capitalized versions of the commands; for example, **P[rint]**, **Pre[serve]**, and **Pu[t]** were all valid command names. POSIX.1-2008 permits, but does not require, this historical practice because capital forms of the commands are used by some implementations for other purposes.

## Put

Historically, an *ex* **put** command, executed from open or visual mode, was the same as the open or visual mode **P** command, if the buffer was named and was cut in character mode, and the same as the **p** command if the buffer was named and cut in line mode. If the unnamed buffer was the source of the text, the entire line from which the text was taken was usually **put**, and the buffer was handled as if in line mode, but it was possible to get extremely anomalous behavior. In addition, using the **Q** command to switch into *ex* mode, and then doing a **put** often resulted in errors as well, such as appending text that was unrelated to the (supposed) contents of the buffer. For consistency and simplicity of specification, POSIX.1-2008 does not permit these behaviors. All *ex* **put** commands are required to operate in line mode, and the contents of the buffers are not altered by changing the mode of the editor.

## Read

Historically, an *ex* **read** command executed from open or visual mode, executed in an empty file, left an empty line as the first line of the file. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior. Historically, a **read** in open or visual mode from a program left the cursor at the last line read in, not the first. For consistency, POSIX.1-2008 does not permit this behavior.

Historical implementations of *ex* were unable to undo **read** commands that read from the output of a program. For consistency, POSIX.1-2008 does not permit this behavior.

Historically, the *ex* and *vi* message after a successful **read** or **write** command specified "characters", not "bytes". POSIX.1-2008 requires that the number of bytes be displayed, not the



number of characters, because it may be difficult in multi-byte implementations to determine the number of characters read. Implementations are encouraged to clarify the message displayed to the user.

Historically, reads were not permitted on files other than type regular, except that FIFO files could be read (probably only because they did not exist when *ex* and *vi* were originally written). Because the historical *ex* evaluated **read!** and **read !** equivalently, there can be no optional way to force the read. POSIX.1-2008 permits, but does not require, this behavior.

### Recover

Some historical implementations of the editor permitted users to recover the edit buffer contents from a previous edit session, and then exit without saving those contents (or explicitly discarding them). The intent of POSIX.1-2008 in requiring that the edit buffer be treated as already modified is to prevent this user error.

### Rewind

Historical implementations supported the **rewind** command when the user was editing the first file in the list; that is, the file that the **rewind** command would edit. POSIX.1-2008 requires conformance to historical practice.

### Substitute

Historically, *ex* accepted an **r** option to the **s** command. The effect of the **r** option was to use the last regular expression used in any command as the pattern, the same as the **~** command. The **r** option is not required by POSIX.1-2008. Historically, the **c** and **g** options were toggled; for example, the command **:s/abc/def/** was the same as **s/abc/def/ccccgggg**. For simplicity of specification, POSIX.1-2008 does not permit this behavior.

The tilde command is often used to replace the last search RE. For example, in the sequence:

```
s/red/blue/
/green
~
```

the **~** command is equivalent to:

```
s/green/blue/
```

Historically, *ex* accepted all of the following forms:

```
s/abc/def/
s/abc/def
s/abc/
s/abc
```

POSIX.1-2008 requires conformance to this historical practice.

The **s** command presumes that the **'^'** character only occupies a single column in the display. Much of the *ex* and *vi* specification presumes that the **<space>** only occupies a single column in the display. There are no known character sets for which this is not true.

Historically, the final column position for the substitute commands was based on previous column movements; a search for a pattern followed by a substitution would leave the column position unchanged, while a **0** command followed by a substitution would change the column position to the first non-**<blank>**. For consistency and simplicity of specification, POSIX.1-2008 requires that the final column position always be set to the first non-**<blank>**.

## Set

Historical implementations redisplayed all of the options for each occurrence of the **all** keyword. POSIX.1-2008 permits, but does not require, this behavior.

## Tag

No requirement is made as to where *ex* and *vi* shall look for the file referenced by the tag entry. Historical practice has been to look for the path found in the **tags** file, based on the current directory. A useful extension found in some implementations is to look based on the directory containing the tags file that held the entry, as well. No requirement is made as to which reference for the tag in the tags file is used. This is deliberate, in order to permit extensions such as multiple entries in a tags file for a tag.

Because users often specify many different tags files, some of which need not be relevant or exist at any particular time, POSIX.1-2008 requires that error messages about problem tags files be displayed only if the requested tag is not found, and then, only once for each time that the **tag** edit option is changed.

The requirement that the current edit buffer be unmodified is only necessary if the file indicated by the tag entry is not the same as the current file (as defined by the current pathname). Historically, the file would be reloaded if the filename had changed, as well as if the filename was different from the current pathname. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior, requiring that the name be the only factor in the decision.

Historically, *vi* only searched for tags in the current file from the current cursor to the end of the file, and therefore, if the **wrapsan** option was not set, tags occurring before the current cursor were not found. POSIX.1-2008 considers this a bug, and implementations are required to search for the first occurrence in the file, regardless.

## Undo

The **undo** description deliberately uses the word “modified”. The **undo** command is not intended to undo commands that replace the contents of the edit buffer, such as **edit**, **next**, **tag**, or **recover**.

Cursor positioning after the **undo** command was inconsistent in the historical *vi*, sometimes attempting to restore the original cursor position (**global**, **undo**, and **v** commands), and sometimes, in the presence of maps, placing the cursor on the last line added or changed instead of the first. POSIX.1-2008 requires a simplified behavior for consistency and simplicity of specification.

## Version

The **version** command cannot be exactly specified since there is no widely-accepted definition of what the version information should contain. Implementations are encouraged to do something reasonably intelligent.

## Write

Historically, the *ex* and *vi* message after a successful **read** or **write** command specified “characters”, not “bytes”. POSIX.1-2008 requires that the number of bytes be displayed, not the number of characters because it may be difficult in multi-byte implementations to determine the number of characters written. Implementations are encouraged to clarify the message displayed to the user.

Implementation-defined tests are permitted so that implementations can make additional checks; for example, for locks or file modification times.

Historically, attempting to append to a nonexistent file caused an error. It has been left unspecified in POSIX.1-2008 to permit implementations to let the **write** succeed, so that the append semantics are similar to those of the historical *csh*.

Historical *vi* permitted empty edit buffers to be written. However, since the way *vi* got around dealing with “empty” files was to always have a line in the edit buffer, no matter what, it wrote them as files of a single, empty line. POSIX.1-2008 does not permit this behavior.

Historically, *ex* restored standard output and standard error to their values as of when *ex* was invoked, before writes to programs were performed. This could disturb the terminal configuration as well as be a security issue for some terminals. POSIX.1-2008 does not permit this, requiring that the program output be captured and displayed as if by the *ex* **print** command.

## Adjust Window

Historically, the line count was set to the value of the **scroll** option if the type character was end-of-file. This feature was broken on most historical implementations long ago, however, and is not documented anywhere. For this reason, POSIX.1-2008 is resolutely silent.

Historically, the **z** command was <blank>-sensitive and **z +** and **z -** did different things than **z+** and **z-** because the type could not be distinguished from a flag. (The commands **z .** and **z =** were historically invalid.) POSIX.1-2008 requires conformance to this historical practice.

Historically, the **z** command was further <blank>-sensitive in that the *count* could not be <blank>-delimited; for example, the commands **z= 5** and **z- 5** were also invalid. Because the *count* is not ambiguous with respect to either the type character or the flags, this is not permitted by POSIX.1-2008.

## Escape

Historically, *ex* filter commands only read the standard output of the commands, letting standard error appear on the terminal as usual. The *vi* utility, however, read both standard output and standard error. POSIX.1-2008 requires the latter behavior for both *ex* and *vi*, for consistency.

## Shift Left and Shift Right

Historically, it was possible to add shift characters to increase the effect of the command; for example, <<< outdented (or >>> indented) the lines 3 levels of indentation instead of the default 1. POSIX.1-2008 requires conformance to historical practice.

**<control>-D**

Historically, the <control>-D command erased the prompt, providing the user with an unbroken presentation of lines from the edit buffer. This is not required by POSIX.1-2008; implementations are encouraged to provide it if possible. Historically, the <control>-D command took, and then ignored, a *count*. POSIX.1-2008 does not permit this behavior.

**Write Line Number**

Historically, the *ex* = command, when executed in *ex* mode in an empty edit buffer, reported 0, and from open or visual mode, reported 1. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

**Execute**

Historically, *ex* did not correctly handle the inclusion of text input commands (that is, **append**, **insert**, and **change**) in executed buffers. POSIX.1-2008 does not permit this exclusion for consistency.

Historically, the logical contents of the buffer being executed did not change if the buffer itself were modified by the commands being executed; that is, buffer execution did not support self-modifying code. POSIX.1-2008 requires conformance to historical practice.

Historically, the @ command took a range of lines, and the @ buffer was executed once per line, with the current line ( ' . ' ) set to each specified line. POSIX.1-2008 requires conformance to historical practice.

Some historical implementations did not notice if errors occurred during buffer execution. This, coupled with the ability to specify a range of lines for the *ex* @ command, makes it trivial to cause them to drop **core**. POSIX.1-2008 requires that implementations stop buffer execution if any error occurs, if the specified line doesn't exist, or if the contents of the edit buffer itself are replaced (for example, the buffer executes the *ex* :**edit** command).

**Regular Expressions in ex**

Historical practice is that the characters in the replacement part of the last **s** command—that is, those matched by entering a ' ~ ' in the regular expression—were not further expanded by the regular expression engine. So, if the characters contained the string "a . , " they would match 'a' followed by " . , " and not 'a' followed by any character. POSIX.1-2008 requires conformance to historical practice.

**Edit Options in ex**

The following paragraphs describe the historical behavior of some edit options that were not, for whatever reason, included in POSIX.1-2008. Implementations are strongly encouraged to only use these names if the functionality described here is fully supported.

**extended** The **extended** edit option has been used in some implementations of *vi* to provide extended regular expressions instead of basic regular expressions. This option was omitted from POSIX.1-2008 because it is not widespread historical practice.

**flash** The **flash** edit option historically caused the screen to flash instead of beeping on error. This option was omitted from POSIX.1-2008 because it is not found in some historical implementations.

|       |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 90306 | <b>hardtabs</b>  | The <b>hardtabs</b> edit option historically defined the number of columns between hardware tab settings. This option was omitted from POSIX.1-2008 because it was believed to no longer be generally useful.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 90307 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90308 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90309 | <b>modeline</b>  | The <b>modeline</b> (sometimes named <b>modelines</b> ) edit option historically caused <i>ex</i> or <i>vi</i> to read the five first and last lines of the file for editor commands. This option is a security problem, and vendors are strongly encouraged to delete it from historical implementations.                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 90310 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90311 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90312 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90313 | <b>open</b>      | The <b>open</b> edit option historically disallowed the <i>ex</i> <b>open</b> and <b>visual</b> commands. This edit option was omitted because these commands are required by POSIX.1-2008.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 90314 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90315 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90316 | <b>optimize</b>  | The <b>optimize</b> edit option historically expedited text throughput by setting the terminal to not do automatic <carriage-return> characters when printing more than one logical line of output. This option was omitted from POSIX.1-2008 because it was intended for terminals without addressable cursors, which are rarely, if ever, still used.                                                                                                                                                                                                                                                                                                                                                                      |
| 90317 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90318 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90319 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90320 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90321 | <b>ruler</b>     | The <b>ruler</b> edit option has been used in some implementations of <i>vi</i> to present a current row/column ruler for the user. This option was omitted from POSIX.1-2008 because it is not widespread historical practice.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90322 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90323 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90324 | <b>sourceany</b> | The <b>sourceany</b> edit option historically caused <i>ex</i> or <i>vi</i> to source start-up files that were owned by users other than the user running the editor. This option is a security problem, and vendors are strongly encouraged to remove it from their implementations.                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 90325 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90326 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90327 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90328 | <b>timeout</b>   | The <b>timeout</b> edit option historically enabled the (now standard) feature of only waiting for a short period before returning keys that could be part of a macro. This feature was omitted from POSIX.1-2008 because its behavior is now standard, it is not widely useful, and it was rarely documented.                                                                                                                                                                                                                                                                                                                                                                                                               |
| 90329 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90330 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90331 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90332 | <b>verbose</b>   | The <b>verbose</b> edit option has been used in some implementations of <i>vi</i> to cause <i>vi</i> to output error messages for common errors; for example, attempting to move the cursor past the beginning or end of the line instead of only alerting the screen. (The historical <i>vi</i> only alerted the terminal and presented no message for such errors. The historical editor option <b>terse</b> did not select when to present error messages, it only made existing error messages more or less verbose.) This option was omitted from POSIX.1-2008 because it is not widespread historical practice; however, implementors are encouraged to use it if they wish to provide error messages for naive users. |
| 90333 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90334 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90335 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90336 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90337 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90338 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90339 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90340 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90341 | <b>wraplen</b>   | The <b>wraplen</b> edit option has been used in some implementations of <i>vi</i> to specify an automatic margin measured from the left margin instead of from the right margin. This is useful when multiple screen sizes are being used to edit a single file. This option was omitted from POSIX.1-2008 because it is not widespread historical practice; however, implementors are encouraged to use it if they add this functionality.                                                                                                                                                                                                                                                                                  |
| 90342 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90343 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90344 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90345 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 90346 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

**autoindent, ai**

Historically, the command **0a** did not do any autoindentation, regardless of the current indentation of line 1. POSIX.1-2008 requires that any indentation present in line 1 be used.

**autoprint, ap**

Historically, the **autoprint** edit option was not completely consistent or based solely on modifications to the edit buffer. Exceptions were the **read** command (when reading from a file, but not from a filter), the **append**, **change**, **insert**, **global**, and **v** commands, all of which were not affected by **autoprint**, and the **tag** command, which was affected by **autoprint**. POSIX.1-2008 requires conformance to historical practice.

Historically, the **autoprint** option only applied to the last of multiple commands entered using <vertical-line> delimiters; for example, **delete** <newline> was affected by **autoprint**, but **delete | version** <newline> was not. POSIX.1-2008 requires conformance to historical practice.

**autowrite, aw**

Appending the '!' character to the *ex* **next** command to avoid performing an automatic write was not supported in historical implementations. POSIX.1-2008 requires that the behavior match the other *ex* commands for consistency.

**ignorecase, ic**

Historical implementations of case-insensitive matching (the **ignorecase** edit option) lead to counter-intuitive situations when uppercase characters were used in range expressions. Historically, the process was as follows:

1. Take a line of text from the edit buffer.
2. Convert uppercase to lowercase in text line.
3. Convert uppercase to lowercase in regular expressions, except in character class specifications.
4. Match regular expressions against text.

This would mean that, with **ignorecase** in effect, the text:

The cat sat on the mat

would be matched by

/^the/

but not by:

/^[A-Z]he/

For consistency with other commands implementing regular expressions, POSIX.1-2008 does not permit this behavior.



### paragraphs, para

The ISO POSIX-2:1993 standard made the default **paragraphs** and **sections** edit options implementation-defined, arguing they were historically oriented to the UNIX system *troff* text formatter, and a “portable user” could use the {, }, [[, ]], (, and ) commands in open or visual mode and have the cursor stop in unexpected places. POSIX.1-2008 specifies their values in the POSIX locale because the unusual grouping (they only work when grouped into two characters at a time) means that they cannot be used for general-purpose movement, regardless.

### readonly

Implementations are encouraged to provide the best possible information to the user as to the read-only status of the file, with the exception that they should not consider the current special privileges of the process. This provides users with a safety net because they must force the overwrite of read-only files, even when running with additional privileges.

The **readonly** edit option specification largely conforms to historical practice. The only difference is that historical implementations did not notice that the user had set the **readonly** edit option in cases where the file was already marked read-only for some reason, and would therefore reinitialize the **readonly** edit option the next time the contents of the edit buffer were replaced. This behavior is disallowed by POSIX.1-2008.

### report

The requirement that lines copied to a buffer interact differently than deleted lines is historical practice. For example, if the **report** edit option is set to 3, deleting 3 lines will cause a report to be written, but 4 lines must be copied before a report is written.

The requirement that the *ex* **global**, **v**, **open**, **undo**, and **visual** commands present reports based on the total number of lines added or deleted during the command execution, and that commands executed by the **global** and **v** commands not present reports, is historical practice. POSIX.1-2008 extends historical practice by requiring that buffer execution be treated similarly. The reasons for this are two-fold. Historically, only the report by the last command executed from the buffer would be seen by the user, as each new report would overwrite the last. In addition, the standard developers believed that buffer execution had more in common with **global** and **v** commands than it did with other *ex* commands, and should behave similarly, for consistency and simplicity of specification.

### showmatch, sm

The length of time the cursor spends on the matching character is unspecified because the timing capabilities of systems are often inexact and variable. The time should be long enough for the user to notice, but not long enough for the user to become annoyed. Some implementations of *vi* have added a **matchtime** option that permits users to set the number of 0,1 second intervals the cursor pauses on the matching character.

### showmode

The **showmode** option has been used in some historical implementations of *ex* and *vi* to display the current editing mode when in open or visual mode. The editing modes have generally included “command” and “input”, and sometimes other modes such as “replace” and “change”. The string was usually displayed on the bottom line of the screen at the far right-hand corner. In addition, a preceding ‘\*’ character often denoted whether the contents of the edit buffer had been modified. The latter display has sometimes been part of the **showmode** option, and sometimes based on another option. This option was not available in the 4 BSD historical

implementation of *vi*, but was viewed as generally useful, particularly to novice users, and is required by POSIX.1-2008.

The **smd** shorthand for the **showmode** option was not present in all historical implementations of the editor. POSIX.1-2008 requires it, for consistency.

Not all historical implementations of the editor displayed a mode string for command mode, differentiating command mode from text input mode by the absence of a mode string. POSIX.1-2008 permits this behavior for consistency with historical practice, but implementations are encouraged to provide a display string for both modes.

### **slowopen**

Historically, the **slowopen** option was automatically set if the terminal baud rate was less than 1 200 baud, or if the baud rate was 1 200 baud and the **redraw** option was not set. The **slowopen** option had two effects. First, when inserting characters in the middle of a line, characters after the cursor would not be pushed ahead, but would appear to be overwritten. Second, when creating a new line of text, lines after the current line would not be scrolled down, but would appear to be overwritten. In both cases, ending text input mode would cause the screen to be refreshed to match the actual contents of the edit buffer. Finally, terminals that were sufficiently intelligent caused the editor to ignore the **slowopen** option. POSIX.1-2008 permits most historical behavior, extending historical practice to require **slowopen** behaviors if the edit option is set by the user.

### **tags**

The default path for tags files is left unspecified as implementations may have their own **tags** implementations that do not correspond to the historical ones. The default **tags** option value should probably at least include the file **./tags**.

### **term**

Historical implementations of *ex* and *vi* ignored changes to the **term** edit option after the initial terminal information was loaded. This is permitted by POSIX.1-2008; however, implementations are encouraged to permit the user to modify their terminal type at any time.

### **terse**

Historically, the **terse** edit option optionally provided a shorter, less descriptive error message, for some error messages. This is permitted, but not required, by POSIX.1-2008. Historically, most common visual mode errors (for example, trying to move the cursor past the end of a line) did not result in an error message, but simply alerted the terminal. Implementations wishing to provide messages for novice users are urged to do so based on the **edit** option **verbose**, and not **terse**.

### **window**

In historical implementations, the default for the **window** edit option was based on the baud rate as follows:

1. If the baud rate was less than 1 200, the **edit** option **w300** set the window value; for example, the line:

```
set w300=12
```

would set the window option to 12 if the baud rate was less than 1 200.



2. If the baud rate was equal to 1 200, the **edit** option **w1200** set the window value.

3. If the baud rate was greater than 1 200, the **edit** option **w9600** set the window value.

The **w300**, **w1200**, and **w9600** options do not appear in POSIX.1-2008 because of their dependence on specific baud rates.

In historical implementations, the size of the window displayed by various commands was related to, but not necessarily the same as, the **window** edit option. For example, the size of the window was set by the *ex* command **visual 10**, but it did not change the value of the **window** edit option. However, changing the value of the **window** edit option did change the number of lines that were displayed when the screen was repainted. POSIX.1-2008 does not permit this behavior in the interests of consistency and simplicity of specification, and requires that all commands that change the number of lines that are displayed do it by setting the value of the **window** edit option.

#### **wrapmargin, wm**

Historically, the **wrapmargin** option did not affect maps inserting characters that also had associated *counts*; for example **:map K 5aABC DEF**. Unfortunately, there are widely used maps that depend on this behavior. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

Historically, **wrapmargin** was calculated using the column display width of all characters on the screen. For example, an implementation using "**^I**" to represent <tab> characters when the **list** edit option was set, where '**^**' and '**I**' each took up a single column on the screen, would calculate the **wrapmargin** based on a value of 2 for each <tab>. The **number** edit option similarly changed the effective length of the line as well. POSIX.1-2008 requires conformance to historical practice.

Earlier versions of this standard allowed for implementations with bytes other than eight bits, but this has been modified in this version.

#### **FUTURE DIRECTIONS**

None.

#### **SEE ALSO**

[Section 2.9.1.1](#) (on page 2357), *ctags*, *ed*, *sed*, *sh*, *stty*, *vi*

[XBD Table 5-1](#) (on page 121), [Chapter 8](#) (on page 173), [Section 9.3](#) (on page 183), [Section 12.2](#) (on page 216)

XSH *access()*

#### **CHANGE HISTORY**

First released in Issue 2.

##### **Issue 5**

The FUTURE DIRECTIONS section is added.

##### **Issue 6**

This utility is marked as part of the User Portability Utilities option.

The obsolescent SYNOPSIS is removed, removing the *+command* and *-* options.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the **map** command description, the sequence *#digit* is added.
- The **directory**, **edcompatible**, **redraw**, and **slowopen** edit options are added.

The *ex* utility is extensively changed for alignment with the IEEE P1003.2b draft standard. This includes changes as a result of the IEEE PASC Interpretations 1003.2 #31, #38, #49, #50, #51, #52, #55, #56, #57, #61, #62, #63, #64, #65, and #78.

The **-l** option is removed.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/23 is applied, correcting a URL.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/8 is applied, making an editorial correction in the EXTENDED DESCRIPTION.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/9 is applied, removing text describing behavior on systems with bytes consisting of more than eight bits.

#### Issue 7

Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying the behavior if an operand is *'-'*.

Austin Group Interpretation 1003.1-2001 #036 is applied, clarifying the behavior for BREs.

Austin Group Interpretation 1003.1-2001 #121 is applied, clarifying the *ex* **write** command.

Austin Group Interpretation 1003.1-2001 #156 is applied.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

90524 **NAME**

90525       expand — convert tabs to spaces

90526 **SYNOPSIS**90527       expand [-t *tablist*] [*file...*]90528 **DESCRIPTION**

90529       The *expand* utility shall write files or the standard input to the standard output with <tab>  
 90530       characters replaced with one or more <space> characters needed to pad to the next tab stop. Any  
 90531       <backspace> characters shall be copied to the output and cause the column position count for  
 90532       tab stop calculations to be decremented; the column position count shall not be decremented  
 90533       below zero.

90534 **OPTIONS**90535       The *expand* utility shall conform to XBD [Section 12.2](#) (on page 216).

90536       The following option shall be supported:

90537       -t *tablist*   Specify the tab stops. The application shall ensure that the argument *tablist* consists  
 90538       of either a single positive decimal integer or a list of tabstops. If a single number is  
 90539       given, tabs shall be set that number of column positions apart instead of the  
 90540       default 8.

90541       If a list of tabstops is given, the application shall ensure that it consists of a list of  
 90542       two or more positive decimal integers, separated by <blank> or <comma>  
 90543       characters, in ascending order. The <tab> characters shall be set at those specific  
 90544       column positions. Each tab stop *N* shall be an integer value greater than zero, and  
 90545       the list is in strictly ascending order. This is taken to mean that, from the start of a  
 90546       line of output, tabbing to position *N* shall cause the next character output to be in  
 90547       the (*N*+1)th column position on that line.

90548       In the event of *expand* having to process a <tab> at a position beyond the last of  
 90549       those specified in a multiple tab-stop list, the <tab> shall be replaced by a single  
 90550       <space> in the output.

90551 **OPERANDS**

90552       The following operand shall be supported:

90553       *file*        The pathname of a text file to be used as input.

90554 **STDIN**

90555       See the INPUT FILES section.

90556 **INPUT FILES**

90557       Input files shall be text files.

90558 **ENVIRONMENT VARIABLES**90559       The following environment variables shall affect the execution of *expand*:

90560       LANG        Provide a default value for the internationalization variables that are unset or null.  
 90561       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 90562       variables used to determine the values of locale categories.)

90563       LC\_ALL      If set to a non-empty string value, override the values of all the other  
 90564       internationalization variables.

90565       LC\_CTYPE    Determine the locale for the interpretation of sequences of bytes of text data as  
 90566       characters (for example, single-byte as opposed to multi-byte characters in  
 90567       arguments and input files), the processing of <tab> and <space> characters, and  
 90568       for the determination of the width in column positions each character would

90569 occupy on an output device.

90570 *LC\_MESSAGES*

90571 Determine the locale that should be used to affect the format and contents of  
90572 diagnostic messages written to standard error.

90573 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

90574 **ASYNCHRONOUS EVENTS**

90575 Default.

90576 **STDOUT**

90577 The standard output shall be equivalent to the input files with <tab> characters converted into  
90578 the appropriate number of <space> characters.

90579 **STDERR**

90580 The standard error shall be used only for diagnostic messages.

90581 **OUTPUT FILES**

90582 None.

90583 **EXTENDED DESCRIPTION**

90584 None.

90585 **EXIT STATUS**

90586 The following exit values shall be returned:

90587 0 Successful completion

90588 >0 An error occurred.

90589 **CONSEQUENCES OF ERRORS**

90590 The *expand* utility shall terminate with an error message and non-zero exit status upon  
90591 encountering difficulties accessing one of the *file* operands.

90592 **APPLICATION USAGE**

90593 None.

90594 **EXAMPLES**

90595 None.

90596 **RATIONALE**

90597 The *expand* utility is useful for preprocessing text files (before sorting, looking at specific  
90598 columns, and so on) that contain <tab> characters.

90599 See XBD [Section 3.103](#) (on page 50).

90600 The *tablist* option-argument consists of integers in ascending order. Utility Syntax Guideline 8  
90601 mandates that *expand* shall accept the integers (within the single argument) separated using  
90602 either <comma> or <blank> characters.

90603 Earlier versions of this standard allowed the following form in the SYNOPSIS:

90604 `expand [-tabstop][-tab1,tab2,...,tabn][file ...]`

90605 This form is no longer specified by POSIX.1-2008 but may be present in some implementations.

90606 **FUTURE DIRECTIONS**

90607 None.

90608 **SEE ALSO**90609 *tabs, unexpand*90610 XBD [Section 3.103](#) (on page 50), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)90611 **CHANGE HISTORY**

90612 First released in Issue 4.

90613 **Issue 6**

90614 This utility is marked as part of the User Portability Utilities option.

90615 The APPLICATION USAGE section is added.

90616 The obsolescent SYNOPSIS is removed.

90617 The *LC\_CTYPE* environment variable description is updated to align with the IEEE P1003.2b  
90618 draft standard.

90619 The normative text is reworded to avoid use of the term “must” for application requirements.

90620 **Issue 7**

90621 Austin Group Interpretation 1003.1-2001 #027 is applied.

90622 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

90623 The *expand* utility is moved from the User Portability Utilities option to the Base. User  
90624 Portability Utilities is now an option for interactive utilities.

90625 **NAME**90626 `expr` — evaluate arguments as an expression90627 **SYNOPSIS**90628 `expr operand...`90629 **DESCRIPTION**90630 The *expr* utility shall evaluate an expression and write the result to standard output.90631 **OPTIONS**

90632 None.

90633 **OPERANDS**90634 The single expression evaluated by *expr* shall be formed from the *operand* operands, as described  
90635 in the EXTENDED DESCRIPTION section. The application shall ensure that each of the  
90636 expression operator symbols:90637 `( ) | & = > >= < <= != + - * / % :`90638 and the symbols *integer* and *string* in the table are provided as separate arguments to *expr*.90639 **STDIN**

90640 Not used.

90641 **INPUT FILES**

90642 None.

90643 **ENVIRONMENT VARIABLES**90644 The following environment variables shall affect the execution of *expr*:90645 **LANG** Provide a default value for the internationalization variables that are unset or null.  
90646 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
90647 variables used to determine the values of locale categories.)90648 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
90649 internationalization variables.90650 **LC\_COLLATE**90651 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
90652 character collating elements within regular expressions and by the string  
90653 comparison operators.90654 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
90655 characters (for example, single-byte as opposed to multi-byte characters in  
90656 arguments) and the behavior of character classes within regular expressions.90657 **LC\_MESSAGES**90658 Determine the locale that should be used to affect the format and contents of  
90659 diagnostic messages written to standard error.90660 **XSI** **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.90661 **ASYNCHRONOUS EVENTS**

90662 Default.

90663 **STDOUT**90664 The *expr* utility shall evaluate the expression and write the result, followed by a <newline>, to  
90665 standard output.

**STDERR**

The standard error shall be used only for diagnostic messages.

**OUTPUT FILES**

None.

**EXTENDED DESCRIPTION**

The formation of the expression to be evaluated is shown in the following table. The symbols *expr*, *expr1*, and *expr2* represent expressions formed from *integer* and *string* symbols and the expression operator symbols (all separate arguments) by recursive application of the constructs described in the table. The expressions are listed in order of decreasing precedence, with equal-precedence operators grouped between horizontal lines. All of the operators shall be left-associative.

| Expression                                                                                                                                                                                | Description                                                                                                                                                                                                                                                                                                                                                                                                |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>integer</i>                                                                                                                                                                            | An argument consisting only of an (optional) unary minus followed by digits.                                                                                                                                                                                                                                                                                                                               |
| <i>string</i>                                                                                                                                                                             | A string argument; see below.                                                                                                                                                                                                                                                                                                                                                                              |
| ( <i>expr</i> )                                                                                                                                                                           | Grouping symbols. Any expression can be placed within parentheses. Parentheses can be nested to a depth of {EXPR_NEST_MAX}.                                                                                                                                                                                                                                                                                |
| <i>expr1</i> : <i>expr2</i>                                                                                                                                                               | Matching expression; see below.                                                                                                                                                                                                                                                                                                                                                                            |
| <i>expr1</i> * <i>expr2</i><br><i>expr1</i> / <i>expr2</i>                                                                                                                                | Multiplication of decimal integer-valued arguments.<br>Integer division of decimal integer-valued arguments, producing an integer result.                                                                                                                                                                                                                                                                  |
| <i>expr1</i> % <i>expr2</i>                                                                                                                                                               | Remainder of integer division of decimal integer-valued arguments.                                                                                                                                                                                                                                                                                                                                         |
| <i>expr1</i> + <i>expr2</i><br><i>expr1</i> - <i>expr2</i>                                                                                                                                | Addition of decimal integer-valued arguments.<br>Subtraction of decimal integer-valued arguments.                                                                                                                                                                                                                                                                                                          |
| <i>expr1</i> = <i>expr2</i><br><i>expr1</i> > <i>expr2</i><br><i>expr1</i> >= <i>expr2</i><br><i>expr1</i> < <i>expr2</i><br><i>expr1</i> <= <i>expr2</i><br><i>expr1</i> != <i>expr2</i> | Returns the result of a decimal integer comparison if both arguments are integers; otherwise, returns the result of a string comparison using the locale-specific collation sequence. The result of each comparison is 1 if the specified relationship is true, or 0 if the relationship is false.<br>Equal.<br>Greater than.<br>Greater than or equal.<br>Less than.<br>Less than or equal.<br>Not equal. |
| <i>expr1</i> & <i>expr2</i>                                                                                                                                                               | Returns the evaluation of <i>expr1</i> if neither expression evaluates to null or zero; otherwise, returns zero.                                                                                                                                                                                                                                                                                           |
| <i>expr1</i>   <i>expr2</i>                                                                                                                                                               | Returns the evaluation of <i>expr1</i> if it is neither null nor zero; otherwise, returns the evaluation of <i>expr2</i> if it is not null; otherwise, zero.                                                                                                                                                                                                                                               |

## Matching Expression

The `' : '` matching operator shall compare the string resulting from the evaluation of *expr1* with the regular expression pattern resulting from the evaluation of *expr2*. Regular expression syntax shall be that defined in XBD [Section 9.3](#) (on page 183), except that all patterns are anchored to the beginning of the string (that is, only sequences starting at the first character of a string are matched by the regular expression) and, therefore, it is unspecified whether `'^'` is a special character in that context. Usually, the matching operator shall return a string representing the number of characters matched (`'0'` on failure). Alternatively, if the pattern contains at least one regular expression subexpression `"[\\(\\.\\.\\.\\)]"`, the string matched by the back-reference expression `"\1"` shall be returned. If the back-reference expression `"\1"` does not match, then the null string shall be returned.

## Identification as Integer or String

An argument or the value of a subexpression that consists only of an optional unary minus followed by digits is a candidate for treatment as an integer if it is used as the left argument to the `|` operator or as either argument to any of the following operators: `& = > >= < <= != + - * / %`. Otherwise, the argument or subexpression value shall be treated as a string.

The use of string arguments **length**, **substr**, **index**, or **match** produces unspecified results.

## EXIT STATUS

The following exit values shall be returned:

- 0 The *expression* evaluates to neither null nor zero.
- 1 The *expression* evaluates to null or zero.
- 2 Invalid *expression*.
- >2 An error occurred.

## CONSEQUENCES OF ERRORS

Default.

## APPLICATION USAGE

The *expr* utility has a rather difficult syntax:

- Many of the operators are also shell control operators or reserved words, so they have to be escaped on the command line.
- Each part of the expression is composed of separate arguments, so liberal usage of `<blank>` characters is required. For example:

| Invalid                   | Valid                          |
|---------------------------|--------------------------------|
| <i>expr</i> 1+2           | <i>expr</i> 1 + 2              |
| <i>expr</i> "1 + 2"       | <i>expr</i> 1 + 2              |
| <i>expr</i> 1 + ( 2 * 3 ) | <i>expr</i> 1 + \ ( 2 \* 3 \ ) |

In many cases, the arithmetic and string features provided as part of the shell command language are easier to use than their equivalents in *expr*. Newly written scripts should avoid *expr* in favor of the new features within the shell; see [Section 2.5](#) (on page 2339) and [Section 2.6.4](#) (on page 2348).

After argument processing by the shell, *expr* is not required to be able to tell the difference between an operator and an operand except by the value. If `"$a"` is `'='`, the command:

```
expr "$a" = '='
```



looks like:

```
expr = = =
```

as the arguments are passed to *expr* (and they all may be taken as the '=' operator). The following works reliably:

```
expr "X$a" = X=
```

Also note that this volume of POSIX.1-2008 permits implementations to extend utilities. The *expr* utility permits the integer arguments to be preceded with a unary minus. This means that an integer argument could look like an option. Therefore, the conforming application must employ the "--" construct of Guideline 10 of XBD [Section 12.2](#) (on page 216) to protect its operands if there is any chance the first operand might be a negative integer (or any string with a leading minus).

For testing string equality the *test* utility is preferred over *expr*, as it is usually implemented as a shell built-in. However, the functionality is not quite the same because the *expr* = and != operators check whether strings collate equally, whereas *test* checks whether they are identical. Therefore, they can produce different results in locales where the collation sequence does not have a total ordering of all characters (see XBD [Section 7.3.2](#), on page 147).

## EXAMPLES

The following command:

```
a=$(expr "$a" + 1)
```

adds 1 to the variable *a*.

The following command, for "\$a" equal to either */usr/abc/file* or just *file*:

```
expr $a : '.*\/(.*\)' \| $a
```

returns the last segment of a pathname (that is, *file*). Applications should avoid the character '/' used alone as an argument; *expr* may interpret it as the division operator.

The following command:

```
expr "//$a" : '.*\/(.*\)'
```

is a better representation of the previous example. The addition of the "/" characters eliminates any ambiguity about the division operator and simplifies the whole expression. Also note that pathnames may contain characters contained in the *IFS* variable and should be quoted to avoid having "\$a" expand into multiple arguments.

The following command:

```
expr "X$VAR" : '.*' - 1
```

returns the number of characters in *VAR*.

## RATIONALE

In an early proposal, EREs were used in the matching expression syntax. This was changed to BREs to avoid breaking historical applications.

The use of a leading <circumflex> in the BRE is unspecified because many historical implementations have treated it as a special character, despite their system documentation. For example:

```
expr foo : ^foo      expr ^foo : ^foo
```

return 3 and 0, respectively, on those systems; their documentation would imply the reverse.

90792 Thus, the anchoring condition is left unspecified to avoid breaking historical scripts relying on  
90793 this undocumented feature.

90794 **FUTURE DIRECTIONS**

90795 None.

90796 **SEE ALSO**

90797 [Section 2.5](#) (on page 2339), [Section 2.6.4](#) (on page 2348)

90798 XBD [Section 7.3.2](#) (on page 147), [Chapter 8](#) (on page 173), [Section 9.3](#) (on page 183), [Section 12.2](#)  
90799 (on page 216)

90800 **CHANGE HISTORY**

90801 First released in Issue 2.

90802 **Issue 5**

90803 The FUTURE DIRECTIONS section is added.

90804 **Issue 6**

90805 The *expr* utility is aligned with the IEEE P1003.2b draft standard, to include resolution of IEEE  
90806 PASC Interpretation 1003.2 #104.

90807 The normative text is reworded to avoid use of the term “must” for application requirements.

90808 **Issue 7**

90809 Austin Group Interpretation 1003.1-2001 #036 is applied, clarifying the behavior for BREs.

90810 The SYNOPSIS and OPERANDS sections are revised to explicitly state that the name of each of  
90811 the operands is *operand*.

**90812 NAME**

90813 false — return false value

**90814 SYNOPSIS**

90815 false

**90816 DESCRIPTION**

90817 The *false* utility shall return with a non-zero exit code.

**90818 OPTIONS**

90819 None.

**90820 OPERANDS**

90821 None.

**90822 STDIN**

90823 Not used.

**90824 INPUT FILES**

90825 None.

**90826 ENVIRONMENT VARIABLES**

90827 None.

**90828 ASYNCHRONOUS EVENTS**

90829 Default.

**90830 STDOUT**

90831 Not used.

**90832 STDERR**

90833 Not used.

**90834 OUTPUT FILES**

90835 None.

**90836 EXTENDED DESCRIPTION**

90837 None.

**90838 EXIT STATUS**

90839 The *false* utility shall always exit with a value other than zero.

**90840 CONSEQUENCES OF ERRORS**

90841 Default.

**90842 APPLICATION USAGE**

90843 None.

**90844 EXAMPLES**

90845 None.

**90846 RATIONALE**

90847 None.

**90848 FUTURE DIRECTIONS**

90849 None.

**90850 SEE ALSO**

90851 *true*

90852 **CHANGE HISTORY**

90853 First released in Issue 2.

90854 **Issue 6**90855 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/24 is applied, changing the STDERR  
90856 section from “None.” to “Not used.” for alignment with [Section 1.4](#) (on page 2326).

90857 **NAME**90858 `fc` — process the command history list90859 **SYNOPSIS**

```
90860 UP fc [-r] [-e editor] [first [last]]
90861 fc -l [-nr] [first [last]]
90862 fc -s [old=new] [first]
```

90863 **DESCRIPTION**

90864 The `fc` utility shall list, or shall edit and re-execute, commands previously entered to an  
 90865 interactive `sh`.

90866 The command history list shall reference commands by number. The first number in the list is  
 90867 selected arbitrarily. The relationship of a number to its command shall not change except when  
 90868 the user logs in and no other process is accessing the list, at which time the system may reset the  
 90869 numbering to start the oldest retained command at another number (usually 1). When the  
 90870 number reaches an implementation-defined upper limit, which shall be no smaller than the  
 90871 value in `HISTSIZE` or 32767 (whichever is greater), the shell may wrap the numbers, starting the  
 90872 next command with a lower number (usually 1). However, despite this optional wrapping of  
 90873 numbers, `fc` shall maintain the time-ordering sequence of the commands. For example, if four  
 90874 commands in sequence are given the numbers 32766, 32767, 1 (wrapped), and 2 as they are  
 90875 executed, command 32767 is considered the command previous to 1, even though its number is  
 90876 higher.

90877 When commands are edited (when the `-l` option is not specified), the resulting lines shall be  
 90878 entered at the end of the history list and then re-executed by `sh`. The `fc` command that caused the  
 90879 editing shall not be entered into the history list. If the editor returns a non-zero exit status, this  
 90880 shall suppress the entry into the history list and the command re-execution. Any command line  
 90881 variable assignments or redirection operators used with `fc` shall affect both the `fc` command itself  
 90882 as well as the command that results; for example:

```
90883 fc -s -- -l 2>/dev/null
```

90884 reinvokes the previous command, suppressing standard error for both `fc` and the previous  
 90885 command.

90886 **OPTIONS**

90887 The `fc` utility shall conform to XBD [Section 12.2](#) (on page 216).

90888 The following options shall be supported:

- 90889 **-e editor** Use the editor named by *editor* to edit the commands. The *editor* string is a utility  
 90890 name, subject to search via the `PATH` variable (see XBD [Chapter 8](#), on page 173).  
 90891 The value in the `FCEDIT` variable shall be used as a default when `-e` is not  
 90892 specified. If `FCEDIT` is null or unset, `ed` shall be used as the editor.
- 90893 **-l** (The letter ell.) List the commands rather than invoking an editor on them. The  
 90894 commands shall be written in the sequence indicated by the *first* and *last* operands,  
 90895 as affected by `-r`, with each command preceded by the command number.
- 90896 **-n** Suppress command numbers when listing with `-l`.
- 90897 **-r** Reverse the order of the commands listed (with `-l`) or edited (with neither `-l` nor  
 90898 `-s`).



90939 **STDIN**

90940 Not used.

90941 **INPUT FILES**

90942 None.

90943 **ENVIRONMENT VARIABLES**90944 The following environment variables shall affect the execution of *fc*:

90945 *FCEDIT* This variable, when expanded by the shell, shall determine the default value for  
 90946 the *-e editor* option's *editor* option-argument. If *FCEDIT* is null or unset, *ed* shall be  
 90947 used as the editor.

90948 *HISTFILE* Determine a pathname naming a command history file. If the *HISTFILE* variable is  
 90949 not set, the shell may attempt to access or create a file *.sh\_history* in the directory  
 90950 referred to by the *HOME* environment variable. If the shell cannot obtain both read  
 90951 and write access to, or create, the history file, it shall use an unspecified  
 90952 mechanism that allows the history to operate properly. (References to history "file"  
 90953 in this section shall be understood to mean this unspecified mechanism in such  
 90954 cases.) An implementation may choose to access this variable only when  
 90955 initializing the history file; this initialization shall occur when *fc* or *sh* first attempt  
 90956 to retrieve entries from, or add entries to, the file, as the result of commands issued  
 90957 by the user, the file named by the *ENV* variable, or implementation-defined system  
 90958 start-up files. In some historical shells, the history file is initialized just after the  
 90959 *ENV* file has been processed. Therefore, it is implementation-defined whether  
 90960 changes made to *HISTFILE* after the history file has been initialized are effective.  
 90961 Implementations may choose to disable the history list mechanism for users with  
 90962 appropriate privileges who do not set *HISTFILE*; the specific circumstances under  
 90963 which this occurs are implementation-defined. If more than one instance of the  
 90964 shell is using the same history file, it is unspecified how updates to the history file  
 90965 from those shells interact. As entries are deleted from the history file, they shall be  
 90966 deleted oldest first. It is unspecified when history file entries are physically  
 90967 removed from the history file.

90968 *HISTSIZE* Determine a decimal number representing the limit to the number of previous  
 90969 commands that are accessible. If this variable is unset, an unspecified default  
 90970 greater than or equal to 128 shall be used. The maximum number of commands in  
 90971 the history list is unspecified, but shall be at least 128. An implementation may  
 90972 choose to access this variable only when initializing the history file, as described  
 90973 under *HISTFILE*. Therefore, it is unspecified whether changes made to *HISTSIZE*  
 90974 after the history file has been initialized are effective.

90975 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 90976 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 90977 variables used to determine the values of locale categories.)

90978 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 90979 internationalization variables.

90980 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 90981 characters (for example, single-byte as opposed to multi-byte characters in  
 90982 arguments and input files).

90983 *LC\_MESSAGES*

90984 Determine the locale that should be used to affect the format and contents of  
 90985 diagnostic messages written to standard error.

90986 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

90987 **ASYNCHRONOUS EVENTS**

90988 Default.

90989 **STDOUT**

90990 When the **-l** option is used to list commands, the format of each command in the list shall be as

90991 follows:

90992 "%d\t%s\n", <line number>, <command>

90993 If both the **-l** and **-n** options are specified, the format of each command shall be:

90994 "\t%s\n", <command>

90995 If the <command> consists of more than one line, the lines after the first shall be displayed as:

90996 "\t%s\n", <continued-command>

90997 **STDERR**

90998 The standard error shall be used only for diagnostic messages.

90999 **OUTPUT FILES**

91000 None.

91001 **EXTENDED DESCRIPTION**

91002 None.

91003 **EXIT STATUS**

91004 The following exit values shall be returned:

91005 0 Successful completion of the listing.

91006 >0 An error occurred.

91007 Otherwise, the exit status shall be that of the commands executed by *fc*.

91008 **CONSEQUENCES OF ERRORS**

91009 Default.

91010 **APPLICATION USAGE**

91011 Since editors sometimes use file descriptors as integral parts of their editing, redirecting their file

91012 descriptors as part of the *fc* command can produce unexpected results. For example, if *vi* is the

91013 *FCEDIT* editor, the command:

91014 *fc -s | more*

91015 does not work correctly on many systems.

91016 Users on windowing systems may want to have separate history files for each window by

91017 setting *HISTFILE* as follows:

91018 *HISTFILE=\$HOME/.sh\_hist\$\$*

91019 **EXAMPLES**

91020 None.

91021 **RATIONALE**

91022 This utility is based on the *fc* built-in of the KornShell.

91023 An early proposal specified the **-e** option as [**-e** *editor* [*old= new* ]], which is not historical

91024 practice. Historical practice in *fc* of either [**-e** *editor*] or [**-e** - [*old= new* ]] is acceptable, but not

91025 both together. To clarify this, a new option **-s** was introduced replacing the [**-e** -]. This resolves



the conflict and makes *fc* conform to the Utility Syntax Guidelines.

**HISTFILE** Some implementations of the KornShell check for the superuser and do not create a history file unless *HISTFILE* is set. This is done primarily to avoid creating unlinked files in the root file system when logging in during single-user mode. *HISTFILE* must be set for the superuser to have history.

**HISTSIZE** Needed to limit the size of history files. It is the intent of the standard developers that when two shells share the same history file, commands that are entered in one shell shall be accessible by the other shell. Because of the difficulties of synchronization over a network, the exact nature of the interaction is unspecified.

The initialization process for the history file can be dependent on the system start-up files, in that they may contain commands that effectively preempt the settings the user has for *HISTFILE* and *HISTSIZE*. For example, function definition commands are recorded in the history file. If the system administrator includes function definitions in some system start-up file called before the *ENV* file, the history file is initialized before the user can influence its characteristics. In some historical shells, the history file is initialized just after the *ENV* file has been processed. Because of these situations, the text requires the initialization process to be implementation-defined.

Consideration was given to omitting the *fc* utility in favor of the command line editing feature in *sh*. For example, in *vi* editing mode, typing "<ESC> v" is equivalent to:

```
EDITOR=vi fc
```

However, the *fc* utility allows the user the flexibility to edit multiple commands simultaneously (such as *fc 10 20*) and to use editors other than those supported by *sh* for command line editing.

In the KornShell, the alias *r* ("re-do") is preset to *fc -e -* (equivalent to the POSIX *fc -s*). This is probably an easier command name to remember than *fc* ("fix command"), but it does not meet the Utility Syntax Guidelines. Renaming *fc* to *hist* or *redo* was considered, but since this description closely matches historical KornShell practice already, such a renaming was seen as gratuitous. Users are free to create aliases whenever odd historical names such as *fc*, *awk*, *cat*, *grep*, or *yacc* are standardized by POSIX.

Command numbers have no ordering effects; they are like serial numbers. The *-r* option and *-number* operand address the sequence of command execution, regardless of serial numbers. So, for example, if the command number wrapped back to 1 at some arbitrary point, there would be no ambiguity associated with traversing the wrap point. For example, if the command history were:

```
32766: echo 1
32767: echo 2
1: echo 3
```

the number *-2* refers to command 32767 because it is the second previous command, regardless of serial number.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*sh*

XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

**91068 CHANGE HISTORY**

91069 First released in Issue 4.

**91070 Issue 5**

91071 The FUTURE DIRECTIONS section is added.

**91072 Issue 6**

91073 This utility is marked as part of the User Portability Utilities option.

91074 In the ENVIRONMENT VARIABLES section, the text “user’s home directory” is updated to  
91075 “directory referred to by the *HOME* environment variable”.

**91076 Issue 7**

91077 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

91078 **NAME**91079 `fg` — run jobs in the foreground91080 **SYNOPSIS**91081 UP `fg [job_id]`91082 **DESCRIPTION**

91083 If job control is enabled (see the description of `set -m`), the `fg` utility shall move a background job  
 91084 from the current environment (see [Section 2.12](#), on page 2371) into the foreground.

91085 Using `fg` to place a job into the foreground shall remove its process ID from the list of those  
 91086 “known in the current shell execution environment”; see [Section 2.9.3.1](#) (on page 2360).

91087 **OPTIONS**

91088 None.

91089 **OPERANDS**

91090 The following operand shall be supported:

91091 *job\_id* Specify the job to be run as a foreground job. If no *job\_id* operand is given, the  
 91092 *job\_id* for the job that was most recently suspended, placed in the background, or  
 91093 run as a background job shall be used. The format of *job\_id* is described in XBD  
 91094 [Section 3.204](#) (on page 66).

91095 **STDIN**

91096 Not used.

91097 **INPUT FILES**

91098 None.

91099 **ENVIRONMENT VARIABLES**91100 The following environment variables shall affect the execution of `fg`:

91101 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 91102 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 91103 variables used to determine the values of locale categories.)

91104 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 91105 internationalization variables.

91106 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 91107 characters (for example, single-byte as opposed to multi-byte characters in  
 91108 arguments).

91109 *LC\_MESSAGES*

91110 Determine the locale that should be used to affect the format and contents of  
 91111 diagnostic messages written to standard error.

91112 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.91113 **ASYNCHRONOUS EVENTS**

91114 Default.

91115 **STDOUT**91116 The `fg` utility shall write the command line of the job to standard output in the following format:91117 `"%s\n", <command>`

**91118 STDERR**

91119 The standard error shall be used only for diagnostic messages.

**91120 OUTPUT FILES**

91121 None.

**91122 EXTENDED DESCRIPTION**

91123 None.

**91124 EXIT STATUS**

91125 The following exit values shall be returned:

91126 0 Successful completion.

91127 >0 An error occurred.

**91128 CONSEQUENCES OF ERRORS**

91129 If job control is disabled, the *fg* utility shall exit with an error and no job shall be placed in the  
91130 foreground.

**91131 APPLICATION USAGE**

91132 The *fg* utility does not work as expected when it is operating in its own utility execution  
91133 environment because that environment has no applicable jobs to manipulate. See the  
91134 APPLICATION USAGE section for *bg*. For this reason, *fg* is generally implemented as a shell  
91135 regular built-in.

**91136 EXAMPLES**

91137 None.

**91138 RATIONALE**

91139 The extensions to the shell specified in this volume of POSIX.1-2008 have mostly been based on  
91140 features provided by the KornShell. The job control features provided by *bg*, *fg*, and *jobs* are also  
91141 based on the KornShell. The standard developers examined the characteristics of the C shell  
91142 versions of these utilities and found that differences exist. Despite widespread use of the C shell,  
91143 the KornShell versions were selected for this volume of POSIX.1-2008 to maintain a degree of  
91144 uniformity with the rest of the KornShell features selected (such as the very popular command  
91145 line editing features).

**91146 FUTURE DIRECTIONS**

91147 None.

**91148 SEE ALSO**

91149 [Section 2.9.3.1](#) (on page 2360), [Section 2.12](#) (on page 2371), *bg*, *kill*, *jobs*, *wait*

91150 XBD [Section 3.204](#) (on page 66), [Chapter 8](#) (on page 173)

**91151 CHANGE HISTORY**

91152 First released in Issue 4.

**91153 Issue 6**

91154 This utility is marked as part of the User Portability Utilities option.

91155 The APPLICATION USAGE section is added.

91156 The JC marking is removed from the SYNOPSIS since job control is mandatory in this version.

**NAME**

file — determine file type

**SYNOPSIS**

file [-dh] [-M *file*] [-m *file*] *file*...

file -i [-h] *file*...

**DESCRIPTION**

The *file* utility shall perform a series of tests in sequence on each specified *file* in an attempt to classify it:

1. If *file* does not exist, cannot be read, or its file status could not be determined, the output shall indicate that the file was processed, but that its type could not be determined.
2. If the file is not a regular file, its file type shall be identified. The file types directory, FIFO, socket, block special, and character special shall be identified as such. Other implementation-defined file types may also be identified. If *file* is a symbolic link, by default the link shall be resolved and *file* shall test the type of file referenced by the symbolic link. (See the **-h** and **-i** options below.)
3. If the length of *file* is zero, it shall be identified as an empty file.
4. The *file* utility shall examine an initial segment of *file* and shall make a guess at identifying its contents based on position-sensitive tests. (The answer is not guaranteed to be correct; see the **-d**, **-M**, and **-m** options below.)
5. The *file* utility shall examine *file* and make a guess at identifying its contents based on context-sensitive default system tests. (The answer is not guaranteed to be correct.)
6. The file shall be identified as a data file.

If *file* does not exist, cannot be read, or its file status could not be determined, the output shall indicate that the file was processed, but that its type could not be determined.

If *file* is a symbolic link, by default the link shall be resolved and *file* shall test the type of file referenced by the symbolic link.

**OPTIONS**

The *file* utility shall conform to XBD [Section 12.2](#) (on page 216), except that the order of the **-m**, **-d**, and **-M** options shall be significant.

The following options shall be supported by the implementation:

- |                       |                                                                                                                                                                                                                                                                                                              |
|-----------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>-d</b>             | Apply any position-sensitive default system tests and context-sensitive default system tests to the file. This is the default if no <b>-M</b> or <b>-m</b> option is specified.                                                                                                                              |
| <b>-h</b>             | When a symbolic link is encountered, identify the file as a symbolic link. If <b>-h</b> is not specified and <i>file</i> is a symbolic link that refers to a nonexistent file, <i>file</i> shall identify the file as a symbolic link, as if <b>-h</b> had been specified.                                   |
| <b>-i</b>             | If a file is a regular file, do not attempt to classify the type of the file further, but identify the file as specified in the STDOUT section.                                                                                                                                                              |
| <b>-M <i>file</i></b> | Specify the name of a file containing position-sensitive tests that shall be applied to a file in order to classify it (see the EXTENDED DESCRIPTION). No position-sensitive default system tests nor context-sensitive default system tests shall be applied unless the <b>-d</b> option is also specified. |

91198        **-m file**        Specify the name of a file containing position-sensitive tests that shall be applied to  
 91199                        a file in order to classify it (see the EXTENDED DESCRIPTION).

91200        If the **-m** option is specified without specifying the **-d** option or the **-M** option, position-  
 91201        sensitive default system tests shall be applied after the position-sensitive tests specified by the  
 91202        **-m** option. If the **-M** option is specified with the **-d** option, the **-m** option, or both, or the **-m**  
 91203        option is specified with the **-d** option, the concatenation of the position-sensitive tests specified  
 91204        by these options shall be applied in the order specified by the appearance of these options. If a  
 91205        **-M** or **-m file** option-argument is **-**, the results are unspecified.

## 91206 OPERANDS

91207        The following operand shall be supported:

91208        *file*            A pathname of a file to be tested.

## 91209 STDIN

91210        The standard input shall be used if a *file* operand is **'-'** and the implementation treats the **'-'**  
 91211        as meaning standard input. Otherwise, the standard input shall not be used.

## 91212 INPUT FILES

91213        The *file* can be any file type.

## 91214 ENVIRONMENT VARIABLES

91215        The following environment variables shall affect the execution of *file*:

91216        *LANG*            Provide a default value for the internationalization variables that are unset or null.  
 91217                        (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 91218                        variables used to determine the values of locale categories.)

91219        *LC\_ALL*          If set to a non-empty string value, override the values of all the other  
 91220                        internationalization variables.

91221        *LC\_CTYPE*       Determine the locale for the interpretation of sequences of bytes of text data as  
 91222                        characters (for example, single-byte as opposed to multi-byte characters in  
 91223                        arguments and input files).

91224        *LC\_MESSAGES*

91225                        Determine the locale that should be used to affect the format and contents of  
 91226                        diagnostic messages written to standard error and informative messages written to  
 91227                        standard output.

91228        *XSI NLSPATH*    Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 91229 ASYNCHRONOUS EVENTS

91230        Default.

## 91231 STDOUT

91232        In the POSIX locale, the following format shall be used to identify each operand, *file* specified:

91233        "%s: %s\n", <file>, <type>

91234        The values for <type> are unspecified, except that in the POSIX locale, if *file* is identified as one  
 91235        of the types listed in the following table, <type> shall contain (but is not limited to) the  
 91236        corresponding string, unless the file is identified by a position-sensitive test specified by a **-M** or  
 91237        **-m** option. Each <space> shown in the strings shall be exactly one <space>.

91238

**Table 4-9** File Utility Output Strings

|       | <b>If <i>file</i> is:</b>                                    | <b>&lt;type&gt; shall contain the string:</b> | <b>Notes</b> |
|-------|--------------------------------------------------------------|-----------------------------------------------|--------------|
| 91239 | Nonexistent                                                  | cannot open                                   |              |
| 91240 |                                                              |                                               |              |
| 91241 | Block special                                                | block special                                 | 1            |
| 91242 | Character special                                            | character special                             | 1            |
| 91243 | Directory                                                    | directory                                     | 1            |
| 91244 | FIFO                                                         | fifo                                          | 1            |
| 91245 | Socket                                                       | socket                                        | 1            |
| 91246 | Symbolic link                                                | symbolic link to                              | 1            |
| 91247 | Regular file                                                 | regular file                                  | 1,2          |
| 91248 | Empty regular file                                           | empty                                         | 3            |
| 91249 | Regular file that cannot be read                             | cannot open                                   | 3            |
| 91250 | Executable binary                                            | executable                                    | 3,4,6        |
| 91251 | <i>ar</i> archive library (see <i>ar</i> )                   | archive                                       | 3,4,6        |
| 91252 | Extended <i>cpio</i> format (see <i>pax</i> )                | cpio archive                                  | 3,4,6        |
| 91253 | Extended <i>tar</i> format (see <b>ustar</b> in <i>pax</i> ) | tar archive                                   | 3,4,6        |
| 91254 | Shell script                                                 | commands text                                 | 3,5,6        |
| 91255 | C-language source                                            | c program text                                | 3,5,6        |
| 91256 | FORTRAN source                                               | fortran program text                          | 3,5,6        |
| 91257 | Regular file whose type cannot be determined                 | data                                          | 3            |

91258

**Notes:**

91259

1. This is a file type test.

91260

2. This test is applied only if the **-i** option is specified.

91261

3. This test is applied only if the **-i** option is not specified.

91262

4. This is a position-sensitive default system test.

91263

5. This is a context-sensitive default system test.

91264

6. Position-sensitive default system tests and context-sensitive default system tests are not applied if the **-M** option is specified unless the **-d** option is also specified.

91265

91266

In the POSIX locale, if *file* is identified as a symbolic link (see the **-h** option), the following alternative output format shall be used:

91267

91268

```
"%s: %s %s\n", <file>, <type>, <contents of link>"
```

91269

If the file named by the *file* operand does not exist, cannot be read, or the type of the file named by the *file* operand cannot be determined, this shall not be considered an error that affects the exit status.

91270

91271

91272

**STDERR**

91273

The standard error shall be used only for diagnostic messages.

91274

**OUTPUT FILES**

91275

None.



## EXTENDED DESCRIPTION

A file specified as an option-argument to the **-m** or **-M** options shall contain one position-sensitive test per line, which shall be applied to the file. If the test succeeds, the message field of the line shall be printed and no further tests shall be applied, with the exception that tests on immediately following lines beginning with a single '**>**' character shall be applied.

Each line shall be composed of the following four <tab>-separated fields. (Implementations may allow any combination of one or more white-space characters other than <newline> to act as field separators.)

*offset*      An unsigned number (optionally preceded by a single '**>**' character) specifying the *offset*, in bytes, of the value in the file that is to be compared against the *value* field of the line. If the file is shorter than the specified offset, the test shall fail.

If the *offset* begins with the character '**>**', the test contained in the line shall not be applied to the file unless the test on the last line for which the *offset* did not begin with a '**>**' was successful. By default, the *offset* shall be interpreted as an unsigned decimal number. With a leading 0x or 0X, the *offset* shall be interpreted as a hexadecimal number; otherwise, with a leading 0, the *offset* shall be interpreted as an octal number.

*type*      The type of the value in the file to be tested. The type shall consist of the type specification characters **d**, **s**, and **u**, specifying signed decimal, string, and unsigned decimal, respectively.

The *type* string shall be interpreted as the bytes from the file starting at the specified *offset* and including the same number of bytes specified by the *value* field. If insufficient bytes remain in the file past the *offset* to match the *value* field, the test shall fail.

The type specification characters **d** and **u** can be followed by an optional unsigned decimal integer that specifies the number of bytes represented by the type. The type specification characters **d** and **u** can be followed by an optional **C**, **S**, **I**, or **L**, indicating that the value is of type **char**, **short**, **int**, or **long**, respectively.

The default number of bytes represented by the type specifiers **d**, **f**, and **u** shall correspond to their respective C-language types as follows. If the system claims conformance to the C-Language Development Utilities option, those specifiers shall correspond to the default sizes used in the *c99* utility. Otherwise, the default sizes shall be implementation-defined.

For the type specifier characters **d** and **u**, the default number of bytes shall correspond to the size of a basic integer type of the implementation. For these specifier characters, the implementation shall support values of the optional number of bytes to be converted corresponding to the number of bytes in the C-language types **char**, **short**, **int**, or **long**. These numbers can also be specified by an application as the characters **C**, **S**, **I**, and **L**, respectively. The byte order used when interpreting numeric values is implementation-defined, but shall correspond to the order in which a constant of the corresponding type is stored in memory on the system.

All type specifiers, except for **s**, can be followed by a mask specifier of the form **&number**. The mask value shall be AND'ed with the value of the input file before the comparison with the *value* field of the line is made. By default, the mask shall be interpreted as an unsigned decimal number. With a leading 0x or 0X, the mask shall be interpreted as an unsigned hexadecimal number; otherwise, with a leading



|       |                   |                                                                                                                  |
|-------|-------------------|------------------------------------------------------------------------------------------------------------------|
| 91323 |                   | 0, the mask shall be interpreted as an unsigned octal number.                                                    |
| 91324 |                   | The strings <b>byte</b> , <b>short</b> , <b>long</b> , and <b>string</b> shall also be supported as type fields, |
| 91325 |                   | being interpreted as dC, dS, dL, and s, respectively.                                                            |
| 91326 | <i>value</i>      | The <i>value</i> to be compared with the value from the file.                                                    |
| 91327 |                   | If the specifier from the type field is s or <b>string</b> , then interpret the value as a string.               |
| 91328 |                   | Otherwise, interpret it as a number. If the value is a string, then the test shall                               |
| 91329 |                   | succeed only when a string value exactly matches the bytes from the file.                                        |
| 91330 |                   | If the <i>value</i> is a string, it can contain the following sequences:                                         |
| 91331 | <i>\character</i> | The <backslash>-escape sequences as specified in XBD Table 5-1                                                   |
| 91332 |                   | (on page 121) ('\\', '\a', '\b', '\f', '\n', '\r', '\t',                                                         |
| 91333 |                   | '\v'). In addition, the escape sequence '\ ' (the <backslash>                                                    |
| 91334 |                   | character followed by a <space> character) shall be recognized to                                                |
| 91335 |                   | represent a <space> character. The results of using any other                                                    |
| 91336 |                   | character, other than an octal digit, following the <backslash>                                                  |
| 91337 |                   | are unspecified.                                                                                                 |
| 91338 | <i>\octal</i>     | Octal sequences that can be used to represent characters with                                                    |
| 91339 |                   | specific coded values. An octal sequence shall consist of a                                                      |
| 91340 |                   | <backslash> followed by the longest sequence of one, two, or                                                     |
| 91341 |                   | three octal-digit characters (01234567).                                                                         |
| 91342 |                   | By default, any value that is not a string shall be interpreted as a signed decimal                              |
| 91343 |                   | number. Any such value, with a leading 0x or 0X, shall be interpreted as an                                      |
| 91344 |                   | unsigned hexadecimal number; otherwise, with a leading zero, the value shall be                                  |
| 91345 |                   | interpreted as an unsigned octal number.                                                                         |
| 91346 |                   | If the value is not a string, it can be preceded by a character indicating the                                   |
| 91347 |                   | comparison to be performed. Permissible characters and the comparisons they                                      |
| 91348 |                   | specify are as follows:                                                                                          |
| 91349 | =                 | The test shall succeed if the value from the file equals the <i>value</i> field.                                 |
| 91350 | <                 | The test shall succeed if the value from the file is less than the <i>value</i> field.                           |
| 91351 | >                 | The test shall succeed if the value from the file is greater than the <i>value</i> field.                        |
| 91352 | &                 | The test shall succeed if all of the set bits in the <i>value</i> field are set in the value                     |
| 91353 |                   | from the file.                                                                                                   |
| 91354 | ^                 | The test shall succeed if at least one of the set bits in the <i>value</i> field is not set in                   |
| 91355 |                   | the value from the file.                                                                                         |
| 91356 | x                 | The test shall succeed if the file is large enough to contain a value of the type                                |
| 91357 |                   | specified starting at the offset specified.                                                                      |
| 91358 | <i>message</i>    | The <i>message</i> to be printed if the test succeeds. The <i>message</i> shall be interpreted                   |
| 91359 |                   | using the notation for the <i>printf</i> formatting specification; see <i>printf</i> . If the <i>value</i>       |
| 91360 |                   | field was a string, then the value from the file shall be the argument for the <i>printf</i>                     |
| 91361 |                   | formatting specification; otherwise, the value from the file shall be the argument.                              |

**EXIT STATUS**

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

**CONSEQUENCES OF ERRORS**

Default.

**APPLICATION USAGE**

The *file* utility can only be required to guess at many of the file types because only exhaustive testing can determine some types with certainty. For example, binary data on some implementations might match the initial segment of an executable or a *tar* archive.

Note that the table indicates that the output contains the stated string. Systems may add text before or after the string. For executables, as an example, the machine architecture and various facts about how the file was link-edited may be included. Note also that on systems that recognize shell script files starting with "#!" as executable files, these may be identified as executable binary files rather than as shell scripts.

**EXAMPLES**

Determine whether an argument is a binary executable file:

```
file -- "$1" | grep -q '.*executable' &&  
printf "%s is executable.\n" "$1"
```

**RATIONALE**

The *-f* option was omitted because the same effect can (and should) be obtained using the *xargs* utility.

Historical versions of the *file* utility attempt to identify the following types of files: symbolic link, directory, character special, block special, socket, *tar* archive, *cpio* archive, SCCS archive, archive library, empty, *compress* output, *pack* output, binary data, C source, FORTRAN source, assembler source, *nroff*/*troff*/*eqn*/*tbl* source *troff* output, shell script, C shell script, English text, ASCII text, various executables, APL workspace, compiled terminfo entries, and CURSES screen images. Only those types that are reasonably well specified in POSIX or are directly related to POSIX utilities are listed in the table.

Historical systems have used a "magic file" named */etc/magic* to help identify file types. Because it is generally useful for users and scripts to be able to identify special file types, the *-m* flag and a portable format for user-created magic files has been specified. No requirement is made that an implementation of *file* use this method of identifying files, only that users be permitted to add their own classifying tests.

In addition, three options have been added to historical practice. The *-d* flag has been added to permit users to cause their tests to follow any default system tests. The *-i* flag has been added to permit users to test portably for regular files in shell scripts. The *-M* flag has been added to permit users to ignore any default system tests.

The POSIX.1-2008 description of default system tests and the interaction between the *-d*, *-M*, and *-m* options did not clearly indicate that there were two types of "default system tests". The "position-sensitive tests" determine file types by looking for certain string or binary values at specific offsets in the file being examined. These position-sensitive tests were implemented in historical systems using the magic file described above. Some of these tests are now built into the *file* utility itself on some implementations so the output can provide more detail than can be provided by magic files. For example, a magic file can easily identify a **core** file on most implementations, but cannot name the program file that dropped the core. A magic file could

produce output such as:

```
/home/dwc/core: ELF 32-bit MSB core file SPARC Version 1
```

but by building the test into the *file* utility, you could get output such as:

```
/home/dwc/core: ELF 32-bit MSB core file SPARC Version 1, from 'testprog'
```

These extended built-in tests are still to be treated as position-sensitive default system tests even if they are not listed in */etc/magic* or any other magic file.

The context-sensitive default system tests were always built into the *file* utility. These tests looked for language constructs in text files trying to identify shell scripts, C, FORTRAN, and other computer language source files, and even plain text files. With the addition of the *-m* and *-M* options the distinction between position-sensitive and context-sensitive default system tests became important because the order of testing is important. The context-sensitive system default tests should never be applied before any position-sensitive tests even if the *-d* option is specified before a *-m* option or *-M* option due to the high probability that the context-sensitive system default tests will incorrectly identify arbitrary text files as text files before position-sensitive tests specified by the *-m* or *-M* option would be applied to give a more accurate identification.

Leaving the meaning of *-M* – and *-m* – unspecified allows an existing prototype of these options to continue to work in a backwards-compatible manner. (In that implementation, *-M* – was roughly equivalent to *-d* in POSIX.1-2008.)

The historical *-c* option was omitted as not particularly useful to users or portable shell scripts. In addition, a reasonable implementation of the *file* utility would report any errors found each time the magic file is read.

The historical format of the magic file was the same as that specified by the Rationale in the ISO POSIX-2:1993 standard for the *offset*, *value*, and *message* fields; however, it used less precise type fields than the format specified by the current normative text. The new type field values are a superset of the historical ones.

The following is an example magic file:

```
0 short      070707      cpio archive
0 short      0143561     Byte-swapped cpio archive
0 string     070707      ASCII cpio archive
0 long       0177555     Very old archive
0 short      0177545     Old archive
0 short      017437      Old packed data
0 string     \037\036     Packed data
0 string     \377\037     Compacted data
0 string     \037\235     Compressed data
>2 byte&0x80 >0         Block compressed
>2 byte&0x1f x          %d bits
0 string     \032\001     Compiled Terminfo Entry
0 short      0433         Curses screen image
0 short      0434         Curses screen image
0 string     <ar>         System V Release 1 archive
0 string     !<arch>\n___.SYMDEF Archive random library
0 string     !<arch>      Archive
0 string     ARF_BEGARF   PHIGS clear text archive
0 long       0x137A2950   Scalable OpenFont binary
0 long       0x137A2951   Encrypted scalable OpenFont binary
```

91454 The use of a basic integer data type is intended to allow the implementation to choose a word  
91455 size commonly used by applications on that architecture.

91456 Earlier versions of this standard allowed for implementations with bytes other than eight bits,  
91457 but this has been modified in this version.

#### 91458 **FUTURE DIRECTIONS**

91459 None.

#### 91460 **SEE ALSO**

91461 *ar, ls, pax, printf*

91462 XBD [Table 5-1](#) (on page 121), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

#### 91463 **CHANGE HISTORY**

91464 First released in Issue 4.

#### 91465 **Issue 6**

91466 This utility is marked as part of the User Portability Utilities option.

91467 Options and an EXTENDED DESCRIPTION are added as specified in the IEEE P1003.2b draft  
91468 standard.

91469 IEEE PASC Interpretations 1003.2 #192 and #178 are applied.

91470 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/25 is applied, making major changes to  
91471 address ambiguities raised in defect reports.

91472 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/26 is applied, making it clear in the  
91473 OPTIONS section that the **-m**, **-d**, and **-M** options do not comply with Guideline 11 of the  
91474 Utility Syntax Guidelines.

91475 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/10 is applied, clarifying the specification  
91476 characters.

91477 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/11 is applied, allowing application  
91478 developers to create portable magic files that can match characters in strings, and allowing  
91479 common extensions found in existing implementations.

91480 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/12 is applied, removing text describing  
91481 behavior on systems with bytes consisting of more than eight bits.

#### 91482 **Issue 7**

91483 Austin Group Interpretation 1003.1-2001 #092 is applied.

91484 SD5-XCU-ERN-4 is applied, adding further entries in the Notes column in [Table 4-9](#).

91485 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

91486 The *file* utility is moved from the User Portability Utilities option to the Base. User Portability  
91487 Utilities is now an option for interactive utilities.

91488 The EXAMPLES section is revised to correct an error with the pathname "\$1".

91489 **NAME**

91490 find — find files

91491 **SYNOPSIS**91492 find [-H|-L] *path...* [*operand\_expression...*]91493 **DESCRIPTION**

91494 The *find* utility shall recursively descend the directory hierarchy from each file specified by *path*,  
 91495 evaluating a Boolean expression composed of the primaries described in the OPERANDS section  
 91496 for each file encountered. Each *path* operand shall be evaluated unaltered as it was provided,  
 91497 including all trailing <slash> characters; all pathnames for other files encountered in the  
 91498 hierarchy shall consist of the concatenation of the current *path* operand, a <slash> if the current  
 91499 *path* operand did not end in one, and the filename relative to the *path* operand. The relative  
 91500 portion shall contain no dot or dot-dot components, no trailing <slash> characters, and only  
 91501 single <slash> characters between pathname components.

91502 The *find* utility shall be able to descend to arbitrary depths in a file hierarchy and shall not fail  
 91503 due to path length limitations (unless a *path* operand specified by the application exceeds  
 91504 {PATH\_MAX} requirements).

91505 The *find* utility shall detect infinite loops; that is, entering a previously visited directory that is an  
 91506 ancestor of the last file encountered. When it detects an infinite loop, *find* shall write a  
 91507 diagnostic message to standard error and shall either recover its position in the hierarchy or  
 91508 terminate.

91509 If a file is removed from or added to the directory hierarchy being searched it is unspecified  
 91510 whether or not *find* includes that file in its search.

91511 **OPTIONS**91512 The *find* utility shall conform to XBD [Section 12.2](#) (on page 216).

91513 The following options shall be supported by the implementation:

91514 **-H** Cause the file information and file type evaluated for each symbolic link  
 91515 encountered as a *path* operand on the command line to be those of the file  
 91516 referenced by the link, and not the link itself. If the referenced file does not exist,  
 91517 the file information and type shall be for the link itself. File information and type  
 91518 for symbolic links encountered during the traversal of a file hierarchy shall be that  
 91519 of the link itself.

91520 **-L** Cause the file information and file type evaluated for each symbolic link  
 91521 encountered as a *path* operand on the command line or encountered during the  
 91522 traversal of a file hierarchy to be those of the file referenced by the link, and not the  
 91523 link itself. If the referenced file does not exist, the file information and type shall be  
 91524 for the link itself.

91525 Specifying more than one of the mutually-exclusive options **-H** and **-L** shall not be considered  
 91526 an error. The last option specified shall determine the behavior of the utility. If neither the **-H**  
 91527 nor the **-L** option is specified, then the file information and type for symbolic links encountered  
 91528 as a *path* operand on the command line or encountered during the traversal of a file hierarchy  
 91529 shall be that of the link itself.

91530 **OPERANDS**

91531 The following operands shall be supported:

91532 The first operand and subsequent operands up to but not including the first operand that starts  
 91533 with a '-', or is a '!' or a '(', shall be interpreted as *path* operands. If the first operand starts  
 91534 with a '-', or is a '!' or a '(', the behavior is unspecified. Each *path* operand is a pathname of

91535 a starting point in the file hierarchy.

91536 The first operand that starts with a `'-'`, or is a `'!'` or a `'('`, and all subsequent arguments shall  
 91537 be interpreted as an *expression* made up of the following primaries and operators. In the  
 91538 descriptions, wherever *n* is used as a primary argument, it shall be interpreted as a decimal  
 91539 integer optionally preceded by a <plus-sign> (`'+'`) or <hyphen-minus> (`'-'`), as follows:

91540 `+n` More than *n*.

91541 `n` Exactly *n*.

91542 `-n` Less than *n*.

91543 The following primaries shall be supported:

91544 `-name pattern`

91545 The primary shall evaluate as true if the basename of the current pathname  
 91546 matches *pattern* using the pattern matching notation described in [Section 2.13](#) (on  
 91547 page 2372). The additional rules in [Section 2.13.3](#) (on page 2373) do not apply as  
 91548 this is a matching operation, not an expansion.

91549 `-path pattern`

91550 The primary shall evaluate as true if the current pathname matches *pattern* using  
 91551 the pattern matching notation described in [Section 2.13](#) (on page 2372). The  
 91552 additional rules in [Section 2.13.3](#) (on page 2373) do not apply as this is a matching  
 91553 operation, not an expansion.

91554 `-nouser` The primary shall evaluate as true if the file belongs to a user ID for which the  
 91555 `getpwuid()` function defined in the System Interfaces volume of POSIX.1-2008 (or  
 91556 equivalent) returns NULL.

91557 `-nogroup` The primary shall evaluate as true if the file belongs to a group ID for which the  
 91558 `getgrgid()` function defined in the System Interfaces volume of POSIX.1-2008 (or  
 91559 equivalent) returns NULL.

91560 `-xdev` The primary shall always evaluate as true; it shall cause *find* not to continue  
 91561 descending past directories that have a different device ID (*st\_dev*, see the *stat()*  
 91562 function defined in the System Interfaces volume of POSIX.1-2008). If any `-xdev`  
 91563 primary is specified, it shall apply to the entire expression even if the `-xdev`  
 91564 primary would not normally be evaluated.

91565 `-prune` The primary shall always evaluate as true; it shall cause *find* not to descend the  
 91566 current pathname if it is a directory. If the `-depth` primary is specified, the `-prune`  
 91567 primary shall have no effect.

91568 `-perm [-]mode`

91569 The *mode* argument is used to represent file mode bits. It shall be identical in  
 91570 format to the *symbolic\_mode* operand described in *chmod*, and shall be interpreted  
 91571 as follows. To start, a template shall be assumed with all file mode bits cleared. An  
 91572 *op* symbol of `'+'` shall set the appropriate mode bits in the template; `'-'` shall  
 91573 clear the appropriate bits; `'='` shall set the appropriate mode bits, without regard  
 91574 to the contents of the file mode creation mask of the process. The *op* symbol of `'-'`  
 91575 cannot be the first character of *mode*; this avoids ambiguity with the optional  
 91576 leading <hyphen-minus>. Since the initial mode is all bits off, there are not any  
 91577 symbolic modes that need to use `'-'` as the first character.

91578 If the <hyphen-minus> is omitted, the primary shall evaluate as true when the file  
 91579 permission bits exactly match the value of the resulting template.



91580 Otherwise, if *mode* is prefixed by a <hyphen-minus>, the primary shall evaluate as  
 91581 true if at least all the bits in the resulting template are set in the file permission bits.

91582 **-perm** [-]*onum*  
 91583 If the <hyphen-minus> is omitted, the primary shall evaluate as true when the file  
 91584 mode bits exactly match the value of the octal number *onum* (see the description of  
 91585 the octal *mode* in *chmod*). Otherwise, if *onum* is prefixed by a <hyphen-minus>, the  
 91586 primary shall evaluate as true if at least all of the bits specified in *onum* are set. In  
 91587 both cases, the behavior is unspecified when *onum* exceeds 07777.

91588 **-type** *c* The primary shall evaluate as true if the type of the file is *c*, where *c* is 'b', 'c',  
 91589 'd', 'l', 'p', 'f', or 's' for block special file, character special file, directory,  
 91590 symbolic link, FIFO, regular file, or socket, respectively.

91591 **-links** *n* The primary shall evaluate as true if the file has *n* links.

91592 **-user** *uname* The primary shall evaluate as true if the file belongs to the user *uname*. If *uname* is  
 91593 a decimal integer and the *getpwnam()* (or equivalent) function does not return a  
 91594 valid user name, *uname* shall be interpreted as a user ID.

91595 **-group** *gname*  
 91596 The primary shall evaluate as true if the file belongs to the group *gname*. If *gname*  
 91597 is a decimal integer and the *getgrnam()* (or equivalent) function does not return a  
 91598 valid group name, *gname* shall be interpreted as a group ID.

91599 **-size** *n*[*c*] The primary shall evaluate as true if the file size in bytes, divided by 512 and  
 91600 rounded up to the next integer, is *n*. If *n* is followed by the character '*c*', the size  
 91601 shall be in bytes.

91602 **-atime** *n* The primary shall evaluate as true if the file access time subtracted from the  
 91603 initialization time, divided by 86 400 (with any remainder discarded), is *n*.

91604 **-ctime** *n* The primary shall evaluate as true if the time of last change of file status  
 91605 information subtracted from the initialization time, divided by 86 400 (with any  
 91606 remainder discarded), is *n*.

91607 **-mtime** *n* The primary shall evaluate as true if the file modification time subtracted from the  
 91608 initialization time, divided by 86 400 (with any remainder discarded), is *n*.

91609 **-exec** *utility\_name* [*argument* ...];  
 91610 **-exec** *utility\_name* [*argument* ...] {} +  
 91611 The end of the primary expression shall be punctuated by a <semicolon> or by a  
 91612 <plus-sign>. Only a <plus-sign> that immediately follows an argument  
 91613 containing only the two characters "{}" shall punctuate the end of the primary  
 91614 expression. Other uses of the <plus-sign> shall not be treated as special.

91615 If the primary expression is punctuated by a <semicolon>, the utility *utility\_name*  
 91616 shall be invoked once for each pathname and the primary shall evaluate as true if  
 91617 the utility returns a zero value as exit status. A *utility\_name* or *argument* containing  
 91618 only the two characters "{}" shall be replaced by the current pathname. If a  
 91619 *utility\_name* or *argument* string contains the two characters "{}", but not just the  
 91620 two characters "{}", it is implementation-defined whether *find* replaces those two  
 91621 characters or uses the string without change.

91622 If the primary expression is punctuated by a <plus-sign>, the primary shall always  
 91623 evaluate as true, and the pathnames for which the primary is evaluated shall be  
 91624 aggregated into sets. The utility *utility\_name* shall be invoked once for each set of  
 91625 aggregated pathnames. Each invocation shall begin after the last pathname in the

set is aggregated, and shall be completed before the *find* utility exits and before the first pathname in the next set (if any) is aggregated for this primary, but it is otherwise unspecified whether the invocation occurs before, during, or after the evaluations of other primaries. If any invocation returns a non-zero value as exit status, the *find* utility shall return a non-zero exit status. An argument containing only the two characters "{}" shall be replaced by the set of aggregated pathnames, with each pathname passed as a separate argument to the invoked utility in the same order that it was aggregated. The size of any set of two or more pathnames shall be limited such that execution of the utility does not cause the system's {ARG\_MAX} limit to be exceeded. If more than one argument containing the two characters "{}" is present, the behavior is unspecified.

The current directory for the invocation of *utility\_name* shall be the same as the current directory when the *find* utility was started. If the *utility\_name* names any of the special built-in utilities (see [Section 2.14](#), on page 2374), the results are undefined.

**-ok** *utility\_name* [*argument* ...];

The **-ok** primary shall be equivalent to **-exec**, except that the use of a <plus-sign> to punctuate the end of the primary expression need not be supported, and *find* shall request affirmation of the invocation of *utility\_name* using the current file as an argument by writing to standard error as described in the STDERR section. If the response on standard input is affirmative, the utility shall be invoked. Otherwise, the command shall not be invoked and the value of the **-ok** operand shall be false.

**-print** The primary shall always evaluate as true; it shall cause the current pathname to be written to standard output.

**-newer** *file* The primary shall evaluate as true if the modification time of the current file is more recent than the modification time of the file named by the pathname *file*.

**-depth** The primary shall always evaluate as true; it shall cause descent of the directory hierarchy to be done so that all entries in a directory are acted on before the directory itself. If a **-depth** primary is not specified, all entries in a directory shall be acted on after the directory itself. If any **-depth** primary is specified, it shall apply to the entire expression even if the **-depth** primary would not normally be evaluated.

The primaries can be combined using the following operators (in order of decreasing precedence):

( *expression* ) True if *expression* is true.

! *expression* Negation of a primary; the unary NOT operator.

*expression* [-a] *expression*

Conjunction of primaries; the AND operator is implied by the juxtaposition of two primaries or made explicit by the optional **-a** operator. The second expression shall not be evaluated if the first expression is false.

*expression* -o *expression*

Alternation of primaries; the OR operator. The second expression shall not be evaluated if the first expression is true.

If no *expression* is present, **-print** shall be used as the expression. Otherwise, if the given expression does not contain any of the primaries **-exec**, **-ok**, or **-print**, the given expression



91672 shall be effectively replaced by:  
 91673 ( *given\_expression* ) -print

91674 The **-user**, **-group**, and **-newer** primaries each shall evaluate their respective arguments only  
 91675 once.

91676 When the file type evaluated for the current file is a symbolic link, the results of evaluating the  
 91677 **-perm** primary are implementation-defined.

#### 91678 STDIN

91679 If the **-ok** primary is used, the response shall be read from the standard input. An entire line  
 91680 shall be read as the response. Otherwise, the standard input shall not be used.

#### 91681 INPUT FILES

91682 None.

#### 91683 ENVIRONMENT VARIABLES

91684 The following environment variables shall affect the execution of *find*:

91685 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 91686 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 91687 variables used to determine the values of locale categories.)

91688 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 91689 internationalization variables.

91690 **LC\_COLLATE**

91691 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 91692 character collating elements used in the pattern matching notation for the **-n**  
 91693 option and in the extended regular expression defined for the **yesexpr** locale  
 91694 keyword in the **LC\_MESSAGES** category.

91695 **LC\_CTYPE** This variable determines the locale for the interpretation of sequences of bytes of  
 91696 text data as characters (for example, single-byte as opposed to multi-byte  
 91697 characters in arguments), the behavior of character classes within the pattern  
 91698 matching notation used for the **-n** option, and the behavior of character classes  
 91699 within regular expressions used in the extended regular expression defined for the  
 91700 **yesexpr** locale keyword in the **LC\_MESSAGES** category.

91701 **LC\_MESSAGES**

91702 Determine the locale used to process affirmative responses, and the locale used to  
 91703 affect the format and contents of diagnostic messages and prompts written to  
 91704 standard error.

91705 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

91706 **PATH** Determine the location of the *utility\_name* for the **-exec** and **-ok** primaries, as  
 91707 described in XBD [Chapter 8](#) (on page 173).

#### 91708 ASYNCHRONOUS EVENTS

91709 Default.

#### 91710 STDOUT

91711 The **-print** primary shall cause the current pathnames to be written to standard output. The  
 91712 format shall be:

91713 "%s\n", <path>

91714 **STDERR**

91715 The **-ok** primary shall write a prompt to standard error containing at least the *utility\_name* to be  
 91716 invoked and the current pathname. In the POSIX locale, the last non-`<blank>` in the prompt shall  
 91717 be `' ? '`. The exact format used is unspecified.

91718 Otherwise, the standard error shall be used only for diagnostic messages.

91719 **OUTPUT FILES**

91720 None.

91721 **EXTENDED DESCRIPTION**

91722 None.

91723 **EXIT STATUS**

91724 The following exit values shall be returned:

91725 0 All *path* operands were traversed successfully.

91726 >0 An error occurred.

91727 **CONSEQUENCES OF ERRORS**

91728 Default.

91729 **APPLICATION USAGE**

91730 When used in operands, pattern matching notation, `<semicolon>`, `<left-parenthesis>`, and  
 91731 `<right-parenthesis>` characters are special to the shell and must be quoted (see [Section 2.2](#), on  
 91732 page 2336).

91733 The bit that is traditionally used for sticky (historically 01000) is specified in the **-perm** primary  
 91734 using the octal number argument form. Since this bit is not defined by this volume of  
 91735 POSIX.1-2008, applications must not assume that it actually refers to the traditional sticky bit.

91736 **EXAMPLES**

91737 1. The following commands are equivalent:

91738 `find .`  
 91739 `find . -print`

91740 They both write out the entire directory hierarchy from the current directory.

91741 2. The following command:

91742 `find / \( -name tmp -o -name '*.xx' \) -atime +7 -exec rm {} \;`

91743 removes all files named **tmp** or ending in **.xx** that have not been accessed for seven or  
 91744 more 24-hour periods.

91745 3. The following command:

91746 `find . -perm -o+w,+s`

91747 prints (**-print** is assumed) the names of all files in or below the current directory, with all  
 91748 of the file permission bits **S\_ISUID**, **S\_ISGID**, and **S\_IWOTH** set.

91749 4. The following command:

91750 `find . -name SCCS -prune -o -print`

91751 recursively prints pathnames of all files in the current directory and below, but skips  
 91752 directories named **SCCS** and files in them.

5. The following command:

```
find . -print -name SCCS -prune
```

behaves as in the previous example, but prints the names of the SCCS directories.

6. The following command is roughly equivalent to the **-nt** extension to *test*:

```
if [ -n "$(find file1 -prune -newer file2)" ]; then
    printf %s\\n "file1 is newer than file2"
fi
```

7. The descriptions of **-atime**, **-ctime**, and **-mtime** use the terminology *n* “86 400 second periods (days)”. For example, a file accessed at 23:59 is selected by:

```
find . -atime -1 -print
```

at 00:01 the next day (less than 24 hours later, not more than one day ago); the midnight boundary between days has no effect on the 24-hour calculation.

8. The following command:

```
find . ! -name . -prune -name '*.old' -exec \
    sh -c 'mv "$@" ../old/' sh {} +
```

performs the same task as:

```
mv /*.old ../old /*.old ../old/
```

while avoiding an “Argument list too long” error if there are a large number of files ending with **.old** and without running *mv* if there are no such files (and avoiding “No such file or directory” errors if **./old** does not exist or no files match **/\*.old** or **./.\*.old**).

The alternative:

```
find . ! -name . -prune -name '*.old' -exec mv {} ../old/ \;
```

is less efficient if there are many files to move because it executes one *mv* command per file.

9. On systems configured to mount removable media on directories under **/media**, the following command searches the file hierarchy for files larger than 100 000 KB without searching any mounted removable media:

```
find / -path /media -prune -o -size +200000 -print
```

10. Except for the root directory, and **"/"** on implementations where **"/"** does not refer to the root directory, no pattern given to **-name** will match a **<slash>**, because trailing **<slash>** characters are ignored when computing the basename of the file under evaluation. Given two empty directories named **foo** and **bar**, the following command:

```
find foo/// bar/// -name foo -o -name 'bar?*'
```

prints only the line **foo///**.

## RATIONALE

The **-a** operator was retained as an optional operator for compatibility with historical shell scripts, even though it is redundant with expression concatenation.

The descriptions of the **'-'** modifier on the *mode* and *onum* arguments to the **-perm** primary agree with historical practice on BSD and System V implementations. System V and BSD documentation both describe it in terms of checking additional bits; in fact, it uses the same bits, but checks for having at least all of the matching bits set instead of having exactly the matching

bits set.

The exact format of the interactive prompts is unspecified. Only the general nature of the contents of prompts are specified because:

- Implementations may desire more descriptive prompts than those used on historical implementations.
- Since the historical prompt strings do not terminate with <newline> characters, there is no portable way for another program to interact with the prompts of this utility via pipes.

Therefore, an application using this prompting option relies on the system to provide the most suitable dialog directly with the user, based on the general guidelines specified.

The **-name** *file* operand was changed to use the shell pattern matching notation so that *find* is consistent with other utilities using pattern matching.

The **-size** operand refers to the size of a file, rather than the number of blocks it may occupy in the file system. The intent is that the *st\_size* field defined in the System Interfaces volume of POSIX.1-2008 should be used, not the *st\_blocks* found in historical implementations. There are at least two reasons for this:

1. In both System V and BSD, *find* only uses *st\_size* in size calculations for the operands specified by this volume of POSIX.1-2008. (BSD uses *st\_blocks* only when processing the **-ls** primary.)
2. Users usually think of file size in terms of bytes, which is also the unit used by the *ls* utility for the output from the **-l** option. (In both System V and BSD, *ls* uses *st\_size* for the **-l** option size field and uses *st\_blocks* for the *ls -s* calculations. This volume of POSIX.1-2008 does not specify *ls -s*.)

The descriptions of **-atime**, **-ctime**, and **-mtime** were changed from the SVID description of *n* "days" to *n* being the result of the integer division of the time difference in seconds by 86 400. The description is also different in terms of the exact timeframe for the *n* case (*versus* the *+n* or *-n*), but it matches all known historical implementations. It refers to one 86 400 second period in the past, not any time from the beginning of that period to the current time. For example, **-atime 2** is true if the file was accessed any time in the period from 72 hours to 48 hours ago.

Historical implementations do not modify "{ }" when it appears as a substring of an **-exec** or **-ok** *utility\_name* or argument string. There have been numerous user requests for this extension, so this volume of POSIX.1-2008 allows the desired behavior. At least one recent implementation does support this feature, but encountered several problems in managing memory allocation and dealing with multiple occurrences of "{ }" in a string while it was being developed, so it is not yet required behavior.

Assuming the presence of **-print** was added to correct a historical pitfall that plagues novice users, it is entirely upwards-compatible from the historical System V *find* utility. In its simplest form (*find directory*), it could be confused with the historical BSD fast *find*. The BSD developers agreed that adding **-print** as a default expression was the correct decision and have added the fast *find* functionality within a new utility called *locate*.

Historically, the **-L** option was implemented using the primary **-follow**. The **-H** and **-L** options were added for two reasons. First, they offer a finer granularity of control and consistency with other programs that walk file hierarchies. Second, the **-follow** primary always evaluated to true. As they were historically really global variables that took effect before the traversal began, some valid expressions had unexpected results. An example is the expression **-print -o -follow**. Because **-print** always evaluates to true, the standard order of evaluation implies that **-follow** would never be evaluated. This was never the case. Historical practice for the **-follow** primary,

however, is not consistent. Some implementations always follow symbolic links on the command line whether **-follow** is specified or not. Others follow symbolic links on the command line only if **-follow** is specified. Both behaviors are provided by the **-H** and **-L** options, but scripts using the current **-follow** primary would be broken if the **-follow** option is specified to work either way.

Since the `-L` option resolves all symbolic links and the `-type l` primary is true for symbolic links that still exist after symbolic links have been resolved, the command:

```
find -L . -type l
```

prints a list of symbolic links reachable from the current directory that do not resolve to accessible files.

A feature of SVR4's *find* utility was the **-exec** primary's + terminator. This allowed filenames containing special characters (especially <newline> characters) to be grouped together without the problems that occur if such filenames are piped to *xargs*. Other implementations have added other ways to get around this problem, notably a **-print0** primary that wrote filenames with a null byte terminator. This was considered here, but not adopted. Using a null terminator meant that any utility that was going to process *find*'s **-print0** output had to add a new option to parse the null terminators it would now be reading.

The "-exec ... {} +" syntax adopted was a result of IEEE PASC Interpretation 1003.2 #210. It should be noted that this is an incompatible change to IEEE Std 1003.2-1992. For example, the following command printed all files with a '-' after their name if they are regular files, and a '+' otherwise:

```
find / -type f -exec echo {} - ';' -o -exec echo {} + ';' 2>/dev/null
```

The change invalidates usage like this. Even though the previous standard stated that this usage would work, in practice many did not support it and the standard developers felt it better to now state that this was not allowable.

## FUTURE DIRECTIONS

None.

**SEE ALSO**

Section 2.2 (on page 2336), Section 2.13 (on page 2372), Section 2.14 (on page 2374), *chmod*, *mv*, *pax*, *sh*, *test*

XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

XSH *fstatat()*, *getgrgid()*, *getpwuid()*

## CHANGE HISTORY

First released in Issue 2.

## Issue 5

The FUTURE DIRECTIONS section is added.

## Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **-perm** [-]onum primary is supported.

The *find* utility is aligned with the IEEE P1003.2b draft standard, to include processing of symbolic links and changes to the description of the **atime**, **ctime**, and **mtime** operands.

IEEE PASC Interpretation 1003.2 #210 is applied, extending the `-exec` operand.

91883 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/13 is applied, updating the RATIONALE  
91884 section to be consistent with the normative text.

91885 **Issue 7**

91886 Austin Group Interpretation 1003.1-2001 #126 is applied, changing the description of the  
91887 *LC\_MESSAGES* environment variable.

91888 Austin Group Interpretation 1003.1-2001 #127 is applied, rephrasing the description of the **-exec**  
91889 primary to be “immediately follows”.

91890 Austin Group Interpretation 1003.1-2001 #185 is applied, clarifying the requirements for the **-H**  
91891 and **-L** options.

91892 Austin Group Interpretation 1003.1-2001 #186 is applied, clarifying the requirements for the  
91893 evaluation of *path* operands.

91894 Austin Group Interpretation 1003.1-2001 #195 is applied, clarifying the interpretation of the first  
91895 operand.

91896 SD5-XCU-ERN-48 is applied, clarifying the **-L** option in the case that the referenced file does not  
91897 exist.

91898 SD5-XCU-ERN-89 is applied, updating the OPERANDS section.

91899 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

91900 SD5-XCU-ERN-117 is applied, clarifying the **-perm** operand.

91901 SD5-XCU-ERN-122 is applied, adding a new EXAMPLE.

91902 The description of the **-name** primary is revised and the **-path** primary is added (with a new  
91903 example).

91904 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0086 [365], XCU/TC1-2008/0087  
91905 [310], XCU/TC1-2008/0088 [309,310,430], XCU/TC1-2008/0089 [235], and XCU/TC1-2008/0090  
91906 [445] are applied.

91907 **NAME**

91908 fold — filter for folding lines

91909 **SYNOPSIS**91910 fold [-bs] [-w *width*] [*file...*]91911 **DESCRIPTION**

91912 The *fold* utility is a filter that shall fold lines from its input files, breaking the lines to have a  
 91913 maximum of *width* column positions (or bytes, if the **-b** option is specified). Lines shall be  
 91914 broken by the insertion of a <newline> such that each output line (referred to later in this section  
 91915 as a *segment*) is the maximum width possible that does not exceed the specified number of  
 91916 column positions (or bytes). A line shall not be broken in the middle of a character. The behavior  
 91917 is undefined if *width* is less than the number of columns any single character in the input would  
 91918 occupy.

91919 If the <carriage-return>, <backspace>, or <tab> characters are encountered in the input, and the  
 91920 **-b** option is not specified, they shall be treated specially:

91921 <backspace> The current count of line width shall be decremented by one, although the count  
 91922 never shall become negative. The *fold* utility shall not insert a <newline>  
 91923 immediately before or after any <backspace>, unless the following character has a  
 91924 width greater than 1 and would cause the line width to exceed *width*.

91925 <carriage-return>  
 91926 The current count of line width shall be set to zero. The *fold* utility shall not insert a  
 91927 <newline> immediately before or after any <carriage-return>.

91928 <tab> Each <tab> encountered shall advance the column position pointer to the next tab  
 91929 stop. Tab stops shall be at each column position *n* such that *n* modulo 8 equals 1.

91930 **OPTIONS**91931 The *fold* utility shall conform to XBD [Section 12.2](#) (on page 216).

91932 The following options shall be supported:

91933 **-b** Count *width* in bytes rather than column positions.

91934 **-s** If a segment of a line contains a <blank> within the first *width* column positions (or  
 91935 bytes), break the line after the last such <blank> meeting the width constraints. If  
 91936 there is no <blank> meeting the requirements, the **-s** option shall have no effect for  
 91937 that output segment of the input line.

91938 **-w *width*** Specify the maximum line length, in column positions (or bytes if **-b** is specified).  
 91939 The results are unspecified if *width* is not a positive decimal number. The default  
 91940 value shall be 80.

91941 **OPERANDS**

91942 The following operand shall be supported:

91943 *file* A pathname of a text file to be folded. If no *file* operands are specified, the standard  
 91944 input shall be used.

91945 **STDIN**

91946 The standard input shall be used if no *file* operands are specified, and shall be used if a *file*  
 91947 operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
 91948 the standard input shall not be used. See the INPUT FILES section.



91949 **INPUT FILES**

91950 If the **-b** option is specified, the input files shall be text files except that the lines are not limited  
 91951 to {LINE\_MAX} bytes in length. If the **-b** option is not specified, the input files shall be text files.

91952 **ENVIRONMENT VARIABLES**

91953 The following environment variables shall affect the execution of *fold*:

91954 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 91955 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 91956 variables used to determine the values of locale categories.)

91957 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 91958 internationalization variables.

91959 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 91960 characters (for example, single-byte as opposed to multi-byte characters in  
 91961 arguments and input files), and for the determination of the width in column  
 91962 positions each character would occupy on a constant-width font output device.

91963 **LC\_MESSAGES**  
 91964 Determine the locale that should be used to affect the format and contents of  
 91965 diagnostic messages written to standard error.

91966 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

91967 **ASYNCHRONOUS EVENTS**

91968 Default.

91969 **STDOUT**

91970 The standard output shall be a file containing a sequence of characters whose order shall be  
 91971 preserved from the input files, possibly with inserted <newline> characters.

91972 **STDERR**

91973 The standard error shall be used only for diagnostic messages.

91974 **OUTPUT FILES**

91975 None.

91976 **EXTENDED DESCRIPTION**

91977 None.

91978 **EXIT STATUS**

91979 The following exit values shall be returned:

91980 0 All input files were processed successfully.

91981 >0 An error occurred.

91982 **CONSEQUENCES OF ERRORS**

91983 Default.



**APPLICATION USAGE**

The *cut* and *fold* utilities can be used to create text files out of files with arbitrary line lengths. The *cut* utility should be used when the number of lines (or records) needs to remain constant. The *fold* utility should be used when the contents of long lines need to be kept contiguous.

The *fold* utility is frequently used to send text files to printers that truncate, rather than fold, lines wider than the printer is able to print (usually 80 or 132 column positions).

**EXAMPLES**

An example invocation that submits a file of possibly long lines to the printer (under the assumption that the user knows the line width of the printer to be assigned by *lp*):

```
fold -w 132 bigfile | lp
```

**RATIONALE**

Although terminal input in canonical processing mode requires the erase character (frequently set to <backspace>) to erase the previous character (not byte or column position), terminal output is not buffered and is extremely difficult, if not impossible, to parse correctly; the interpretation depends entirely on the physical device that actually displays/prints/stores the output. In all known internationalized implementations, the utilities producing output for mixed column-width output assume that a <backspace> character backs up one column position and outputs enough <backspace> characters to return to the start of the character when <backspace> is used to provide local line motions to support underlining and emboldening operations. Since *fold* without the **-b** option is dealing with these same constraints, <backspace> is always treated as backing up one column position rather than backing up one character.

Historical versions of the *fold* utility assumed 1 byte was one character and occupied one column position when written out. This is no longer always true. Since the most common usage of *fold* is believed to be folding long lines for output to limited-length output devices, this capability was preserved as the default case. The **-b** option was added so that applications could *fold* files with arbitrary length lines into text files that could then be processed by the standard utilities. Note that although the width for the **-b** option is in bytes, a line is never split in the middle of a character. (It is unspecified what happens if a width is specified that is too small to hold a single character found in the input followed by a <newline>.)

The tab stops are hardcoded to be every eighth column to meet historical practice. No new method of specifying other tab stops was invented.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*cut*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

**CHANGE HISTORY**

First released in Issue 4.

**Issue 6**

The normative text is reworded to avoid use of the term “must” for application requirements.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #092 is applied.

92026 Austin Group Interpretation 1003.1-2001 #204 is applied, updating the DESCRIPTION to clarify  
92027 when a <newline> can be inserted before or after a <backspace>.  
92028 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

92029 **NAME**92030 fort77 — FORTRAN compiler (**FORTRAN**)92031 **SYNOPSIS**

```
92032 OB FD fort77 [-c] [-g] [-L directory]... [-O optlevel] [-o outfile] [-s]
92033          [-w] operand...
```

92034 **DESCRIPTION**

92035 The *fort77* utility is the interface to the FORTRAN compilation system; it shall accept the full  
 92036 FORTRAN-77 language defined by the ANSI X3.9-1978 standard. The system conceptually  
 92037 consists of a compiler and link editor. The files referenced by *operands* are compiled and linked  
 92038 to produce an executable file. It is unspecified whether the linking occurs entirely within the  
 92039 operation of *fort77*; some implementations may produce objects that are not fully resolved until  
 92040 the file is executed.

92041 If the *-c* option is present, for all pathname operands of the form *file.f*, the files:

92042  $\$(\text{basename } \textit{pathname.f}) .o$

92043 shall be created or overwritten as the result of successful compilation. If the *-c* option is not  
 92044 specified, it is unspecified whether such *.o* files are created or deleted for the *file.f* operands.

92045 If there are no options that prevent link editing (such as *-c*) and all operands compile and link  
 92046 without error, the resulting executable file shall be written into the file named by the *-o* option  
 92047 (if present) or to the file **a.out**. The executable file shall be created as specified in the System  
 92048 Interfaces volume of POSIX.1-2008, except that the file permissions shall be set to:

92049  $S\_IRWXO \mid S\_IRWXG \mid S\_IRWXU$

92050 and that the bits specified by the *umask* of the process shall be cleared.

92051 **OPTIONS**

92052 The *fort77* utility shall conform to XBD [Section 12.2](#) (on page 216), except that:

- 92053 • The *-l library* operands have the format of options, but their position within a list of  
 92054 operands affects the order in which libraries are searched.
- 92055 • The order of specifying the multiple *-L* options is significant.
- 92056 • Conforming applications shall specify each option separately; that is, grouping option  
 92057 letters (for example, *-cg*) need not be recognized by all implementations.

92058 The following options shall be supported:

- 92059 *-c* Suppress the link-edit phase of the compilation, and do not remove any object files  
 92060 that are produced.
- 92061 *-g* Produce symbolic information in the object or executable files; the nature of this  
 92062 information is unspecified, and may be modified by implementation-defined  
 92063 interactions with other options.
- 92064 *-s* Produce object or executable files, or both, from which symbolic and other  
 92065 information not required for proper execution using the *exec* family of functions  
 92066 defined in the System Interfaces volume of POSIX.1-2008 has been removed  
 92067 (stripped). If both *-g* and *-s* options are present, the action taken is unspecified.
- 92068 *-o outfile* Use the pathname *outfile*, instead of the default **a.out**, for the executable file  
 92069 produced. If the *-o* option is present with *-c*, the result is unspecified.

92070        **-L *directory***    Change the algorithm of searching for the libraries named in **-I** operands to look in  
 92071                            the *directory* named by the *directory* pathname before looking in the usual places.  
 92072                            Directories named in **-L** options shall be searched in the specified order. At least  
 92073                            ten instances of this option shall be supported in a single *fort77* command  
 92074                            invocation. If a directory specified by a **-L** option contains a file named **libf.a**, the  
 92075                            results are unspecified.

92076        **-O *optlevel***    Specify the level of code optimization. If the *optlevel* option-argument is the digit  
 92077                            '0', all special code optimizations shall be disabled. If it is the digit '1', the  
 92078                            nature of the optimization is unspecified. If the **-O** option is omitted, the nature of  
 92079                            the system's default optimization is unspecified. It is unspecified whether code  
 92080                            generated in the presence of the **-O 0** option is the same as that generated when  
 92081                            **-O** is omitted. Other *optlevel* values may be supported.

92082        **-w**                Suppress warnings.

92083        Multiple instances of **-L** options can be specified.

## 92084 OPERANDS

92085        An *operand* is either in the form of a pathname or the form **-I *library***. At least one operand of the  
 92086        pathname form shall be specified. The following operands shall be supported:

92087        *file.f*              The pathname of a FORTRAN source file to be compiled and optionally passed to  
 92088                            the link editor. The filename operand shall be of this form if the **-c** option is used.

92089        *file.a*              A library of object files typically produced by *ar*, and passed directly to the link  
 92090                            editor. Implementations may recognize implementation-defined suffixes other  
 92091                            than **.a** as denoting object file libraries.

92092        *file.o*              An object file produced by *fort77 -c* and passed directly to the link editor.  
 92093                            Implementations may recognize implementation-defined suffixes other than **.o** as  
 92094                            denoting object files.

92095        The processing of other files is implementation-defined.

92096        **-I *library***        (The letter ell.) Search the library named:

92097                            *liblibrary.a*

92098                            A library is searched when its name is encountered, so the placement of a **-I**  
 92099                            operand is significant. Several standard libraries can be specified in this manner, as  
 92100                            described in the EXTENDED DESCRIPTION section. Implementations may  
 92101                            recognize implementation-defined suffixes other than **.a** as denoting libraries.

## 92102 STDIN

92103        Not used.

## 92104 INPUT FILES

92105        The input file shall be one of the following: a text file containing FORTRAN source code; an  
 92106        object file in the format produced by *fort77 -c*; or a library of object files, in the format produced  
 92107        by archiving zero or more object files, using *ar*. Implementations may supply additional utilities  
 92108        that produce files in these formats. Additional input files are implementation-defined.

92109        A <tab> encountered within the first six characters on a line of source code shall cause the  
 92110        compiler to interpret the following character as if it were the seventh character on the line (that  
 92111        is, in column 7).

92112 **ENVIRONMENT VARIABLES**92113 The following environment variables shall affect the execution of *fort77*:

92114 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 92115 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 92116 variables used to determine the values of locale categories.)

92117 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 92118 internationalization variables.

92119 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 92120 characters (for example, single-byte as opposed to multi-byte characters in  
 92121 arguments and input files).

92122 **LC\_MESSAGES**

92123 Determine the locale that should be used to affect the format and contents of  
 92124 diagnostic messages written to standard error.

92125 **XSI NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

92126 **TMPDIR** Determine the pathname that should override the default directory for temporary  
 92127 files, if any.

92128 **ASYNCHRONOUS EVENTS**

92129 Default.

92130 **STDOUT**

92131 Not used.

92132 **STDERR**

92133 The standard error shall be used only for diagnostic messages. If more than one *file* operand  
 92134 ending in *.f* (or possibly other unspecified suffixes) is given, for each such file:

92135 "%s:\n", *<file>*

92136 may be written to allow identification of the diagnostic message with the appropriate input file.

92137 This utility may produce warning messages about certain conditions that do not warrant  
 92138 returning an error (non-zero) exit value.

92139 **OUTPUT FILES**

92140 Object files, listing files, and executable files shall be produced in unspecified formats.

92141 **EXTENDED DESCRIPTION**92142 **Standard Libraries**92143 The *fort77* utility shall recognize the following **-l** operand for the standard library:

92144 **-l f** This library contains all functions referenced in the ANSI X3.9-1978 standard. This  
 92145 operand shall not be required to be present to cause a search of this library.

92146 In the absence of options that inhibit invocation of the link editor, such as **-c**, the *fort77* utility  
 92147 shall cause the equivalent of a **-l f** operand to be passed to the link editor as the last **-l** operand,  
 92148 causing it to be searched after all other object files and libraries are loaded.

92149 It is unspecified whether the library **libf.a** exists as a regular file. The implementation may  
 92150 accept as **-l** operands names of objects that do not exist as regular files.

## External Symbols

The FORTRAN compiler and link editor shall support the significance of external symbols up to a length of at least 31 bytes; case folding is permitted. The action taken upon encountering symbols exceeding the implementation-defined maximum symbol length is unspecified.

The compiler and link editor shall support a minimum of 511 external symbols per source or object file, and a minimum of 4095 external symbols total. A diagnostic message is written to standard output if the implementation-defined limit is exceeded; other actions are unspecified.

## EXIT STATUS

The following exit values shall be returned:

0 Successful compilation or link edit.

>0 An error occurred.

## CONSEQUENCES OF ERRORS

When *fort77* encounters a compilation error, it shall write a diagnostic to standard error and continue to compile other source code operands. It shall return a non-zero exit status, but it is implementation-defined whether an object module is created. If the link edit is unsuccessful, a diagnostic message shall be written to standard error, and *fort77* shall exit with a non-zero status.

## APPLICATION USAGE

None.

## EXAMPLES

The following usage example compiles **xyz.f** and creates the executable file **foo**:

```
fort77 -o foo xyz.f
```

The following example compiles **xyz.f** and creates the object file **xyz.o**:

```
fort77 -c xyz.f
```

The following example compiles **xyz.f** and creates the executable file **a.out**:

```
fort77 xyz.f
```

The following example compiles **xyz.f**, links it with **b.o**, and creates the executable **a.out**:

```
fort77 xyz.f b.o
```

## RATIONALE

The name of this utility was chosen as *fort77* to parallel the renaming of the C compiler. The name *f77* was not chosen to avoid problems with historical implementations. The ANSI X3.9-1978 standard was selected as a normative reference because the ISO/IEC version of FORTRAN-77 has been superseded by the ISO/IEC 1539:1991 standard.

The file inclusion and symbol definition **#define** mechanisms used by the *c99* utility were not included in this volume of POSIX.1-2008—even though they are commonly implemented—since there is no requirement that the FORTRAN compiler use the C preprocessor.

The **-onetrip** option was not included in this volume of POSIX.1-2008, even though many historical compilers support it, because it is derived from FORTRAN-66; it is an anachronism that should not be perpetuated.

Some implementations produce compilation listings. This aspect of FORTRAN has been left unspecified because there was controversy concerning the various methods proposed for implementing it: a **-V** option overlapped with historical vendor practice and a naming convention of creating files with **.l** suffixes collided with historical *lex* file naming practice.

There is no **-I** option in this version of this volume of POSIX.1-2008 to specify a directory for file inclusion. An **INCLUDE** directive has been a part of the Fortran-90 discussions, but an interface supporting that standard is not in the current scope.

It is noted that many FORTRAN compilers produce an object module even when compilation errors occur; during a subsequent compilation, the compiler may patch the object module rather than recompiling all the code. Consequently, it is left to the implementor whether or not an object file is created.

A reference to MIL-STD-1753 was removed from an early proposal in response to a request from the POSIX FORTRAN-binding standard developers. It was not the intention of the standard developers to require certification of the FORTRAN compiler, and IEEE Std 1003.9-1992 does not specify the military standard or any special preprocessing requirements. Furthermore, use of that document would have been inappropriate for an international standard.

The specification of optimization has been subject to changes through early proposals. At one time, **-O** and **-N** were Booleans: optimize and do not optimize (with an unspecified default). Some historical practice led this to be changed to:

**-O 0**        No optimization.

**-O 1**        Some level of optimization.

**-O n**        Other, unspecified levels of optimization.

It is not always clear whether “good code generation” is the same thing as optimization. Simple optimizations of local actions do not usually affect the semantics of a program. The **-O 0** option has been included to accommodate the very particular nature of scientific calculations in a highly optimized environment; compilers make errors. Some degree of optimization is expected, even if it is not documented here, and the ability to shut it off completely could be important when porting an application. An implementation may treat **-O 0** as “do less than normal” if it wishes, but this is only meaningful if any of the operations it performs can affect the semantics of a program. It is highly dependent on the implementation whether doing less than normal is logical. It is not the intent of the **-O 0** option to ask for inefficient code generation, but rather to assure that any semantically visible optimization is suppressed.

The specification of standard library access is consistent with the C compiler specification. Implementations are not required to have **/usr/lib/libf.a**, as many historical implementations do, but if not they are required to recognize **f** as a token.

External symbol size limits are in normative text; conforming applications need to know these limits. However, the minimum maximum symbol length should be taken as a constraint on a conforming application, not on an implementation, and consequently the action taken for a symbol exceeding the limit is unspecified. The minimum size for the external symbol table was added for similar reasons.

The CONSEQUENCES OF ERRORS section clearly specifies the behavior of the compiler when compilation or link-edit errors occur. The behavior of several historical implementations was examined, and the choice was made to be silent on the status of the executable, or **a.out**, file in the face of compiler or linker errors. If a linker writes the executable file, then links it on disk with *lseek()*s and *write()*s, the partially linked executable file can be left on disk and its execute bits turned off if the link edit fails. However, if the linker links the image in memory before writing the file to disk, it need not touch the executable file (if it already exists) because the link edit fails. Since both approaches are historical practice, a conforming application shall rely on the exit status of *fort77*, rather than on the existence or mode of the executable file.

The **-g** and **-s** options are not specified as mutually-exclusive. Historically, these two options



92239 have been mutually-exclusive, but because both are so loosely specified, it seemed appropriate  
92240 to leave their interaction unspecified.

92241 The requirement that conforming applications specify compiler options separately is to reserve  
92242 the multi-character option name space for vendor-specific compiler options, which are known to  
92243 exist in many historical implementations. Implementations are not required to recognize, for  
92244 example, `-gc` as if it were `-g -c`; nor are they forbidden from doing so. The SYNOPSIS shows all  
92245 of the options separately to highlight this requirement on applications.

92246 Echoing filenames to standard error is considered a diagnostic message because it would  
92247 otherwise be difficult to associate an error message with the erring file. They are described with  
92248 “may” to allow implementations to use other methods of identifying files and to parallel the  
92249 description in *c99*.

#### 92250 FUTURE DIRECTIONS

92251 Future versions of this standard may withdraw this utility. There are implementations of |  
92252 compilers that conform to much more recent versions of the FORTRAN programming language. |  
92253 Since there is no active FORTRAN binding to POSIX.1-2008, this standard does not need to |  
92254 specify any compiler.

#### 92255 SEE ALSO

92256 *ar, asa, c99, umask*

92257 XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

92258 XSH *exec*

#### 92259 CHANGE HISTORY

92260 First released in Issue 4.

#### 92261 Issue 6

92262 This utility is marked as part of the FORTRAN Development Utilities option.

92263 The normative text is reworded to avoid use of the term “must” for application requirements.

#### 92264 Issue 7

92265 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



92266 **NAME**

92267 fuser — list process IDs of all processes that have one or more files open

92268 **SYNOPSIS**92269 XSI `fuser [-cfu] file...`92270 **DESCRIPTION**

92271 The *fuser* utility shall write to standard output the process IDs of processes running on the local  
 92272 system that have one or more named files open. For block special devices, all processes using  
 92273 any file on that device are listed.

92274 The *fuser* utility shall write to standard error additional information about the named files  
 92275 indicating how the file is being used.

92276 Any output for processes running on remote systems that have a named file open is unspecified.

92277 A user may need appropriate privileges to invoke the *fuser* utility.

92278 **OPTIONS**

92279 The *fuser* utility shall conform to XBD [Section 12.2](#) (on page 216).

92280 The following options shall be supported:

92281 **-c** The file is treated as a mount point and the utility shall report on any files open in  
 92282 the file system.

92283 **-f** The report shall be only for the named files.

92284 **-u** The user name, in parentheses, associated with each process ID written to standard  
 92285 output shall be written to standard error.

92286 **OPERANDS**

92287 The following operand shall be supported:

92288 *file* A pathname on which the file or file system is to be reported.

92289 **STDIN**

92290 Not used.

92291 **INPUT FILES**

92292 The user database.

92293 **ENVIRONMENT VARIABLES**

92294 The following environment variables shall affect the execution of *fuser*:

92295 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 92296 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 92297 variables used to determine the values of locale categories.)

92298 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 92299 internationalization variables.

92300 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 92301 characters (for example, single-byte as opposed to multi-byte characters in  
 92302 arguments).

92303 **LC\_MESSAGES**

92304 Determine the locale that should be used to affect the format and contents of  
 92305 diagnostic messages written to standard error.

92306 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

92307 **ASYNCHRONOUS EVENTS**

92308 Default.

92309 **STDOUT**

92310 The *fuser* utility shall write the process ID for each process using each file given as an operand to

92311 standard output in the following format:

92312 "%d", <*process\_id*>

92313 **STDERR**

92314 The *fuser* utility shall write diagnostic messages to standard error.

92315 The *fuser* utility also shall write the following to standard error:

92316

- The pathname of each named file is written followed immediately by a <colon>.
- For each process ID written to standard output, the character 'c' shall be written to standard error if the process is using the file as its current directory and the character 'r' shall be written to standard error if the process is using the file as its root directory. Implementations may write other alphabetic characters to indicate other uses of files.
- When the *-u* option is specified, characters indicating the use of the file shall be followed immediately by the user name, in parentheses, corresponding to the real user ID of the process. If the user name cannot be resolved from the real user ID of the process, the real user ID of the process shall be written instead of the user name.

92325 When standard output and standard error are directed to the same file, the output shall be

92326 interleaved so that the filename appears at the start of each line, followed by the process ID and

92327 characters indicating the use of the file. Then, if the *-u* option is specified, the user name or user

92328 ID for each process using that file shall be written.

92329 A <newline> shall be written to standard error after the last output described above for each *file*

92330 operand.

92331 **OUTPUT FILES**

92332 None.

92333 **EXTENDED DESCRIPTION**

92334 None.

92335 **EXIT STATUS**

92336 The following exit values shall be returned:

92337 0 Successful completion.

92338 >0 An error occurred.

92339 **CONSEQUENCES OF ERRORS**

92340 Default.

**92341 APPLICATION USAGE**

92342 None.

**92343 EXAMPLES**

92344 The command:

92345 `fuser -fu .`

92346 writes to standard output the process IDs of processes that are using the current directory and  
92347 writes to standard error an indication of how those processes are using the directory and the  
92348 user names associated with the processes that are using the current directory.

92349 `fuser -c <mount point>`

92350 writes to standard output the process IDs of processes that are using any file in the file system  
92351 which is mounted on <mount point> and writes to standard error an indication of how those  
92352 processes are using the files.

92353 `fuser <mount point>`

92354 writes to standard output the process IDs of processes that are using the file which is named by  
92355 <mount point> and writes to standard error an indication of how those processes are using the  
92356 file.

92357 `fuser <block device>`

92358 writes to standard output the process IDs of processes that are using any file which is on the  
92359 device named by <block device> and writes to standard error an indication of how those  
92360 processes are using the file.

92361 `fuser -f <block device>`

92362 writes to standard output the process IDs of processes that are using the file <block device> itself  
92363 and writes to standard error an indication of how those processes are using the file.

**92364 RATIONALE**

92365 The definition of the *fuser* utility follows existing practice.

**92366 FUTURE DIRECTIONS**

92367 None.

**92368 SEE ALSO**

92369 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

**92370 CHANGE HISTORY**

92371 First released in Issue 5.

**92372 Issue 7**

92373 SD5-XCU-ERN-90 is applied, updating the EXAMPLES section.

92374 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

92375 **NAME**

92376 gencat — generate a formatted message catalog

92377 **SYNOPSIS**92378 gencat *catfile* *msgfile*...92379 **DESCRIPTION**

92380 The *gencat* utility shall merge the message text source file *msgfile* into a formatted message  
 92381 catalog *catfile*. The file *catfile* shall be created if it does not already exist. If *catfile* does exist, its  
 92382 messages shall be included in the new *catfile*. If set and message numbers collide, the new  
 92383 message text defined in *msgfile* shall replace the old message text currently contained in *catfile*.

92384 **OPTIONS**

92385 None.

92386 **OPERANDS**

92387 The following operands shall be supported:

92388 *catfile* A pathname of the formatted message catalog. If '-' is specified, standard output  
 92389 shall be used. The format of the message catalog produced is unspecified.

92390 *msgfile* A pathname of a message text source file. If '-' is specified for an instance of  
 92391 *msgfile*, standard input shall be used. The format of message text source files is  
 92392 defined in the EXTENDED DESCRIPTION section.

92393 **STDIN**92394 The standard input shall not be used unless a *msgfile* operand is specified as '-'.92395 **INPUT FILES**

92396 The input files shall be text files.

92397 **ENVIRONMENT VARIABLES**92398 The following environment variables shall affect the execution of *gencat*:

92399 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 92400 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 92401 variables used to determine the values of locale categories.)

92402 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 92403 internationalization variables.

92404 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 92405 characters (for example, single-byte as opposed to multi-byte characters in  
 92406 arguments and input files).

92407 *LC\_MESSAGES*

92408 Determine the locale that should be used to affect the format and contents of  
 92409 diagnostic messages written to standard error.

92410 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

92411 **ASYNCHRONOUS EVENTS**

92412 Default.

92413 **STDOUT**92414 The standard output shall not be used unless the *catfile* operand is specified as '-'.

92415 **STDERR**

92416 The standard error shall be used only for diagnostic messages.

92417 **OUTPUT FILES**

92418 None.

92419 **EXTENDED DESCRIPTION**

92420 The content of a message text file shall be in the format defined as follows. Note that the fields of  
 92421 a message text source line are separated by a single <blank> character. Any other <blank>  
 92422 characters are considered to be part of the subsequent field.

92423 **\$set** *n comment*

92424 This line specifies the set identifier of the following messages until the next **\$set** or  
 92425 end-of-file appears. The *n* denotes the set identifier, which is defined as a number  
 92426 in the range [1, {NL\_SETMAX}] (see the <limits.h> header defined in the Base  
 92427 Definitions volume of POSIX.1-2008). The application shall ensure that set  
 92428 identifiers are presented in ascending order within a single source file, but need  
 92429 not be contiguous. Any string following the set identifier shall be treated as a  
 92430 comment. If no **\$set** directive is specified in a message text source file, all messages  
 92431 shall be located in an implementation-defined default message set NL\_SETD (see  
 92432 the <nl\_types.h> header defined in the Base Definitions volume of POSIX.1-2008).

92433 **\$delset** *n comment*

92434 This line deletes message set *n* from an existing message catalog. The *n* denotes the  
 92435 set number [1, {NL\_SETMAX}]. Any string following the set number shall be  
 92436 treated as a comment.

92437 **\$ comment** A line beginning with ' \$ ' followed by a <blank> shall be treated as a comment.

92438 *m message-text*

92439 The *m* denotes the message identifier, which is defined as a number in the range [1,  
 92440 {NL\_MSGMAX}] (see the <limits.h> header). The *message-text* shall be stored in the  
 92441 message catalog with the set identifier specified by the last **\$set** directive, and with  
 92442 message identifier *m*. If the *message-text* is empty, and a <blank> field separator is  
 92443 present, an empty string shall be stored in the message catalog. If a message source  
 92444 line has a message number, but neither a field separator nor *message-text*, the  
 92445 existing message with that number (if any) shall be deleted from the catalog. The  
 92446 application shall ensure that message identifiers are in ascending order within a  
 92447 single set, but need not be contiguous. The application shall ensure that the length  
 92448 of *message-text* is in the range [0, {NL\_TEXTMAX}] (see the <limits.h> header).

92449 **\$quote** *n* This line specifies an optional quote character *c*, which can be used to surround  
 92450 *message-text* so that trailing <space> characters or null (empty) messages are visible  
 92451 in a message source line. By default, or if an empty **\$quote** directive is supplied, no  
 92452 quoting of *message-text* shall be recognized.

92453 Empty lines in a message text source file shall be ignored. The effects of lines starting with any  
 92454 character other than those defined above are implementation-defined.

92455 Text strings can contain the special characters and escape sequences defined in the following  
 92456 table:

92457  
92458  
92459  
92460  
92461  
92462  
92463  
92464  
92465

| Description       | Symbol | Sequence |
|-------------------|--------|----------|
| <newline>         | NL(LF) | \n       |
| Horizontal-tab    | HT     | \t       |
| <vertical-tab>    | VT     | \v       |
| <backspace>       | BS     | \b       |
| <carriage-return> | CR     | \r       |
| <form-feed>       | FF     | \f       |
| Backslash         | \      | \\       |
| Bit pattern       | ddd    | \ddd     |

92466  
92467  
92468

The escape sequence "\ddd" consists of <backslash> followed by one, two, or three octal digits, which shall be taken to specify the value of the desired character. If the character following a <backslash> is not one of those specified, the <backslash> shall be ignored.

92469  
92470

A <backslash> followed by a <newline> is also used to continue a string on the following line. Thus, the following two lines describe a single message string:

92471  
92472

```
1 This line continues \
to the next line
```

92473

which shall be equivalent to:

92474

```
1 This line continues to the next line
```

## 92475 EXIT STATUS

92476

The following exit values shall be returned:

92477

0 Successful completion.

92478

>0 An error occurred.

## 92479 CONSEQUENCES OF ERRORS

92480

Default.

## 92481 APPLICATION USAGE

92482  
92483  
92484

Message catalogs produced by *gencat* are binary encoded, meaning that their portability cannot be guaranteed between different types of machine. Thus, just as C programs need to be recompiled for each type of machine, so message catalogs must be recreated via *gencat*.

## 92485 EXAMPLES

92486

None.

## 92487 RATIONALE

92488

None.

## 92489 FUTURE DIRECTIONS

92490

None.

## 92491 SEE ALSO

92492

*iconv*

92493

XBD Chapter 8 (on page 173), <limits.h>, <nl\_types.h>

## 92494 CHANGE HISTORY

92495

First released in Issue 3.

## 92496 Issue 6

92497

The normative text is reworded to avoid use of the term “must” for application requirements.

92498 **Issue 7**  
92499

The *gencat* utility is moved from the XSI option to the Base.

## 92500 NAME

92501 get — get a version of an SCCS file (**DEVELOPMENT**)

## 92502 SYNOPSIS

92503 XSI `get [-begkmnlLpst] [-c cutoff] [-i list] [-r SID] [-x list] file...`

## 92504 DESCRIPTION

92505 The *get* utility shall generate a text file from each named SCCS *file* according to the specifications  
92506 given by its options.92507 The generated text shall normally be written into a file called the **g-file** whose name is derived  
92508 from the SCCS filename by simply removing the leading "s.". 

## 92509 OPTIONS

92510 The *get* utility shall conform to XBD [Section 12.2](#) (on page 216).

92511 The following options shall be supported:

92512 **-r** *SID* Indicate the SCCS Identification String (SID) of the version (delta) of an SCCS file  
92513 to be retrieved. The table shows, for the most useful cases, what version of an  
92514 SCCS file is retrieved (as well as the SID of the version to be eventually created by  
92515 *delta* if the **-e** option is also used), as a function of the SID specified.92516 **-c** *cutoff* Indicate the *cutoff* date-time, in the form:92517 `YY[MM[DD[HH[MM[SS]]]]]`92518 For the YY component, values in the range [69,99] shall refer to years 1969 to 1999  
92519 inclusive, and values in the range [00,68] shall refer to years 2000 to 2068 inclusive.92520 **Note:** It is expected that in a future version of this standard the default century inferred  
92521 from a 2-digit year will change. (This would apply to all commands accepting a  
92522 2-digit year as input.)92523 No changes (deltas) to the SCCS file that were created after the specified *cutoff*  
92524 date-time shall be included in the generated text file. Units omitted from the date-  
92525 time default to their maximum possible values; for example, **-c** 7502 is equivalent  
92526 to **-c** 750228235959.92527 Any number of non-numeric characters may separate the various 2-digit pieces of  
92528 the *cutoff* date-time. This feature allows the user to specify a *cutoff* date in the form:  
92529 **-c** "77/2/2 9:22:25".92530 **-e** Indicate that the *get* is for the purpose of editing or making a change (delta) to the  
92531 SCCS file via a subsequent use of *delta*. The **-e** option used in a *get* for a particular  
92532 version (SID) of the SCCS file shall prevent further *get* commands from editing on  
92533 the same SID until *delta* is executed or the **j** (joint edit) flag is set in the SCCS file.  
92534 Concurrent use of *get* **-e** for different SIDs is always allowed.92535 If the **g-file** generated by *get* with a **-e** option is accidentally ruined in the process  
92536 of editing, it may be regenerated by re-executing the *get* command with the **-k**  
92537 option in place of the **-e** option.92538 SCCS file protection specified via the ceiling, floor, and authorized user list stored  
92539 in the SCCS file shall be enforced when the **-e** option is used.92540 **-b** Use with the **-e** option to indicate that the new delta should have an SID in a new  
92541 branch as shown in the table below. This option shall be ignored if the **b** flag is not  
92542 present in the file or if the retrieved delta is not a leaf delta. (A leaf delta is one that  
92543 has no successors on the SCCS file tree.)



|       |                |                                                                                                                     |
|-------|----------------|---------------------------------------------------------------------------------------------------------------------|
| 92544 |                | <b>Note:</b> A branch delta may always be created from a non-leaf delta.                                            |
| 92545 | <b>-i list</b> | Indicate a <i>list</i> of deltas to be included (forced to be applied) in the creation of the                       |
| 92546 |                | generated file. The <i>list</i> has the following syntax:                                                           |
| 92547 |                | <code>&lt;list&gt; ::= &lt;range&gt;   &lt;list&gt; , &lt;range&gt;</code>                                          |
| 92548 |                | <code>&lt;range&gt; ::= SID   SID - SID</code>                                                                      |
| 92549 |                | SID, the SCCS Identification of a delta, may be in any form shown in the "SID                                       |
| 92550 |                | Specified" column of the table in the EXTENDED DESCRIPTION section, except                                          |
| 92551 |                | that the result of supplying a partial SID is unspecified. A diagnostic message shall                               |
| 92552 |                | be written if the first SID in the range is not an ancestor of the second SID in the                                |
| 92553 |                | range.                                                                                                              |
| 92554 | <b>-x list</b> | Indicate a <i>list</i> of deltas to be excluded (forced not to be applied) in the creation of                       |
| 92555 |                | the generated file. See the <b>-i</b> option for the <i>list</i> format.                                            |
| 92556 | <b>-k</b>      | Suppress replacement of identification keywords (see below) in the retrieved text                                   |
| 92557 |                | by their value. The <b>-k</b> option shall be implied by the <b>-e</b> option.                                      |
| 92558 | <b>-l</b>      | Write a delta summary into an <b>l-file</b> .                                                                       |
| 92559 | <b>-L</b>      | Write a delta summary to standard output. All informative output that normally is                                   |
| 92560 |                | written to standard output shall be written to standard error instead, unless the <b>-s</b>                         |
| 92561 |                | option is used, in which case it shall be suppressed.                                                               |
| 92562 | <b>-p</b>      | Write the text retrieved from the SCCS file to the standard output. No <b>g-file</b> shall                          |
| 92563 |                | be created. All informative output that normally goes to the standard output shall                                  |
| 92564 |                | go to standard error instead, unless the <b>-s</b> option is used, in which case it shall                           |
| 92565 |                | disappear.                                                                                                          |
| 92566 | <b>-s</b>      | Suppress all informative output normally written to standard output. However,                                       |
| 92567 |                | fatal error messages (which shall always be written to the standard error) shall                                    |
| 92568 |                | remain unaffected.                                                                                                  |
| 92569 | <b>-m</b>      | Precede each text line retrieved from the SCCS file by the SID of the delta that                                    |
| 92570 |                | inserted the text line in the SCCS file. The format shall be:                                                       |
| 92571 |                | <code>"%s\t%s", &lt;SID&gt;, &lt;text line&gt;</code>                                                               |
| 92572 | <b>-n</b>      | Precede each generated text line with the <b>%M%</b> identification keyword value (see                              |
| 92573 |                | below). The format shall be:                                                                                        |
| 92574 |                | <code>"%s\t%s", &lt;%M% value&gt;, &lt;text line&gt;</code>                                                         |
| 92575 |                | When both the <b>-m</b> and <b>-n</b> options are used, the <code>&lt;text line&gt;</code> shall be replaced by the |
| 92576 |                | <b>-m</b> option-generated format.                                                                                  |
| 92577 | <b>-g</b>      | Suppress the actual retrieval of text from the SCCS file. It is primarily used to                                   |
| 92578 |                | generate an <b>l-file</b> , or to verify the existence of a particular SID.                                         |
| 92579 | <b>-t</b>      | Use to access the most recently created (top) delta in a given release (for example,                                |
| 92580 |                | <b>-r 1</b> ), or release and level (for example, <b>-r 1.2</b> ).                                                  |

**OPERANDS**

The following operands shall be supported:

*file* A pathname of an existing SCCS file or a directory. If *file* is a directory, the *get* utility shall behave as though each file in the directory were specified as a named file, except that non-SCCS files (last component of the pathname does not begin with **s**.) and unreadable files shall be silently ignored.

If exactly one *file* operand appears, and it is **'-'**, the standard input shall be read; each line of the standard input is taken to be the name of an SCCS file to be processed. Non-SCCS files and unreadable files shall be silently ignored.

**STDIN**

The standard input shall be a text file used only if the *file* operand is specified as **'-'**. Each line of the text file shall be interpreted as an SCCS pathname.

**INPUT FILES**

The SCCS files shall be files of an unspecified format.

**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *get*:

**LANG** Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

**LC\_ALL** If set to a non-empty string value, override the values of all the other internationalization variables.

**LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files).

**LC\_MESSAGES**

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error, and informative messages written to standard output (or standard error, if the **-p** option is used).

**NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

**TZ** Determine the timezone in which the times and dates written in the SCCS file are evaluated. If the **TZ** variable is unset or NULL, an unspecified system default timezone is used.

**ASYNCHRONOUS EVENTS**

Default.

**STDOUT**

For each file processed, *get* shall write to standard output the SID being accessed and the number of lines retrieved from the SCCS file, in the following format:

**"%s\n%d lines\n", <SID>, <number of lines>**

If the **-e** option is used, the SID of the delta to be made shall appear after the SID accessed and before the number of lines generated, in the POSIX locale:

**"%s\nnew delta %s\n%d lines\n", <SID accessed>,  
<SID to be made>, <number of lines>**

If there is more than one named file or if a directory or standard input is named, each pathname

shall be written before each of the lines shown in one of the preceding formats:

```
"\n%s:\n", <pathname>
```

If the **-L** option is used, a delta summary shall be written following the format specified below for **l-files**.

If the **-i** option is used, included deltas shall be listed following the notation, in the POSIX locale:

```
"Included:\n"
```

If the **-x** option is used, excluded deltas shall be listed following the notation, in the POSIX locale:

```
"Excluded:\n"
```

If the **-p** or **-L** options are specified, the standard output shall consist of the text retrieved from the SCCS file.

## STDERR

The standard error shall be used only for diagnostic messages, except if the **-p** or **-L** options are specified, it shall include all informative messages normally sent to standard output.

## OUTPUT FILES

Several auxiliary files may be created by *get*. These files are known generically as the **g-file**, **l-file**, **p-file**, and **z-file**. The letter before the <hyphen-minus> is called the *tag*. An auxiliary filename shall be formed from the SCCS filename: the application shall ensure that the last component of all SCCS filenames is of the form *s.module-name*; the auxiliary files shall be named by replacing the leading *s* with the tag. The **g-file** shall be an exception to this scheme: the **g-file** is named by removing the *s*. prefix. For example, for *s.xyz.c*, the auxiliary filenames would be *xyz.c*, *l.xyz.c*, *p.xyz.c*, and *z.xyz.c*, respectively.

The **g-file**, which contains the generated text, shall be created in the current directory (unless the **-p** option is used). A **g-file** shall be created in all cases, whether or not any lines of text were generated by the *get*. It shall be owned by the real user. If the **-k** option is used or implied, the **g-file** shall be writable by the owner only (read-only for everyone else); otherwise, it shall be read-only. Only the real user need have write permission in the current directory.

The **l-file** shall contain a table showing which deltas were applied in generating the retrieved text. The **l-file** shall be created in the current directory if the **-l** option is used; it shall be read-only and it is owned by the real user. Only the real user need have write permission in the current directory.

Lines in the **l-file** shall have the following format:

```
"%c%c%Δ%s\t%sΔ%s\n", <code1>, <code2>, <code3>,
    <SID>, <date-time>, <login>
```

where the entries are:

**<code1>**      A <space> if the delta was applied; ' \* ' otherwise.

**<code2>**      A <space> if the delta was applied or was not applied and ignored; ' \* ' if the delta was not applied and was not ignored.

**<code3>**      A character indicating a special reason why the delta was or was not applied:

**I**      Included.

92665 X Excluded.

92666 C Cut off (by a `-c` option).

92667 `<date-time>` Date and time (using the format of the *date* utility's `%Y/%m/%d %T` conversion  
92668 specification format) of creation.

92669 `<login>` Login name of person who created *delta*.

92670 The comments and MR data shall follow on subsequent lines, indented one `<tab>`. A blank line  
92671 shall terminate each entry.

92672 The **p-file** shall be used to pass information resulting from a *get* with a `-e` option along to *delta*.  
92673 Its contents shall also be used to prevent a subsequent execution of *get* with a `-e` option for the  
92674 same SID until *delta* is executed or the joint edit flag, *j*, is set in the SCCS file. The **p-file** shall be  
92675 created in the directory containing the SCCS file and the application shall ensure that the  
92676 effective user has write permission in that directory. It shall be writable by owner only, and  
92677 owned by the effective user. Each line in the **p-file** shall have the following format:

92678 `"%sΔ%sΔ%sΔ%s%s\n", <g-file SID> ,`  
92679 `<SID of new delta> , <login-name of real user> ,`  
92680 `<date-time> , <i-value> , <x-value>`

92681 where `<i-value>` uses the format " " if no `-i` option was specified, and shall use the format:

92682 `"Δ-i%s", <-i option option-argument>`

92683 if a `-i` option was specified and `<x-value>` uses the format " " if no `-x` option was specified, and  
92684 shall use the format:

92685 `"Δ-x%s", <-x option option-argument>`

92686 if a `-x` option was specified. There can be an arbitrary number of lines in the **p-file** at any time;  
92687 no two lines shall have the same new delta SID.

92688 The **z-file** shall serve as a lock-out mechanism against simultaneous updates. Its contents shall  
92689 be the binary process ID of the command (that is, *get*) that created it. The **z-file** shall be created  
92690 in the directory containing the SCCS file for the duration of *get*. The same protection restrictions  
92691 as those for the **p-file** shall apply for the **z-file**. The **z-file** shall be created read-only.

## 92692 EXTENDED DESCRIPTION

| Determination of SCCS Identification String |                    |                                             |               |                            |
|---------------------------------------------|--------------------|---------------------------------------------|---------------|----------------------------|
| SID* Specified                              | –b Keyletter Used† | Other Conditions                            | SID Retrieved | SID of Delta to be Created |
| none‡                                       | no                 | R defaults to mR                            | mR.mL         | mR.(mL+1)                  |
| none‡                                       | yes                | R defaults to mR                            | mR.mL         | mR.mL.(mB+1).1             |
| R                                           | no                 | R > mR                                      | mR.mL         | R.1***                     |
| R                                           | no                 | R = mR                                      | mR.mL         | mR.(mL+1)                  |
| R                                           | yes                | R > mR                                      | mR.mL         | mR.mL.(mB+1).1             |
| R                                           | yes                | R = mR                                      | mR.mL         | mR.mL.(mB+1).1             |
| R                                           | –                  | R < mR and R does not exist                 | hR.mL**       | hR.mL.(mB+1).1             |
| R                                           | –                  | Trunk successor in release > R and R exists | R.mL          | R.mL.(mB+1).1              |
| R.L                                         | no                 | No trunk successor                          | R.L           | R.(L+1)                    |
| R.L                                         | yes                | No trunk successor                          | R.L           | R.L.(mB+1).1               |
| R.L                                         | –                  | Trunk successor in release ≥ R              | R.L           | R.L.(mB+1).1               |
| R.L.B                                       | no                 | No branch successor                         | R.L.B.mS      | R.L.B.(mS+1)               |
| R.L.B                                       | yes                | No branch successor                         | R.L.B.mS      | R.L.(mB+1).1               |
| R.L.B.S                                     | no                 | No branch successor                         | R.L.B.S       | R.L.B.(S+1)                |
| R.L.B.S                                     | yes                | No branch successor                         | R.L.B.S       | R.L.(mB+1).1               |
| R.L.B.S                                     | –                  | Branch successor                            | R.L.B.S       | R.L.(mB+1).1               |

92715 \* R, L, B, and S are the release, level, branch, and sequence components of the SID,  
 92716 respectively; m means maximum. Thus, for example, R.mL means “the maximum level  
 92717 number within release R”; R.L.(mB+1).1 means “the first sequence number on the new  
 92718 branch (that is, maximum branch number plus one) of level L within release R”. Note  
 92719 that if the SID specified is of the form R.L, R.L.B, or R.L.B.S, each of the specified  
 92720 components shall exist.

92721 \*\* hR is the highest existing release that is lower than the specified, nonexistent, release R.

92722 \*\*\* This is used to force creation of the first delta in a new release.

92723 † The –b option is effective only if the b flag is present in the file. An entry of ‘–’ means  
 92724 “irrelevant”.

92725 ‡ This case applies if the d (default SID) flag is not present in the file. If the d flag is  
 92726 present in the file, then the SID obtained from the d flag is interpreted as if it had been  
 92727 specified on the command line. Thus, one of the other cases in this table applies.

**System Date and Time**

When a **g-file** is generated, the creation time of deltas in the SCCS file may be taken into account. If any of these times are apparently in the future, the behavior is unspecified.

**Identification Keywords**

Identifying information shall be inserted into the text retrieved from the SCCS file by replacing identification keywords with their value wherever they occur. The following keywords may be used in the text stored in an SCCS file:

|     |                                                                                                                                                                                                                             |
|-----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| %M% | Module name: either the value of the <b>m</b> flag in the file, or if absent, the name of the SCCS file with the leading <b>s.</b> removed.                                                                                 |
| %I% | SCCS identification (SID) (%R%.%L% or %R%.%L%.%B%.%S%) of the retrieved text.                                                                                                                                               |
| %R% | Release.                                                                                                                                                                                                                    |
| %L% | Level.                                                                                                                                                                                                                      |
| %B% | Branch.                                                                                                                                                                                                                     |
| %S% | Sequence.                                                                                                                                                                                                                   |
| %D% | Current date (YY/MM/DD).                                                                                                                                                                                                    |
| %H% | Current date (MM/DD/YY).                                                                                                                                                                                                    |
| %T% | Current time (HH:MM:SS).                                                                                                                                                                                                    |
| %E% | Date newest applied delta was created (YY/MM/DD).                                                                                                                                                                           |
| %G% | Date newest applied delta was created (MM/DD/YY).                                                                                                                                                                           |
| %U% | Time newest applied delta was created (HH:MM:SS).                                                                                                                                                                           |
| %Y% | Module type: value of the <b>t</b> flag in the SCCS file.                                                                                                                                                                   |
| %F% | SCCS filename.                                                                                                                                                                                                              |
| %P% | SCCS absolute pathname.                                                                                                                                                                                                     |
| %Q% | The value of the <b>q</b> flag in the file.                                                                                                                                                                                 |
| %C% | Current line number. This keyword is intended for identifying messages output by the program, such as "this should not have happened" type errors. It is not intended to be used on every line to provide sequence numbers. |
| %Z% | The four-character string "@ ( # )" recognizable by <i>what</i> .                                                                                                                                                           |
| %W% | A shorthand notation for constructing <i>what</i> strings:<br>%W%=%Z%%M%<tab>%I%                                                                                                                                            |
| %A% | Another shorthand notation for constructing <i>what</i> strings:<br>%A%=%Z%%Y%%M%%I%%Z%                                                                                                                                     |

**EXIT STATUS**

The following exit values shall be returned:

0 Successful completion.

92764 >0 An error occurred.

## 92765 CONSEQUENCES OF ERRORS

92766 Default.

## 92767 APPLICATION USAGE

92768 Problems can arise if the system date and time have been modified (for example, put forward  
92769 and then back again, or unsynchronized clocks across a network) and can also arise when  
92770 different values of the *TZ* environment variable are used.

92771 Problems of a similar nature can also arise for the operation of the *delta* utility, which compares  
92772 the previous file body against the working file as part of its normal operation.

## 92773 EXAMPLES

92774 None.

## 92775 RATIONALE

92776 None.

## 92777 FUTURE DIRECTIONS

92778 None.

## 92779 SEE ALSO

92780 *admin, delta, prs, what*

92781 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

## 92782 CHANGE HISTORY

92783 First released in Issue 2.

### 92784 Issue 5

92785 A correction is made to the first format string in STDOUT.

92786 The interpretation of the *YY* component of the *-c cutoff* argument is noted.

### 92787 Issue 6

92788 The obsolescent SYNOPSIS is removed, removing the *-lp* option.

92789 The normative text is reworded to avoid use of the term “must” for application requirements.

92790 The Open Group Corrigendum U025/5 is applied, correcting text in the OPTIONS section.

92791 The Open Group Corrigendum U048/1 is applied.

92792 The Open Group Interpretation PIN4C.00014 is applied.

92793 The Open Group Base Resolution bwg2001-007 is applied as follows:

- 92794 • The EXTENDED DESCRIPTION section is updated to make partial SID handling
- 92795 unspecified, reflecting common usage, and to clarify SID ranges.
- 92796 • New text is added to the EXTENDED DESCRIPTION and APPLICATION USAGE sections
- 92797 regarding how the system date and time may be taken into account.
- 92798 • The *TZ* environment variable is added to the ENVIRONMENT VARIABLES section.

### 92799 Issue 7

92800 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



92801 **NAME**

92802       getconf — get configuration values

92803 **SYNOPSIS**92804       getconf [-v specification] *system\_var*92805       getconf [-v specification] *path\_var* *pathname*92806 **DESCRIPTION**92807       In the first synopsis form, the *getconf* utility shall write to the standard output the value of the  
92808       variable specified by the *system\_var* operand.92809       In the second synopsis form, the *getconf* utility shall write to the standard output the value of the  
92810       variable specified by the *path\_var* operand for the path specified by the *pathname* operand.92811       The value of each configuration variable shall be determined as if it were obtained by calling the  
92812       function from which it is defined to be available by this volume of POSIX.1-2008 or by the  
92813       System Interfaces volume of POSIX.1-2008 (see the OPERANDS section). The value shall reflect  
92814       conditions in the current operating environment.92815 **OPTIONS**92816       The *getconf* utility shall conform to XBD [Section 12.2](#) (on page 216).

92817       The following option shall be supported:

92818       -v *specification*92819               Indicate a specific specification and version for which configuration variables shall  
92820               be determined. If this option is not specified, the values returned correspond to an  
92821               implementation default conforming compilation environment.

92822               If the command:

92823               getconf \_POSIX\_V7\_ILP32\_OFF32

92824               does not write "-1\n" or "undefined\n" to standard output, then commands of  
92825               the form:

92826               getconf -v POSIX\_V7\_ILP32\_OFF32 ...

92827               determine values for configuration variables corresponding to the  
92828               POSIX\_V7\_ILP32\_OFF32 compilation environment specified in [c99](#), the  
92829               EXTENDED DESCRIPTION.

92830               If the command:

92831               getconf \_POSIX\_V7\_ILP32\_OFFBIG

92832               does not write "-1\n" or "undefined\n" to standard output, then commands of  
92833               the form:

92834               getconf -v POSIX\_V7\_ILP32\_OFFBIG ...

92835               determine values for configuration variables corresponding to the  
92836               POSIX\_V7\_ILP32\_OFFBIG compilation environment specified in [c99](#), the  
92837               EXTENDED DESCRIPTION.

92838               If the command:

92839               getconf \_POSIX\_V7\_LP64\_OFF64

92840               does not write "-1\n" or "undefined\n" to standard output, then commands of  
92841               the form:



92842 `getconf -v POSIX_V7_LP64_OFF64 ...`  
 92843 determine values for configuration variables corresponding to the  
 92844 POSIX\_V7\_LP64\_OFF64 compilation environment specified in [c99](#), the  
 92845 EXTENDED DESCRIPTION.

92846 If the command:

92847 `getconf _POSIX_V7_LPBIG_OFFBIG`

92848 does not write "-l\n" or "undefined\n" to standard output, then commands of  
 92849 the form:

92850 `getconf -v POSIX_V7_LPBIG_OFFBIG ...`

92851 determine values for configuration variables corresponding to the  
 92852 POSIX\_V7\_LPBIG\_OFFBIG compilation environment specified in [c99](#), the  
 92853 EXTENDED DESCRIPTION.

## 92854 OPERANDS

92855 The following operands shall be supported:

92856 *path\_var* A name of a configuration variable. All of the variables in the Variable column of  
 92857 the table in the DESCRIPTION of the *fpathconf()* function defined in the System  
 92858 Interfaces volume of POSIX.1-2008, without the enclosing braces, shall be  
 92859 supported. The implementation may add other local variables.

92860 *pathname* A pathname for which the variable specified by *path\_var* is to be determined.

92861 *system\_var* A name of a configuration variable. All of the following variables shall be  
 92862 supported:

- 92863 • The names in the Variable column of the table in the DESCRIPTION of the
- 92864 *sysconf()* function in the System Interfaces volume of POSIX.1-2008, except
- 92865 for the entries corresponding to `_SC_CLK_TCK`, `_SC_GETGR_R_SIZE_MAX`,
- 92866 and `_SC_GETPW_R_SIZE_MAX`, without the enclosing braces.

92867 For compatibility with earlier versions, the following variable names shall  
 92868 also be supported:

92869 `POSIX2_C_BIND`  
 92870 `POSIX2_C_DEV`  
 92871 `POSIX2_CHAR_TERM`  
 92872 `POSIX2_FORT_DEV`  
 92873 `POSIX2_FORT_RUN`  
 92874 `POSIX2_LOCALEDEF`  
 92875 `POSIX2_SW_DEV`  
 92876 `POSIX2_UPE`  
 92877 `POSIX2_VERSION`

92878 and shall be equivalent to the same name prefixed with an <underscore>.  
 92879 This requirement may be removed in a future version.

- 92880 • The names of the symbolic constants used as the *name* argument of the
- 92881 *confstr()* function in the System Interfaces volume of POSIX.1-2008, without
- 92882 the `_CS_` prefix.

92883 • The names of the symbolic constants listed under the headings “Maximum  
 92884 Values” and “Minimum Values” in the description of the `<limits.h>` header  
 92885 in the Base Definitions volume of POSIX.1-2008, without the enclosing  
 92886 braces.

92887 For compatibility with earlier versions, the following variable names shall  
 92888 also be supported:

92889 POSIX2\_BC\_BASE\_MAX  
 92890 POSIX2\_BC\_DIM\_MAX  
 92891 POSIX2\_BC\_SCALE\_MAX  
 92892 POSIX2\_BC\_STRING\_MAX  
 92893 POSIX2\_COLL\_WEIGHTS\_MAX  
 92894 POSIX2\_EXPR\_NEST\_MAX  
 92895 POSIX2\_LINE\_MAX  
 92896 POSIX2\_RE\_DUP\_MAX

92897 and shall be equivalent to the same name prefixed with an `<underscore>`.  
 92898 This requirement may be removed in a future version.

92899 The implementation may add other local values.

## 92900 STDIN

92901 Not used.

## 92902 INPUT FILES

92903 None.

## 92904 ENVIRONMENT VARIABLES

92905 The following environment variables shall affect the execution of *getconf*:

92906 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 92907 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 92908 variables used to determine the values of locale categories.)

92909 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 92910 internationalization variables.

92911 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 92912 characters (for example, single-byte as opposed to multi-byte characters in  
 92913 arguments).

92914 *LC\_MESSAGES*

92915 Determine the locale that should be used to affect the format and contents of  
 92916 diagnostic messages written to standard error.

92917 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 92918 ASYNCHRONOUS EVENTS

92919 Default.

## 92920 STDOUT

92921 If the specified variable is defined on the system and its value is described to be available from  
 92922 the *confstr()* function defined in the System Interfaces volume of POSIX.1-2008, its value shall be  
 92923 written in the following format:

92924 "%s\n", *<value>*

92925 Otherwise, if the specified variable is defined on the system, its value shall be written in the

92926 following format:

92927 "%d\n", <value>

92928 If the specified variable is valid, but is undefined on the system, *getconf* shall write using the

92929 following format:

92930 "undefined\n"

92931 If the variable name is invalid or an error occurs, nothing shall be written to standard output.

92932 **STDERR**

92933 The standard error shall be used only for diagnostic messages.

92934 **OUTPUT FILES**

92935 None.

92936 **EXTENDED DESCRIPTION**

92937 None.

92938 **EXIT STATUS**

92939 The following exit values shall be returned:

92940 0 The specified variable is valid and information about its current state was written

92941 successfully.

92942 >0 An error occurred.

92943 **CONSEQUENCES OF ERRORS**

92944 Default.

92945 **APPLICATION USAGE**

92946 None.

92947 **EXAMPLES**

92948 The following example illustrates the value of {NGROUPS\_MAX}:

92949 `getconf NGROUPS_MAX`

92950 The following example illustrates the value of {NAME\_MAX} for a specific directory:

92951 `getconf NAME_MAX /usr`

92952 The following example shows how to deal more carefully with results that might be unspecified:

92953 `if value=$(getconf PATH_MAX /usr); then`

92954 `if [ "$value" = "undefined" ]; then`

92955 `echo PATH_MAX in /usr is indeterminate.`

92956 `else`

92957 `echo PATH_MAX in /usr is $value.`

92958 `fi`

92959 `else`

92960 `echo Error in getconf.`

92961 `fi`

92962 **RATIONALE**

92963 The original need for this utility, and for the *confstr()* function, was to provide a way of finding

92964 the configuration-defined default value for the *PATH* environment variable. Since *PATH* can be

92965 modified by the user to include directories that could contain utilities replacing the standard

92966 utilities, shell scripts need a way to determine the system-supplied *PATH* environment variable

92967 value that contains the correct search path for the standard utilities. It was later suggested that

92968 access to the other variables described in this volume of POSIX.1-2008 could also be useful to  
92969 applications.

92970 This functionality of *getconf* would not be adequately subsumed by another command such as:

92971 `grep var /etc/conf`

92972 because such a strategy would provide correct values for neither those variables that can vary at  
92973 runtime, nor those that can vary depending on the path.

92974 Early proposal versions of *getconf* specified exit status 1 when the specified variable was valid,  
92975 but not defined on the system. The output string "undefined" is now used to specify this case  
92976 with exit code 0 because so many things depend on an exit code of zero when an invoked utility  
92977 is successful.

#### 92978 FUTURE DIRECTIONS

92979 None.

#### 92980 SEE ALSO

92981 *c99*

92982 XBD Chapter 8 (on page 173), Section 12.2 (on page 216), `<limits.h>`

92983 XSH *confstr()*, *fpathconf()*, *sysconf()*, *system()*

#### 92984 CHANGE HISTORY

92985 First released in Issue 4.

#### 92986 Issue 5

92987 In the OPERANDS section:

- 92988 • {NL\_MAX} is changed to {NL\_NMAX}.
- 92989 • Entries beginning NL\_ are deleted from the list of standard configuration variables.
- 92990 • The list of variables previously marked UX is merged with the list marked EX.
- 92991 • Operands are added to support new Option Groups.
- 92992 • Operands are added so that *getconf* can determine supported programming environments.

#### 92993 Issue 6

92994 The Open Group Corrigendum U029/4 is applied, correcting the example command in the last  
92995 paragraph of the OPTIONS section.

92996 The following new requirements on POSIX implementations derive from alignment with the  
92997 Single UNIX Specification:

- 92998 • Operands are added to determine supported programming environments.

92999 This reference page is updated for alignment with the ISO/IEC 9899: 1999 standard. Specifically,  
93000 new macros for *c99* programming environments are introduced.

93001 XSI marked *system\_var* (XBS5\_\*) values are marked LEGACY.

93002 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/27 is applied, correcting the descriptions  
93003 of *path\_var* and *system\_var* in the OPERANDS section.


#### 93004 Issue 7

93005 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

93006 The EXAMPLES section is corrected.

 93007

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0091 [125] is applied.



93008 **NAME**

93009       getopts — parse utility options

93010 **SYNOPSIS**93011       getopts *optstring name* [*arg...*]93012 **DESCRIPTION**

93013       The *getopts* utility shall retrieve options and option-arguments from a list of parameters. It shall  
 93014       support the Utility Syntax Guidelines 3 to 10, inclusive, described in XBD [Section 12.2](#) (on page  
 93015       216).

93016       Each time it is invoked, the *getopts* utility shall place the value of the next option in the shell  
 93017       variable specified by the *name* operand and the index of the next argument to be processed in the  
 93018       shell variable *OPTIND*. Whenever the shell is invoked, *OPTIND* shall be initialized to 1.

93019       When the option requires an option-argument, the *getopts* utility shall place it in the shell  
 93020       variable *OPTARG*. If no option was found, or if the option that was found does not have an  
 93021       option-argument, *OPTARG* shall be unset.

93022       If an option character not contained in the *optstring* operand is found where an option character  
 93023       is expected, the shell variable specified by *name* shall be set to the <question-mark> ('?')  
 93024       character. In this case, if the first character in *optstring* is a <colon> (':'), the shell variable  
 93025       *OPTARG* shall be set to the option character found, but no output shall be written to standard  
 93026       error; otherwise, the shell variable *OPTARG* shall be unset and a diagnostic message shall be  
 93027       written to standard error. This condition shall be considered to be an error detected in the way  
 93028       arguments were presented to the invoking application, but shall not be an error in *getopts*  
 93029       processing.

93030       If an option-argument is missing:

- 93031       • If the first character of *optstring* is a <colon>, the shell variable specified by *name* shall be  
 93032       set to the <colon> character and the shell variable *OPTARG* shall be set to the option  
 93033       character found.
- 93034       • Otherwise, the shell variable specified by *name* shall be set to the <question-mark>  
 93035       character, the shell variable *OPTARG* shall be unset, and a diagnostic message shall be  
 93036       written to standard error. This condition shall be considered to be an error detected in the  
 93037       way arguments were presented to the invoking application, but shall not be an error in  
 93038       *getopts* processing; a diagnostic message shall be written as stated, but the exit status shall  
 93039       be zero.

93040       When the end of options is encountered, the *getopts* utility shall exit with a return value greater  
 93041       than zero; the shell variable *OPTIND* shall be set to the index of the first operand, or the value  
 93042       "\$#"+1 if there are no operands; the *name* variable shall be set to the <question-mark> character.  
 93043       Any of the following shall identify the end of options: the first "--" argument that is not an  
 93044       option-argument, finding an argument that is not an option-argument and does not begin with a  
 93045       '-' , or encountering an error.

93046       The shell variables *OPTIND* and *OPTARG* shall be local to the caller of *getopts* and shall not be  
 93047       exported by default.

93048       The shell variable specified by the *name* operand, *OPTIND*, and *OPTARG* shall affect the current  
 93049       shell execution environment; see [Section 2.12](#) (on page 2371).

93050       If the application sets *OPTIND* to the value 1, a new set of parameters can be used: either the  
 93051       current positional parameters or new *arg* values. Any other attempt to invoke *getopts* multiple  
 93052       times in a single shell execution environment with parameters (positional parameters or *arg*  
 93053       operands) that are not the same in all invocations, or with an *OPTIND* value modified to be a

93054 value other than 1, produces unspecified results.

#### 93055 OPTIONS

93056 None.

#### 93057 OPERANDS

93058 The following operands shall be supported:

93059 *optstring* A string containing the option characters recognized by the utility invoking *getopts*.  
 93060 If a character is followed by a <colon>, the option shall be expected to have an  
 93061 argument, which should be supplied as a separate argument. Applications should  
 93062 specify an option character and its option-argument as separate arguments, but  
 93063 *getopts* shall interpret the characters following an option character requiring  
 93064 arguments as an argument whether or not this is done. An explicit null option-  
 93065 argument need not be recognized if it is not supplied as a separate argument when  
 93066 *getopts* is invoked. (See also the *getopt()* function defined in the System Interfaces  
 93067 volume of POSIX.1-2008.) The characters <question-mark> and <colon> shall not  
 93068 be used as option characters by an application. The use of other option characters  
 93069 that are not alphanumeric produces unspecified results. If the option-argument is  
 93070 not supplied as a separate argument from the option character, the value in  
 93071 *OPTARG* shall be stripped of the option character and the '-'. The first character  
 93072 in *optstring* determines how *getopts* behaves if an option character is not known or  
 93073 an option-argument is missing.

93074 *name* The name of a shell variable that shall be set by the *getopts* utility to the option  
 93075 character that was found.

93076 The *getopts* utility by default shall parse positional parameters passed to the invoking shell  
 93077 procedure. If *args* are given, they shall be parsed instead of the positional parameters.

#### 93078 STDIN

93079 Not used.

#### 93080 INPUT FILES

93081 None.

#### 93082 ENVIRONMENT VARIABLES

93083 The following environment variables shall affect the execution of *getopts*:

93084 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 93085 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 93086 variables used to determine the values of locale categories.)

93087 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 93088 internationalization variables.

93089 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 93090 characters (for example, single-byte as opposed to multi-byte characters in  
 93091 arguments and input files).

93092 *LC\_MESSAGES*

93093 Determine the locale that should be used to affect the format and contents of  
 93094 diagnostic messages written to standard error.

93095 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

93096 *OPTIND* This variable shall be used by the *getopts* utility as the index of the next argument  
 93097 to be processed.

**93098 ASYNCHRONOUS EVENTS**

93099 Default.

**93100 STDOUT**

93101 Not used.

**93102 STDERR**

93103 Whenever an error is detected and the first character in the *optstring* operand is not a <colon>  
93104 (':'), a diagnostic message shall be written to standard error with the following information in  
93105 an unspecified format:

- 93106 • The invoking program name shall be identified in the message. The invoking program  
93107 name shall be the value of the shell special parameter 0 (see [Section 2.5.2](#), on page 2340) at  
93108 the time the *getopts* utility is invoked. A name equivalent to:

93109 `basename "$0"`

93110 may be used.

- 93111 • If an option is found that was not specified in *optstring*, this error is identified and the  
93112 invalid option character shall be identified in the message.

- 93113 • If an option requiring an option-argument is found, but an option-argument is not found,  
93114 this error shall be identified and the invalid option character shall be identified in the  
93115 message.

**93116 OUTPUT FILES**

93117 None.

**93118 EXTENDED DESCRIPTION**

93119 None.

**93120 EXIT STATUS**

93121 The following exit values shall be returned:

93122 0 An option, specified or unspecified by *optstring*, was found.

93123 >0 The end of options was encountered or an error occurred.

**93124 CONSEQUENCES OF ERRORS**

93125 Default.

**93126 APPLICATION USAGE**

93127 Since *getopts* affects the current shell execution environment, it is generally provided as a shell  
93128 regular built-in. If it is called in a subshell or separate utility execution environment, such as one  
93129 of the following:

93130 `(getopts abc value "$@")`

93131 `nohup getopts ...`

93132 `find . -exec getopts ... \;`

93133 it does not affect the shell variables in the caller's environment.

93134 Note that shell functions share *OPTIND* with the calling shell even though the positional  
93135 parameters are changed. If the calling shell and any of its functions uses *getopts* to parse  
93136 arguments, the results are unspecified.



**EXAMPLES**

The following example script parses and displays its arguments:

```
aflag=
bflag=
while getopts ab: name
do
    case $name in
        a)    aflag=1;;
        b)    bflag=1
              bval="$OPTARG";;
        ?)    printf "Usage: %s: [-a] [-b value] args\n" $0
              exit 2;;
    esac
done
if [ ! -z "$aflag" ]; then
    printf "Option -a specified\n"
fi
if [ ! -z "$bflag" ]; then
    printf 'Option -b "%s" specified\n' "$bval"
fi
shift $(( $OPTIND - 1 ))
printf "Remaining arguments are: %s\n" "$@"
```

**RATIONALE**

The *getopts* utility was chosen in preference to the System V *getopt* utility because *getopts* handles option-arguments containing <blank> characters.

The *OPTARG* variable is not mentioned in the ENVIRONMENT VARIABLES section because it does not affect the execution of *getopts*; it is one of the few “output-only” variables used by the standard utilities.

The <colon> is not allowed as an option character because that is not historical behavior, and it violates the Utility Syntax Guidelines. The <colon> is now specified to behave as in the KornShell version of the *getopts* utility; when used as the first character in the *optstring* operand, it disables diagnostics concerning missing option-arguments and unexpected option characters. This replaces the use of the *OPTERR* variable that was specified in an early proposal.

The formats of the diagnostic messages produced by the *getopts* utility and the *getopt()* function are not fully specified because implementations with superior (“friendlier”) formats objected to the formats used by some historical implementations. The standard developers considered it important that the information in the messages used be uniform between *getopts* and *getopt()*. Exact duplication of the messages might not be possible, particularly if a utility is built on another system that has a different *getopt()* function, but the messages must have specific information included so that the program name, invalid option character, and type of error can be distinguished by a user.

Only a rare application program intercepts a *getopts* standard error message and wants to parse it. Therefore, implementations are free to choose the most usable messages they can devise. The following formats are used by many historical implementations:

```
%s: illegal option -- %c\n", <program name>, <option character>

%s: option requires an argument -- %c\n", <program name>, \
    <option character>
```

93184 Historical shells with built-in versions of *getopt()* or *getopts* have used different formats,  
93185 frequently not even indicating the option character found in error.

93186 **FUTURE DIRECTIONS**

93187 None.

93188 **SEE ALSO**

93189 [Section 2.5.2](#) (on page 2340)

93190 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

93191 XSH *getopt()*

93192 **CHANGE HISTORY**

93193 First released in Issue 4.

93194 **Issue 6**

93195 The normative text is reworded to avoid use of the term “must” for application requirements.

93196 **Issue 7**

93197 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0092 [159] is applied.

**NAME**

grep — search a file for a pattern

**SYNOPSIS**

```
grep [-E|-F] [-c|-l|-q] [-insvx] -e pattern_list
    [-e pattern_list]... [-f pattern_file]... [file...]

grep [-E|-F] [-c|-l|-q] [-insvx] [-e pattern_list]...
    -f pattern_file [-f pattern_file]... [file...]

grep [-E|-F] [-c|-l|-q] [-insvx] pattern_list [file...]
```

**DESCRIPTION**

The *grep* utility shall search the input files, selecting lines matching one or more patterns; the types of patterns are controlled by the options specified. The patterns are specified by the *-e* option, *-f* option, or the *pattern\_list* operand. The *pattern\_list*'s value shall consist of one or more patterns separated by <newline> characters; the *pattern\_file*'s contents shall consist of one or more patterns terminated by a <newline> character. By default, an input line shall be selected if any pattern, treated as an entire basic regular expression (BRE) as described in XBD [Section 9.3](#) (on page 183), matches any part of the line excluding the terminating <newline>; a null BRE shall match every line. By default, each selected input line shall be written to the standard output.

Regular expression matching shall be based on text lines. Since a <newline> separates or terminates patterns (see the *-e* and *-f* options below), regular expressions cannot contain a <newline>. Similarly, since patterns are matched against individual lines (excluding the terminating <newline> characters) of the input, there is no way for a pattern to match a <newline> found in the input.

**OPTIONS**

The *grep* utility shall conform to XBD [Section 12.2](#) (on page 216).

The following options shall be supported:

**-E** Match using extended regular expressions. Treat each pattern specified as an ERE, as described in XBD [Section 9.4](#) (on page 188). If any entire ERE pattern matches some part of an input line excluding the terminating <newline>, the line shall be matched. A null ERE shall match every line.

**-F** Match using fixed strings. Treat each pattern specified as a string instead of a regular expression. If an input line contains any of the patterns as a contiguous sequence of bytes, the line shall be matched. A null string shall match every line.

**-c** Write only a count of selected lines to standard output.

**-e pattern\_list**

Specify one or more patterns to be used during the search for input. The application shall ensure that patterns in *pattern\_list* are separated by a <newline>. A null pattern can be specified by two adjacent <newline> characters in *pattern\_list*. Unless the *-E* or *-F* option is also specified, each pattern shall be treated as a BRE, as described in XBD [Section 9.3](#) (on page 183). Multiple *-e* and *-f* options shall be accepted by the *grep* utility. All of the specified patterns shall be used when matching lines, but the order of evaluation is unspecified.

**-f pattern\_file**

Read one or more patterns from the file named by the pathname *pattern\_file*. Patterns in *pattern\_file* shall be terminated by a <newline>. A null pattern can be specified by an empty line in *pattern\_file*. Unless the *-E* or *-F* option is also

- 93244 specified, each pattern shall be treated as a BRE, as described in XBD [Section 9.3](#)  
 93245 (on page 183).
- 93246 **-i** Perform pattern matching in searches without regard to case; see XBD [Section 9.2](#)  
 93247 (on page 182).
- 93248 **-l** (The letter ell.) Write only the names of files containing selected lines to standard  
 93249 output. Pathnames shall be written once per file searched. If the standard input is  
 93250 searched, a pathname of "(standard input)" shall be written, in the POSIX  
 93251 locale. In other locales, "standard input" may be replaced by something more  
 93252 appropriate in those locales.
- 93253 **-n** Precede each output line by its relative line number in the file, each file starting at  
 93254 line 1. The line number counter shall be reset for each file processed.
- 93255 **-q** Quiet. Nothing shall be written to the standard output, regardless of matching  
 93256 lines. Exit with zero status if an input line is selected.
- 93257 **-s** Suppress the error messages ordinarily written for nonexistent or unreadable files.  
 93258 Other error messages shall not be suppressed.
- 93259 **-v** Select lines not matching any of the specified patterns. If the **-v** option is not  
 93260 specified, selected lines shall be those that match any of the specified patterns.
- 93261 **-x** Consider only input lines that use all characters in the line excluding the  
 93262 terminating <newline> to match an entire fixed string or regular expression to be  
 93263 matching lines.

## 93264 OPERANDS

93265 The following operands shall be supported:

- 93266 *pattern\_list* Specify one or more patterns to be used during the search for input. This operand  
 93267 shall be treated as if it were specified as **-e pattern\_list**.
- 93268 *file* A pathname of a file to be searched for the patterns. If no *file* operands are  
 93269 specified, the standard input shall be used.

## 93270 STDIN

93271 The standard input shall be used if no *file* operands are specified, and shall be used if a *file*  
 93272 operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
 93273 the standard input shall not be used. See the INPUT FILES section.

## 93274 INPUT FILES

93275 The input files shall be text files.

## 93276 ENVIRONMENT VARIABLES

93277 The following environment variables shall affect the execution of *grep*:

- 93278 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 93279 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 93280 variables used to determine the values of locale categories.)
- 93281 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 93282 internationalization variables.
- 93283 *LC\_COLLATE*  
 93284 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 93285 character collating elements within regular expressions.

93286 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 93287 characters (for example, single-byte as opposed to multi-byte characters in  
 93288 arguments and input files) and the behavior of character classes within regular  
 93289 expressions.

93290 **LC\_MESSAGES**  
 93291 Determine the locale that should be used to affect the format and contents of  
 93292 diagnostic messages written to standard error.

93293 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

## 93294 **ASYNCHRONOUS EVENTS**

93295 Default.

## 93296 **STDOUT**

93297 If the **-l** option is in effect, the following shall be written for each file containing at least one  
 93298 selected input line:

93299 "%s\n", <file>

93300 Otherwise, if more than one *file* argument appears, and **-q** is not specified, the *grep* utility shall  
 93301 prefix each output line by:

93302 "%s: ", <file>

93303 The remainder of each output line shall depend on the other options specified:

- 93304 • If the **-c** option is in effect, the remainder of each output line shall contain:  
 93305 "%d\n", <count>
- 93306 • Otherwise, if **-c** is not in effect and the **-n** option is in effect, the following shall be written  
 93307 to standard output:  
 93308 "%d: ", <line number>
- 93309 • Finally, the following shall be written to standard output:  
 93310 "%s", <selected-line contents>

## 93311 **STDERR**

93312 The standard error shall be used only for diagnostic messages.

## 93313 **OUTPUT FILES**

93314 None.

## 93315 **EXTENDED DESCRIPTION**

93316 None.

## 93317 **EXIT STATUS**

93318 The following exit values shall be returned:

- 93319 0 One or more lines were selected.
- 93320 1 No lines were selected.
- 93321 >1 An error occurred.

## 93322 **CONSEQUENCES OF ERRORS**

93323 If the **-q** option is specified, the exit status shall be zero if an input line is selected, even if an  
 93324 error was detected. Otherwise, default actions shall be performed.

## APPLICATION USAGE

Care should be taken when using characters in *pattern\_list* that may also be meaningful to the command interpreter. It is safest to enclose the entire *pattern\_list* argument in single-quotes:

' . . . '

The **-e** *pattern\_list* option has the same effect as the *pattern\_list* operand, but is useful when *pattern\_list* begins with the <hyphen-minus> delimiter. It is also useful when it is more convenient to provide multiple patterns as separate arguments.

Multiple **-e** and **-f** options are accepted and *grep* uses all of the patterns it is given while matching input text lines. (Note that the order of evaluation is not specified. If an implementation finds a null string as a pattern, it is allowed to use that pattern first, matching every line, and effectively ignore any other patterns.)

The **-q** option provides a means of easily determining whether or not a pattern (or string) exists in a group of files. When searching several files, it provides a performance improvement (because it can quit as soon as it finds the first match) and requires less care by the user in choosing the set of files to supply as arguments (because it exits zero if it finds a match even if *grep* detected an access or read error on earlier *file* operands).

When using *grep* to process pathnames, it is recommended that LC\_ALL, or at least LC\_CTYPE and LC\_COLLATE, are set to POSIX or C in the environment, since pathnames can contain byte sequences that do not form valid characters in some locales, in which case the utility's behavior would be undefined. In the POSIX locale each byte is a valid single-byte character, and therefore this problem is avoided.

## EXAMPLES

1. To find all uses of the word "Posix" (in any case) in file **text.mm** and write with line numbers:

```
grep -i -n posix text.mm
```

2. To find all empty lines in the standard input:

```
grep ^$
```

or:

```
grep -v .
```

3. Both of the following commands print all lines containing strings "abc" or "def" or both:

```
grep -E 'abc|def'
```

```
grep -F 'abc
def'
```

4. Both of the following commands print all lines matching exactly "abc" or "def":

```
grep -E '^abc$|^def$'
```

```
grep -F -x 'abc
def'
```

**RATIONALE**

This *grep* has been enhanced in an upwards-compatible way to provide the exact functionality of the historical *egrep* and *fgrep* commands as well. It was the clear intention of the standard developers to consolidate the three *greps* into a single command.

The old *egrep* and *fgrep* commands are likely to be supported for many years to come as implementation extensions, allowing historical applications to operate unmodified.

Historical implementations usually silently ignored all but one of multiply-specified **-e** and **-f** options, but were not consistent as to which specification was actually used.

The **-b** option was omitted from the OPTIONS section because block numbers are implementation-defined.

The System V restriction on using **-** to mean standard input was omitted.

A definition of action taken when given a null BRE or ERE is specified. This is an error condition in some historical implementations.

The **-l** option previously indicated that its use was undefined when no files were explicitly named. This behavior was historical and placed an unnecessary restriction on future implementations. It has been removed.

The historical BSD *grep* **-s** option practice is easily duplicated by redirecting standard output to **/dev/null**. The **-s** option required here is from System V.

The **-x** option, historically available only with *fgrep*, is available here for all of the non-obsolete versions.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*sed*

XBD [Chapter 8](#) (on page 173), [Chapter 9](#) (on page 181), [Section 12.2](#) (on page 216)

**CHANGE HISTORY**

First released in Issue 2.

**Issue 6**

The Open Group Corrigendum U029/5 is applied, correcting the SYNOPSIS.

The normative text is reworded to avoid use of the term “must” for application requirements.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/28 is applied, correcting the examples using the *grep* **-F** option which did not match the normative description of the **-F** option.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #092 is applied.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

SD5-XCU-ERN-98 is applied, updating the STDOUT section.



93399 **NAME**

93400 hash — remember or report utility locations

93401 **SYNOPSIS**93402 hash [*utility*...]

93403 hash -r

93404 **DESCRIPTION**

93405 The *hash* utility shall affect the way the current shell environment remembers the locations of  
 93406 utilities found as described in [Section 2.9.1.1](#) (on page 2357). Depending on the arguments  
 93407 specified, it shall add utility locations to its list of remembered locations or it shall purge the  
 93408 contents of the list. When no arguments are specified, it shall report on the contents of the list.

93409 Utilities provided as built-ins to the shell shall not be reported by *hash*.93410 **OPTIONS**93411 The *hash* utility shall conform to XBD [Section 12.2](#) (on page 216).

93412 The following option shall be supported:

93413 -r Forget all previously remembered utility locations.

93414 **OPERANDS**

93415 The following operand shall be supported:

93416 *utility* The name of a utility to be searched for and added to the list of remembered  
 93417 locations. If *utility* contains one or more <slash> characters, the results are  
 93418 unspecified.

93419 **STDIN**

93420 Not used.

93421 **INPUT FILES**

93422 None.

93423 **ENVIRONMENT VARIABLES**93424 The following environment variables shall affect the execution of *hash*:

93425 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 93426 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 93427 variables used to determine the values of locale categories.)

93428 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 93429 internationalization variables.

93430 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 93431 characters (for example, single-byte as opposed to multi-byte characters in  
 93432 arguments).

93433 *LC\_MESSAGES*

93434 Determine the locale that should be used to affect the format and contents of  
 93435 diagnostic messages written to standard error.

93436 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.93437 *PATH* Determine the location of *utility*, as described in XBD [Chapter 8](#) (on page 173).



**93438 ASYNCHRONOUS EVENTS**

93439 Default.

**93440 STDOUT**

93441 The standard output of *hash* shall be used when no arguments are specified. Its format is  
93442 unspecified, but includes the pathname of each utility in the list of remembered locations for the  
93443 current shell environment. This list shall consist of those utilities named in previous *hash*  
93444 invocations that have been invoked, and may contain those invoked and found through the  
93445 normal command search process.

**93446 STDERR**

93447 The standard error shall be used only for diagnostic messages.

**93448 OUTPUT FILES**

93449 None.

**93450 EXTENDED DESCRIPTION**

93451 None.

**93452 EXIT STATUS**

93453 The following exit values shall be returned:

93454 0 Successful completion.

93455 >0 An error occurred.

**93456 CONSEQUENCES OF ERRORS**

93457 Default.

**93458 APPLICATION USAGE**

93459 Since *hash* affects the current shell execution environment, it is always provided as a shell  
93460 regular built-in. If it is called in a separate utility execution environment, such as one of the  
93461 following:

93462 `nohup hash -r`  
93463 `find . -type f | xargs hash`

93464 it does not affect the command search process of the caller's environment.

93465 The *hash* utility may be implemented as an alias—for example, *alias -t -*, in which case utilities  
93466 found through normal command search are not listed by the *hash* command.

93467 The effects of *hash -r* can also be achieved portably by resetting the value of *PATH*; in the  
93468 simplest form, this can be:

93469 `PATH= "$PATH"`

93470 The use of *hash* with *utility* names is unnecessary for most applications, but may provide a  
93471 performance improvement on a few implementations; normally, the hashing process is included  
93472 by default.

**93473 EXAMPLES**

93474 None.

**93475 RATIONALE**

93476 None.

93477 **FUTURE DIRECTIONS**

93478 None.

93479 **SEE ALSO**93480 [Section 2.9.1.1](#) (on page 2357)93481 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)93482 **CHANGE HISTORY**

93483 First released in Issue 2.

93484 **Issue 7**93485 The *hash* utility is moved from the XSI option to the Base.

93486 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0093 [241] is applied.

93487 **NAME**

93488       head — copy the first part of files

93489 **SYNOPSIS**93490       head [-n *number*] [*file*...]93491 **DESCRIPTION**93492       The *head* utility shall copy its input files to the standard output, ending the output for each file at  
93493       a designated point.93494       Copying shall end at the point in each input file indicated by the **-n *number*** option. The option-  
93495       argument *number* shall be counted in units of lines.93496 **OPTIONS**93497       The *head* utility shall conform to XBD [Section 12.2](#) (on page 216).

93498       The following option shall be supported:

93499       **-n *number***   The first *number* lines of each input file shall be copied to standard output. The  
93500                      application shall ensure that the *number* option-argument is a positive decimal  
93501                      integer.93502       When a file contains less than *number* lines, it shall be copied to standard output in its entirety.  
93503       This shall not be an error.93504       If no options are specified, *head* shall act as if **-n 10** had been specified.93505 **OPERANDS**

93506       The following operand shall be supported:

93507       *file*           A pathname of an input file. If no *file* operands are specified, the standard input  
93508                      shall be used.93509 **STDIN**93510       The standard input shall be used if no *file* operands are specified, and shall be used if a *file*  
93511       operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
93512       the standard input shall not be used. See the INPUT FILES section.93513 **INPUT FILES**

93514       Input files shall be text files, but the line length is not restricted to {LINE\_MAX} bytes.

93515 **ENVIRONMENT VARIABLES**93516       The following environment variables shall affect the execution of *head*:93517       **LANG**           Provide a default value for the internationalization variables that are unset or null.  
93518                      (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
93519                      variables used to determine the values of locale categories.)93520       **LC\_ALL**          If set to a non-empty string value, override the values of all the other  
93521                      internationalization variables.93522       **LC\_CTYPE**       Determine the locale for the interpretation of sequences of bytes of text data as  
93523                      characters (for example, single-byte as opposed to multi-byte characters in  
93524                      arguments and input files).93525       **LC\_MESSAGES**93526                      Determine the locale that should be used to affect the format and contents of  
93527                      diagnostic messages written to standard error.

93528 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 93529 ASYNCHRONOUS EVENTS

93530 Default.

## 93531 STDOUT

93532 The standard output shall contain designated portions of the input files.

93533 If multiple *file* operands are specified, *head* shall precede the output for each with the header:

93534 "*\n==> %s <==\n*", *<pathname>*

93535 except that the first header written shall not include the initial *<newline>*.

## 93536 STDERR

93537 The standard error shall be used only for diagnostic messages.

## 93538 OUTPUT FILES

93539 None.

## 93540 EXTENDED DESCRIPTION

93541 None.

## 93542 EXIT STATUS

93543 The following exit values shall be returned:

93544 0 Successful completion.

93545 >0 An error occurred.

## 93546 CONSEQUENCES OF ERRORS

93547 Default.

## 93548 APPLICATION USAGE

93549 When using *head* to process pathnames, it is recommended that *LC\_ALL*, or at least *LC\_CTYPE* |  
 93550 and *LC\_COLLATE*, are set to *POSIX* or *C* in the environment, since pathnames can contain byte |  
 93551 sequences that do not form valid characters in some locales, in which case the utility's behavior |  
 93552 would be undefined. In the *POSIX* locale each byte is a valid single-byte character, and therefore |  
 93553 this problem is avoided.

## 93554 EXAMPLES

93555 To write the first ten lines of all files (except those with a leading period) in the directory:

93556 *head -- \**

## 93557 RATIONALE

93558 Although it is possible to simulate *head* with *sed 10q* for a single file, the standard developers |  
 93559 decided that the popularity of *head* on historical BSD systems warranted its inclusion alongside |  
 93560 *tail*.

93561 *POSIX.1-2008* version of *head* follows the Utility Syntax Guidelines. The *-n* option was added to |  
 93562 this new interface so that *head* and *tail* would be more logically related. Earlier versions of this |  
 93563 standard allowed a *-number* option. This form is no longer specified by *POSIX.1-2008* but may |  
 93564 be present in some implementations.

93565 There is no *-c* option (as there is in *tail*) because it is not historical practice and because other |  
 93566 utilities in this volume of *POSIX.1-2008* provide similar functionality.

93567 **FUTURE DIRECTIONS**

93568 None.

93569 **SEE ALSO**93570 *sed, tail*93571 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)93572 **CHANGE HISTORY**

93573 First released in Issue 4.

93574 **Issue 6**93575 The obsolescent **–number** form is removed.

93576 The normative text is reworded to avoid use of the term “must” for application requirements.

93577 The DESCRIPTION is updated to clarify that when a file contains less than the number of lines  
93578 requested, the entire file is copied to standard output.93579 **Issue 7**

93580 Austin Group Interpretations 1003.1-2001 #027 and #092 are applied.

93581 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

93582 The APPLICATION USAGE section is removed and the EXAMPLES section is corrected.

**NAME**

iconv — codeset conversion

**SYNOPSIS**

```
iconv [-cs] -f frommap -t tomap [file...]
iconv -f fromcode [-cs] [-t tocode] [file...]
iconv -t tocode [-cs] [-f fromcode] [file...]
iconv -l
```

**DESCRIPTION**

The *iconv* utility shall convert the encoding of characters in *file* from one codeset to another and write the results to standard output.

When the options indicate that charmap files are used to specify the codesets (see **OPTIONS**), the codeset conversion shall be accomplished by performing a logical join on the symbolic character names in the two charmaps. The implementation need not support the use of charmap files for codeset conversion unless the POSIX2\_LOCALEDEF symbol is defined on the system.

**OPTIONS**

The *iconv* utility shall conform to XBD [Section 12.2](#) (on page 216).

The following options shall be supported:

**-c** Omit any characters that are invalid in the codeset of the input file from the output. When **-c** is not used, the results of encountering invalid characters in the input stream (either those that are not characters in the codeset of the input file or that have no corresponding character in the codeset of the output file) shall be specified in the system documentation. The presence or absence of **-c** shall not affect the exit status of *iconv*.

**-f** *fromcodeset*

Identify the codeset of the input file. The implementation shall recognize the following two forms of the *fromcodeset* option-argument:

*fromcode* The *fromcode* option-argument must not contain a <slash> character. It shall be interpreted as the name of one of the codeset descriptions provided by the implementation in an unspecified format. Valid values of *fromcode* are implementation-defined.

*frommap* The *frommap* option-argument must contain a <slash> character. It shall be interpreted as the pathname of a charmap file as defined in XBD [Section 6.4](#) (on page 129). If the pathname does not represent a valid, readable charmap file, the results are undefined.

If this option is omitted, the codeset of the current locale shall be used.

**-l** Write all supported *fromcode* and *tocode* values to standard output in an unspecified format.

**-s** Suppress any messages written to standard error concerning invalid characters. When **-s** is not used, the results of encountering invalid characters in the input stream (either those that are not valid characters in the codeset of the input file or that have no corresponding character in the codeset of the output file) shall be specified in the system documentation. The presence or absence of **-s** shall not affect the exit status of *iconv*.



**OUTPUT FILES**

None.

**EXTENDED DESCRIPTION**

None.

**EXIT STATUS**

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

**CONSEQUENCES OF ERRORS**

Default.

**APPLICATION USAGE**

The user must ensure that both charmap files use the same symbolic names for characters the two codesets have in common.

**EXAMPLES**

The following example converts the contents of file **mail.x400** from the ISO/IEC 6937:2001 standard codeset to the ISO/IEC 8859-1:1998 standard codeset, and stores the results in file **mail.local**:

```
iconv -f IS6937 -t IS8859 mail.x400 > mail.local
```

**RATIONALE**

The *iconv* utility can be used portably only when the user provides two charmap files as option-arguments. This is because a single charmap provided by the user cannot reliably be joined with the names in a system-provided character set description. The valid values for *fromcode* and *tocode* are implementation-defined and do not have to have any relation to the charmap mechanisms. As an aid to interactive users, the **-l** option was adopted from the Plan 9 operating system. It writes information concerning these implementation-defined values. The format is unspecified because there are many possible useful formats that could be chosen, such as a matrix of valid combinations of *fromcode* and *tocode*. The **-l** option is not intended for shell script usage; conforming applications will have to use charmaps.

The *iconv* utility may support the conversion between ASCII and EBCDIC-based encodings, but is not required to do so. In an XSI-compliant implementation, the *dd* utility is the only method guaranteed to support conversion between these two character sets.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*dd*, *gencat*

XBD [Section 6.4](#) (on page 129), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

**CHANGE HISTORY**

First released in Issue 3.

**Issue 6**

This utility has been rewritten to align with the IEEE P1003.2b draft standard. Specifically, the ability to use charmap files for conversion has been added.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/29 is applied, making changes to address inconsistencies with the *iconv()* function in the System Interfaces volume of POSIX.1-2008.



93709 **Issue 7**

93710 Austin Group Interpretation 1003.1-2001 #206 is applied, correcting the *tomap* option.

93711 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

93712 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0094 [291] and XCU/TC1-2008/0095  
93713 [291] are applied.

93714 **NAME**93715 `id` — return user identity93716 **SYNOPSIS**93717 `id [user]`93718 `id -G [-n] [user]`93719 `id -g [-nr] [user]`93720 `id -u [-nr] [user]`93721 **DESCRIPTION**

93722 If no *user* operand is provided, the *id* utility shall write the user and group IDs and the  
 93723 corresponding user and group names of the invoking process to standard output. If the effective  
 93724 and real IDs do not match, both shall be written. If multiple groups are supported by the  
 93725 underlying system (see the description of {NGROUPS\_MAX} in the System Interfaces volume of  
 93726 POSIX.1-2008), the supplementary group affiliations of the invoking process shall also be  
 93727 written.

93728 If a *user* operand is provided and the process has appropriate privileges, the user and group IDs  
 93729 of the selected user shall be written. In this case, effective IDs shall be assumed to be identical to  
 93730 real IDs. If the selected user has more than one allowable group membership listed in the group  
 93731 database, these shall be written in the same manner as the supplementary groups described in  
 93732 the preceding paragraph.

93733 **OPTIONS**93734 The *id* utility shall conform to XBD [Section 12.2](#) (on page 216).

93735 The following options shall be supported:

93736 **-G** Output all different group IDs (effective, real, and supplementary) only, using the  
 93737 format "`%u\n`". If there is more than one distinct group affiliation, output each  
 93738 such affiliation, using the format " `%u`", before the <newline> is output.

93739 **-g** Output only the effective group ID, using the format "`%u\n`".

93740 **-n** Output the name in the format "`%s`" instead of the numeric ID using the format  
 93741 "`%u`".

93742 **-r** Output the real ID instead of the effective ID.

93743 **-u** Output only the effective user ID, using the format "`%u\n`".

93744 **OPERANDS**

93745 The following operand shall be supported:

93746 *user* The login name for which information is to be written.

93747 **STDIN**

93748 Not used.

93749 **INPUT FILES**

93750 None.

93751 **ENVIRONMENT VARIABLES**93752 The following environment variables shall affect the execution of *id*:

93753 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 93754 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 93755 variables used to determine the values of locale categories.)

|       |                            |                                                                                                                                                                                                                                                                                              |
|-------|----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 93756 | <i>LC_ALL</i>              | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                     |
| 93757 |                            |                                                                                                                                                                                                                                                                                              |
| 93758 | <i>LC_CTYPE</i>            | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).                                                                                                                    |
| 93759 |                            |                                                                                                                                                                                                                                                                                              |
| 93760 |                            |                                                                                                                                                                                                                                                                                              |
| 93761 | <i>LC_MESSAGES</i>         |                                                                                                                                                                                                                                                                                              |
| 93762 |                            | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.                                                                                                             |
| 93763 |                            |                                                                                                                                                                                                                                                                                              |
| 93764 |                            |                                                                                                                                                                                                                                                                                              |
| 93765 | XSI                        | <i>NLSPATH</i> Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                         |
| 93766 | <b>ASYNCHRONOUS EVENTS</b> |                                                                                                                                                                                                                                                                                              |
| 93767 |                            | Default.                                                                                                                                                                                                                                                                                     |
| 93768 | <b>STDOUT</b>              |                                                                                                                                                                                                                                                                                              |
| 93769 |                            | The following formats shall be used when the <i>LC_MESSAGES</i> locale category specifies the POSIX locale. In other locales, the strings <i>uid</i> , <i>gid</i> , <i>euid</i> , <i>egid</i> , and <i>groups</i> may be replaced with more appropriate strings corresponding to the locale. |
| 93770 |                            |                                                                                                                                                                                                                                                                                              |
| 93771 |                            |                                                                                                                                                                                                                                                                                              |
| 93772 |                            | "uid=%u(%s) gid=%u(%s)\n", <real user ID>, <user-name> ,                                                                                                                                                                                                                                     |
| 93773 |                            | <real group ID>, <group-name>                                                                                                                                                                                                                                                                |
| 93774 |                            | If the effective and real user IDs do not match, the following shall be inserted immediately before the '\n' character in the previous format:                                                                                                                                               |
| 93775 |                            |                                                                                                                                                                                                                                                                                              |
| 93776 |                            | " euid=%u(%s)"                                                                                                                                                                                                                                                                               |
| 93777 |                            | with the following arguments added at the end of the argument list:                                                                                                                                                                                                                          |
| 93778 |                            | <effective user ID>, <effective user-name>                                                                                                                                                                                                                                                   |
| 93779 |                            | If the effective and real group IDs do not match, the following shall be inserted directly before the '\n' character in the format string (and after any addition resulting from the effective and real user IDs not matching):                                                              |
| 93780 |                            |                                                                                                                                                                                                                                                                                              |
| 93781 |                            |                                                                                                                                                                                                                                                                                              |
| 93782 |                            | " egid=%u(%s)"                                                                                                                                                                                                                                                                               |
| 93783 |                            | with the following arguments added at the end of the argument list:                                                                                                                                                                                                                          |
| 93784 |                            | <effective group-ID>, <effective group name>                                                                                                                                                                                                                                                 |
| 93785 |                            | If the process has supplementary group affiliations or the selected user is allowed to belong to multiple groups, the first shall be added directly before the <newline> in the format string:                                                                                               |
| 93786 |                            |                                                                                                                                                                                                                                                                                              |
| 93787 |                            | " groups=%u(%s)"                                                                                                                                                                                                                                                                             |
| 93788 |                            | with the following arguments added at the end of the argument list:                                                                                                                                                                                                                          |
| 93789 |                            | <supplementary group ID>, <supplementary group name>                                                                                                                                                                                                                                         |
| 93790 |                            | and the necessary number of the following added after that for any remaining supplementary group IDs:                                                                                                                                                                                        |
| 93791 |                            |                                                                                                                                                                                                                                                                                              |
| 93792 |                            | ", %u(%s)"                                                                                                                                                                                                                                                                                   |
| 93793 |                            | and the necessary number of the following arguments added at the end of the argument list:                                                                                                                                                                                                   |
| 93794 |                            | <supplementary group ID>, <supplementary group name>                                                                                                                                                                                                                                         |
| 93795 |                            | If any of the user ID, group ID, effective user ID, effective group ID, or supplementary/multiple                                                                                                                                                                                            |

93796 group IDs cannot be mapped by the system into printable user or group names, the  
93797 corresponding "(%s)" and *name* argument shall be omitted from the corresponding format  
93798 string.

93799 When any of the options are specified, the output format shall be as described in the OPTIONS  
93800 section.

#### 93801 **STDERR**

93802 The standard error shall be used only for diagnostic messages.

#### 93803 **OUTPUT FILES**

93804 None.

#### 93805 **EXTENDED DESCRIPTION**

93806 None.

#### 93807 **EXIT STATUS**

93808 The following exit values shall be returned:

93809 0 Successful completion.

93810 >0 An error occurred.

#### 93811 **CONSEQUENCES OF ERRORS**

93812 Default.

#### 93813 **APPLICATION USAGE**

93814 Output produced by the **-G** option and by the default case could potentially produce very long  
93815 lines on systems that support large numbers of supplementary groups. (On systems with user  
93816 and group IDs that are 32-bit integers and with group names with a maximum of 8 bytes per  
93817 name, 93 supplementary groups plus distinct effective and real group and user IDs could  
93818 theoretically overflow the 2048-byte {LINE\_MAX} text file line limit on the default output case.  
93819 It would take about 186 supplementary groups to overflow the 2048-byte barrier using *id -G*).  
93820 This is not expected to be a problem in practice, but in cases where it is a concern, applications  
93821 should consider using *fold -s* before post-processing the output of *id*.

#### 93822 **EXAMPLES**

93823 None.

#### 93824 **RATIONALE**

93825 The functionality provided by the 4 BSD *groups* utility can be simulated using:

93826 `id -Gn [ user ]`

93827 The 4 BSD command *groups* was considered, but it was not included because it did not provide  
93828 the functionality of the *id* utility of the SVID. Also, it was thought that it would be easier to  
93829 modify *id* to provide the additional functionality necessary to systems with multiple groups than  
93830 to invent another command.

93831 The options **-u**, **-g**, **-n**, and **-r** were added to ease the use of *id* with shell commands  
93832 substitution. Without these options it is necessary to use some preprocessor such as *sed* to select  
93833 the desired piece of information. Since output such as that produced by:

93834 `id -u -n`

93835 is frequently wanted, it seemed desirable to add the options.

93836 **FUTURE DIRECTIONS**

93837 None.

93838 **SEE ALSO**93839 *fold, logname, who*93840 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)93841 XSH *getgid()*, *getgroups()*, *getuid()*93842 **CHANGE HISTORY**

93843 First released in Issue 2.

93844 **Issue 7**

93845 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

93846 **NAME**

93847 ipcrm — remove an XSI message queue, semaphore set, or shared memory segment identifier

93848 **SYNOPSIS**93849 XSI `ipcrm [-q msgid|-Q msgkey|-s semid|-S semkey|-m shmid|-M shmkey]...`93850 **DESCRIPTION**93851 The *ipcrm* utility shall remove zero or more message queues, semaphore sets, or shared memory  
93852 segments. The interprocess communication facilities to be removed are specified by the options.93853 Only a user with appropriate privileges shall be allowed to remove an interprocess  
93854 communication facility that was not created by or owned by the user invoking *ipcrm*.93855 **OPTIONS**93856 The *ipcrm* utility shall conform to XBD [Section 12.2](#) (on page 216).

93857 The following options shall be supported:

93858 **-q msgid** Remove the message queue identifier *msgid* from the system and destroy the  
93859 message queue and data structure associated with it.93860 **-m shmid** Remove the shared memory identifier *shmid* from the system. The shared memory  
93861 segment and data structure associated with it shall be destroyed after the last  
93862 detach.93863 **-s semid** Remove the semaphore identifier *semid* from the system and destroy the set of  
93864 semaphores and data structure associated with it.93865 **-Q msgkey** Remove the message queue identifier, created with key *msgkey*, from the system  
93866 and destroy the message queue and data structure associated with it.93867 **-M shmkey** Remove the shared memory identifier, created with key *shmkey*, from the system.  
93868 The shared memory segment and data structure associated with it shall be  
93869 destroyed after the last detach.93870 **-S semkey** Remove the semaphore identifier, created with key *semkey*, from the system and  
93871 destroy the set of semaphores and data structure associated with it.93872 **OPERANDS**

93873 None.

93874 **STDIN**

93875 Not used.

93876 **INPUT FILES**

93877 None.

93878 **ENVIRONMENT VARIABLES**93879 The following environment variables shall affect the execution of *ipcrm*:93880 **LANG** Provide a default value for the internationalization variables that are unset or null.  
93881 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
93882 variables used to determine the values of locale categories.)93883 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
93884 internationalization variables.93885 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
93886 characters (for example, single-byte as opposed to multi-byte characters in  
93887 arguments).

93888 **LC\_MESSAGES**  
93889 Determine the locale that should be used to affect the format and contents of  
93890 diagnostic messages written to standard error.

93891 **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

93892 **ASYNCHRONOUS EVENTS**  
93893 Default.

93894 **STDOUT**  
93895 Not used.

93896 **STDERR**  
93897 The standard error shall be used only for diagnostic messages.

93898 **OUTPUT FILES**  
93899 None.

93900 **EXTENDED DESCRIPTION**  
93901 None.

93902 **EXIT STATUS**  
93903 The following exit values shall be returned:  
93904 0 Successful completion.  
93905 >0 An error occurred.

93906 **CONSEQUENCES OF ERRORS**  
93907 Default.

93908 **APPLICATION USAGE**  
93909 None.

93910 **EXAMPLES**  
93911 None.

93912 **RATIONALE**  
93913 None.

93914 **FUTURE DIRECTIONS**  
93915 None.

93916 **SEE ALSO**  
93917 *ipcs*  
93918 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)  
93919 XSH *msgctl()*, *semctl()*, *shmctl()*

93920 **CHANGE HISTORY**  
93921 First released in Issue 5.

93922 **Issue 7**  
93923 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

93924 **NAME**

93925 `ipcs` — report XSI interprocess communication facilities status

93926 **SYNOPSIS**

93927 XSI `ipcs [-qms] [-a|-bcopt]`

93928 **DESCRIPTION**

93929 The *ipcs* utility shall write information about active interprocess communication facilities.

93930 Without options, information shall be written in short format for message queues, shared  
93931 memory segments, and semaphore sets that are currently active in the system. Otherwise, the  
93932 information that is displayed is controlled by the options specified.

93933 **OPTIONS**

93934 The *ipcs* utility shall conform to XBD [Section 12.2](#) (on page 216).

93935 The *ipcs* utility accepts the following options:

93936 **-q** Write information about active message queues.

93937 **-m** Write information about active shared memory segments.

93938 **-s** Write information about active semaphore sets.

93939 If **-q**, **-m**, or **-s** are specified, only information about those facilities shall be written. If none of  
93940 these three are specified, information about all three shall be written subject to the following  
93941 options:

93942 **-a** Use all print options. (This is a shorthand notation for **-b**, **-c**, **-o**, **-p**, and **-t**.)

93943 **-b** Write information on maximum allowable size. (Maximum number of bytes in  
93944 messages on queue for message queues, size of segments for shared memory, and  
93945 number of semaphores in each set for semaphores.)

93946 **-c** Write creator's user name and group name; see below.

93947 **-o** Write information on outstanding usage. (Number of messages on queue and total  
93948 number of bytes in messages on queue for message queues, and number of  
93949 processes attached to shared memory segments.)

93950 **-p** Write process number information. (Process ID of the last process to send a  
93951 message and process ID of the last process to receive a message on message  
93952 queues, process ID of the creating process, and process ID of the last process to  
93953 attach or detach on shared memory segments.)

93954 **-t** Write time information. (Time of the last control operation that changed the access  
93955 permissions for all facilities, time of the last *msgsnd()* and *msgrcv()* operations on  
93956 message queues, time of the last *shmat()* and *shmdt()* operations on shared  
93957 memory, and time of the last *semop()* operation on semaphores.)

93958 **OPERANDS**

93959 None.

93960 **STDIN**

93961 Not used.

93962 **INPUT FILES**

- 93963 • The group database



- The user database

## ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of *ipcs*:

**LANG** Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

**LC\_ALL** If set to a non-empty string value, override the values of all the other internationalization variables.

**LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).

### LC\_MESSAGES

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

**NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

**TZ** Determine the timezone for the date and time strings written by *ipcs*. If *TZ* is unset or null, an unspecified default timezone shall be used.

## ASYNCHRONOUS EVENTS

Default.

## STDOUT

An introductory line shall be written with the format:

```
"IPC status from %s as of %s\n", <source>, <date>
```

where *<source>* indicates the source used to gather the statistics and *<date>* is the information that would be produced by the *date* command when invoked in the POSIX locale.

The *ipcs* utility then shall create up to three reports depending upon the *-q*, *-m*, and *-s* options. The first report shall indicate the status of message queues, the second report shall indicate the status of shared memory segments, and the third report shall indicate the status of semaphore sets.

If the corresponding facility is not installed or has not been used since the last reboot, then the report shall be written out in the format:

```
"%s facility not in system.\n", <facility>
```

where *<facility>* is *Message Queue*, *Shared Memory*, or *Semaphore*, as appropriate. If the facility has been installed and has been used since the last reboot, column headings separated by one or more *<space>* characters and followed by a *<newline>* shall be written as indicated below followed by the facility name written out using the format:

```
"%s:\n", <facility>
```

where *<facility>* is *Message Queues*, *Shared Memory*, or *Semaphores*, as appropriate. On the second and third reports the column headings need not be written if the last column headings written already provide column headings for all information in that report.

The column headings provided in the first column below and the meaning of the information in those columns shall be given in order below; the letters in parentheses indicate the options that shall cause the corresponding column to appear; “all” means that the column shall always appear. Each column is separated by one or more *<space>* characters. Note that these options

|       |                                                                                          |                                                                                                     |
|-------|------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------|
| 94007 | only determine what information is provided for each report; they do not determine which |                                                                                                     |
| 94008 | reports are written.                                                                     |                                                                                                     |
| 94009 | T (all)                                                                                  | Type of facility:                                                                                   |
| 94010 |                                                                                          | q Message queue.                                                                                    |
| 94011 |                                                                                          | m Shared memory segment.                                                                            |
| 94012 |                                                                                          | s Semaphore.                                                                                        |
| 94013 | This field is a single character written using the format %c.                            |                                                                                                     |
| 94014 | ID (all)                                                                                 | The identifier for the facility entry. This field shall be written using the format                 |
| 94015 |                                                                                          | %d.                                                                                                 |
| 94016 | KEY (all)                                                                                | The key used as an argument to <i>msgget()</i> , <i>semget()</i> , or <i>shmget()</i> to create the |
| 94017 |                                                                                          | facility entry.                                                                                     |
| 94018 |                                                                                          | <b>Note:</b> The key of a shared memory segment is changed to IPC_PRIVATE when the                  |
| 94019 |                                                                                          | segment has been removed until all processes attached to the segment                                |
| 94020 |                                                                                          | detach it.                                                                                          |
| 94021 | This field shall be written using the format 0x%x.                                       |                                                                                                     |
| 94022 | MODE (all)                                                                               | The facility access modes and flags. The mode shall consist of 11 characters                        |
| 94023 |                                                                                          | that are interpreted as follows.                                                                    |
| 94024 |                                                                                          | The first character shall be:                                                                       |
| 94025 |                                                                                          | S If a process is waiting on a <i>msgsnd()</i> operation.                                           |
| 94026 |                                                                                          | – If the above is not true.                                                                         |
| 94027 |                                                                                          | The second character shall be:                                                                      |
| 94028 |                                                                                          | R If a process is waiting on a <i>msgrcv()</i> operation.                                           |
| 94029 |                                                                                          | C or – If the associated shared memory segment is to be cleared when the                            |
| 94030 |                                                                                          | first attach operation is executed.                                                                 |
| 94031 |                                                                                          | – If none of the above is true.                                                                     |
| 94032 | The next nine characters shall be interpreted as three sets of three bits each.          |                                                                                                     |
| 94033 | The first set refers to the owner's permissions; the next to permissions of              |                                                                                                     |
| 94034 | others in the usergroup of the facility entry; and the last to all others. Within        |                                                                                                     |
| 94035 | each set, the first character indicates permission to read, the second character         |                                                                                                     |
| 94036 | indicates permission to write or alter the facility entry, and the last character is     |                                                                                                     |
| 94037 | a <hyphen-minus> ('-').                                                                  |                                                                                                     |
| 94038 | The permissions shall be indicated as follows:                                           |                                                                                                     |
| 94039 |                                                                                          | r If read permission is granted.                                                                    |
| 94040 |                                                                                          | w If write permission is granted.                                                                   |
| 94041 |                                                                                          | a If alter permission is granted.                                                                   |
| 94042 |                                                                                          | – If the indicated permission is not granted.                                                       |
| 94043 | The first character following the permissions specifies if there is an alternate or      |                                                                                                     |
| 94044 | additional access control method associated with the facility. If there is no            |                                                                                                     |
| 94045 | alternate or additional access control method associated with the facility, a            |                                                                                                     |
| 94046 | single <space> shall be written; otherwise, another printable character is               |                                                                                                     |

|       |                                                                          |                                                                                     |
|-------|--------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| 94047 |                                                                          | written.                                                                            |
| 94048 | OWNER (all)                                                              | The user name of the owner of the facility entry. If the user name of the owner     |
| 94049 |                                                                          | is found in the user database, at least the first eight column positions of the     |
| 94050 |                                                                          | name shall be written using the format %s. Otherwise, the user ID of the            |
| 94051 |                                                                          | owner shall be written using the format %d.                                         |
| 94052 | GROUP (all)                                                              | The group name of the owner of the facility entry. If the group name of the         |
| 94053 |                                                                          | owner is found in the group database, at least the first eight column positions     |
| 94054 |                                                                          | of the name shall be written using the format %s. Otherwise, the group ID of        |
| 94055 |                                                                          | the owner shall be written using the format %d.                                     |
| 94056 | The following nine columns shall be only written out for message queues: |                                                                                     |
| 94057 | CREATOR (a,c)                                                            | The user name of the creator of the facility entry. If the user name of the         |
| 94058 |                                                                          | creator is found in the user database, at least the first eight column positions of |
| 94059 |                                                                          | the name shall be written using the format %s. Otherwise, the user ID of the        |
| 94060 |                                                                          | creator shall be written using the format %d.                                       |
| 94061 | CGROUP (a,c)                                                             | The group name of the creator of the facility entry. If the group name of the       |
| 94062 |                                                                          | creator is found in the group database, at least the first eight column positions   |
| 94063 |                                                                          | of the name shall be written using the format %s. Otherwise, the group ID of        |
| 94064 |                                                                          | the creator shall be written using the format %d.                                   |
| 94065 | CBYTES (a,o)                                                             | The number of bytes in messages currently outstanding on the associated             |
| 94066 |                                                                          | message queue. This field shall be written using the format %d.                     |
| 94067 | QNUM (a,o)                                                               | The number of messages currently outstanding on the associated message              |
| 94068 |                                                                          | queue. This field shall be written using the format %d.                             |
| 94069 | QBYTES (a,b)                                                             | The maximum number of bytes allowed in messages outstanding on the                  |
| 94070 |                                                                          | associated message queue. This field shall be written using the format %d.          |
| 94071 | LSPID (a,p)                                                              | The process ID of the last process to send a message to the associated queue.       |
| 94072 |                                                                          | This field shall be written using the format:                                       |
| 94073 |                                                                          | "%d", <pid>                                                                         |
| 94074 |                                                                          | where <pid> is 0 if no message has been sent to the corresponding message           |
| 94075 |                                                                          | queue; otherwise, <pid> shall be the process ID of the last process to send a       |
| 94076 |                                                                          | message to the queue.                                                               |
| 94077 | LRPID (a,p)                                                              | The process ID of the last process to receive a message from the associated         |
| 94078 |                                                                          | queue. This field shall be written using the format:                                |
| 94079 |                                                                          | "%d", <pid>                                                                         |
| 94080 |                                                                          | where <pid> is 0 if no message has been received from the corresponding             |
| 94081 |                                                                          | message queue; otherwise, <pid> shall be the process ID of the last process to      |
| 94082 |                                                                          | receive a message from the queue.                                                   |
| 94083 | STIME (a,t)                                                              | The time the last message was sent to the associated queue. If a message has        |
| 94084 |                                                                          | been sent to the corresponding message queue, the hour, minute, and second          |
| 94085 |                                                                          | of the last time a message was sent to the queue shall be written using the         |
| 94086 |                                                                          | format %d:%2.2d:%2.2d. Otherwise, the format "no-entry" shall be                    |
| 94087 |                                                                          | written.                                                                            |
| 94088 | RTIME (a,t)                                                              | The time the last message was received from the associated queue. If a              |
| 94089 |                                                                          | message has been received from the corresponding message queue, the hour,           |
| 94090 |                                                                          | minute, and second of the last time a message was received from the queue           |

94091 shall be written using the format %d:%2.2d:%2.2d. Otherwise, the format  
 94092 " no-entry" shall be written.

94093 The following eight columns shall be only written out for shared memory segments.

94094 CREATOR (a,c) The user of the creator of the facility entry. If the user name of the creator is  
 94095 found in the user database, at least the first eight column positions of the  
 94096 name shall be written using the format %s. Otherwise, the user ID of the  
 94097 creator shall be written using the format %d.

94098 CGROUP (a,c) The group name of the creator of the facility entry. If the group name of the  
 94099 creator is found in the group database, at least the first eight column positions  
 94100 of the name shall be written using the format %s. Otherwise, the group ID of  
 94101 the creator shall be written using the format %d.

94102 NATTCH (a,o) The number of processes attached to the associated shared memory segment.  
 94103 This field shall be written using the format %d.

94104 SEGSZ (a,b) The size of the associated shared memory segment. This field shall be written  
 94105 using the format %d.

94106 CPID (a,p) The process ID of the creator of the shared memory entry. This field shall be  
 94107 written using the format %d.

94108 LPID (a,p) The process ID of the last process to attach or detach the shared memory  
 94109 segment. This field shall be written using the format:

94110 "%d", <pid>

94111 where <pid> is 0 if no process has attached the corresponding shared memory  
 94112 segment; otherwise, <pid> shall be the process ID of the last process to attach  
 94113 or detach the segment.

94114 ATIME (a,t) The time the last attach on the associated shared memory segment was  
 94115 completed. If the corresponding shared memory segment has ever been  
 94116 attached, the hour, minute, and second of the last time the segment was  
 94117 attached shall be written using the format %d:%2.2d:%2.2d. Otherwise, the  
 94118 format " no-entry" shall be written.

94119 DTIME (a,t) The time the last detach on the associated shared memory segment was  
 94120 completed. If the corresponding shared memory segment has ever been  
 94121 detached, the hour, minute, and second of the last time the segment was  
 94122 detached shall be written using the format %d:%2.2d:%2.2d. Otherwise, the  
 94123 format " no-entry" shall be written.

94124 The following four columns shall be only written out for semaphore sets:

94125 CREATOR (a,c) The user of the creator of the facility entry. If the user name of the creator is  
 94126 found in the user database, at least the first eight column positions of the  
 94127 name shall be written using the format %s. Otherwise, the user ID of the  
 94128 creator shall be written using the format %d.

94129 CGROUP (a,c) The group name of the creator of the facility entry. If the group name of the  
 94130 creator is found in the group database, at least the first eight column positions  
 94131 of the name shall be written using the format %s. Otherwise, the group ID of  
 94132 the creator shall be written using the format %d.

|       |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
|-------|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 94133 | NSEMS (a,b)                   | The number of semaphores in the set associated with the semaphore entry.                                                                                                                                                                                                                                                                                                         |
| 94134 |                               | This field shall be written using the format %d.                                                                                                                                                                                                                                                                                                                                 |
| 94135 | OTIME (a,t)                   | The time the last semaphore operation on the set associated with the semaphore entry was completed. If a semaphore operation has ever been performed on the corresponding semaphore set, the hour, minute, and second of the last semaphore operation on the semaphore set shall be written using the format %d:%2.2d:%2.2d. Otherwise, the format " no-entry" shall be written. |
| 94136 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94137 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94138 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94139 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94140 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94141 |                               | The following column shall be written for all three reports when it is requested:                                                                                                                                                                                                                                                                                                |
| 94142 | CTIME (a,t)                   | The time the associated entry was created or changed. The hour, minute, and second of the time when the associated entry was created shall be written using the format %d:%2.2d:%2.2d.                                                                                                                                                                                           |
| 94143 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94144 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94145 | <b>STDERR</b>                 |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94146 |                               | The standard error shall be used only for diagnostic messages.                                                                                                                                                                                                                                                                                                                   |
| 94147 | <b>OUTPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94148 |                               | None.                                                                                                                                                                                                                                                                                                                                                                            |
| 94149 | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94150 |                               | None.                                                                                                                                                                                                                                                                                                                                                                            |
| 94151 | <b>EXIT STATUS</b>            |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94152 |                               | The following exit values shall be returned:                                                                                                                                                                                                                                                                                                                                     |
| 94153 | 0                             | Successful completion.                                                                                                                                                                                                                                                                                                                                                           |
| 94154 | >0                            | An error occurred.                                                                                                                                                                                                                                                                                                                                                               |
| 94155 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94156 |                               | Default.                                                                                                                                                                                                                                                                                                                                                                         |
| 94157 | <b>APPLICATION USAGE</b>      |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94158 |                               | Things can change while <i>ipcs</i> is running; the information it gives is guaranteed to be accurate only when it was retrieved.                                                                                                                                                                                                                                                |
| 94159 |                               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94160 | <b>EXAMPLES</b>               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94161 |                               | None.                                                                                                                                                                                                                                                                                                                                                                            |
| 94162 | <b>RATIONALE</b>              |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94163 |                               | None.                                                                                                                                                                                                                                                                                                                                                                            |
| 94164 | <b>FUTURE DIRECTIONS</b>      |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94165 |                               | None.                                                                                                                                                                                                                                                                                                                                                                            |
| 94166 | <b>SEE ALSO</b>               |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94167 |                               | <i>ipcrm</i>                                                                                                                                                                                                                                                                                                                                                                     |
| 94168 |                               | XBD <a href="#">Chapter 8</a> (on page 173), <a href="#">Section 12.2</a> (on page 216)                                                                                                                                                                                                                                                                                          |
| 94169 |                               | XSH <i>msgrcv()</i> , <i>msgsnd()</i> , <i>semget()</i> , <i>semop()</i> , <i>shmat()</i> , <i>shmdt()</i> , <i>shmget()</i>                                                                                                                                                                                                                                                     |
| 94170 | <b>CHANGE HISTORY</b>         |                                                                                                                                                                                                                                                                                                                                                                                  |
| 94171 |                               | First released in Issue 5.                                                                                                                                                                                                                                                                                                                                                       |

94172 **Issue 6**

94173 The Open Group Corrigendum U020/1 is applied, correcting the SYNOPSIS.

94174 The Open Group Corrigenda U032/1 and U032/2 are applied, clarifying the output format.

94175 The Open Group Base Resolution bwg98-004 is applied.

94176 **Issue 7**

94177 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

94178 SD5-XCU-ERN-139 is applied, adding the *ipcrm* utility to the SEE ALSO section.

94179 **NAME**94180 `jobs` — display status of jobs in the current session94181 **SYNOPSIS**94182 UP `jobs [-l | -p] [job_id...]`94183 **DESCRIPTION**94184 The *jobs* utility shall display the status of jobs that were started in the current shell environment;  
94185 see [Section 2.12](#) (on page 2371).94186 When *jobs* reports the termination status of a job, the shell shall remove its process ID from the  
94187 list of those “known in the current shell execution environment”; see [Section 2.9.3.1](#) (on page  
94188 2360).94189 **OPTIONS**94190 The *jobs* utility shall conform to XBD [Section 12.2](#) (on page 216).

94191 The following options shall be supported:

94192 **-l** (The letter ell.) Provide more information about each job listed. This information  
94193 shall include the job number, current job, process group ID, state, and the  
94194 command that formed the job.94195 **-p** Display only the process IDs for the process group leaders of the selected jobs.94196 By default, the *jobs* utility shall display the status of all stopped jobs, running background jobs  
94197 and all jobs whose status has changed and have not been reported by the shell.94198 **OPERANDS**

94199 The following operand shall be supported:

94200 *job\_id* Specifies the jobs for which the status is to be displayed. If no *job\_id* is given, the  
94201 status information for all jobs shall be displayed. The format of *job\_id* is described  
94202 in XBD [Section 3.204](#) (on page 66).94203 **STDIN**

94204 Not used.

94205 **INPUT FILES**

94206 None.

94207 **ENVIRONMENT VARIABLES**94208 The following environment variables shall affect the execution of *jobs*:94209 **LANG** Provide a default value for the internationalization variables that are unset or null.  
94210 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
94211 variables used to determine the values of locale categories.)94212 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
94213 internationalization variables.94214 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
94215 characters (for example, single-byte as opposed to multi-byte characters in  
94216 arguments).94217 **LC\_MESSAGES**94218 Determine the locale that should be used to affect the format and contents of  
94219 diagnostic messages written to standard error and informative messages written to  
94220 standard output.



94221 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

94222 **ASYNCHRONOUS EVENTS**

94223 Default.

94224 **STDOUT**

94225 If the **-p** option is specified, the output shall consist of one line for each process ID:

94226 "%d\n", <process ID>

94227 Otherwise, if the **-l** option is not specified, the output shall be a series of lines of the form:

94228 "[%d] %c %s %s\n", <job-number>, <current>, <state>, <command>

94229 where the fields shall be as follows:

94230 <current> The character '+' identifies the job that would be used as a default for the *fg* or *bg* utilities; this job can also be specified using the *job\_id* %+ or "%%". The character '-' identifies the job that would become the default if the current default job were to exit; this job can also be specified using the *job\_id* %-. For other jobs, this field is a <space>. At most one job can be identified with '+' and at most one job can be identified with '-'. If there is any suspended job, then the current job shall be a suspended job. If there are at least two suspended jobs, then the previous job also shall be a suspended job.

94238 <job-number> A number that can be used to identify the process group to the *wait*, *fg*, *bg*, and *kill* utilities. Using these utilities, the job can be identified by prefixing the job number with '% '.

94241 <state> One of the following strings (in the POSIX locale):

94242 **Running** Indicates that the job has not been suspended by a signal and has not exited.

94243

94244 **Done** Indicates that the job completed and returned exit status zero.

94245 **Done(code)** Indicates that the job completed normally and that it exited with the specified non-zero exit status, *code*, expressed as a decimal number.

94246

94247 **Stopped** Indicates that the job was suspended by the SIGTSTP signal.

94248 **Stopped (SIGTSTP)**

94249 Indicates that the job was suspended by the SIGTSTP signal.

94250 **Stopped (SIGSTOP)**

94251 Indicates that the job was suspended by the SIGSTOP signal.

94252 **Stopped (SIGTTIN)**

94253 Indicates that the job was suspended by the SIGTTIN signal.

94254 **Stopped (SIGTTOU)**

94255 Indicates that the job was suspended by the SIGTTOU signal.

94256 The implementation may substitute the string **Suspended** in place of **Stopped**. If the job was terminated by a signal, the format of <state> is unspecified, but it shall be visibly distinct from all of the other <state> formats shown here and shall indicate the name or description of the signal causing the termination.

94257

94258

94259

94260 <command> The associated command that was given to the shell.

94261 If the **-l** option is specified, a field containing the process group ID shall be inserted before the <state> field. Also, more processes in a process group may be output on separate lines, using

94262



94263 only the process ID and *<command>* fields.

#### 94264 **STDERR**

94265 The standard error shall be used only for diagnostic messages.

#### 94266 **OUTPUT FILES**

94267 None.

#### 94268 **EXTENDED DESCRIPTION**

94269 None.

#### 94270 **EXIT STATUS**

94271 The following exit values shall be returned:

94272 0 Successful completion.

94273 >0 An error occurred.

#### 94274 **CONSEQUENCES OF ERRORS**

94275 Default.

#### 94276 **APPLICATION USAGE**

94277 The **-p** option is the only portable way to find out the process group of a job because different  
94278 implementations have different strategies for defining the process group of the job. Usage such  
94279 as *\$(jobs -p)* provides a way of referring to the process group of the job in an implementation-  
94280 independent way.

94281 The *jobs* utility does not work as expected when it is operating in its own utility execution  
94282 environment because that environment has no applicable jobs to manipulate. See the  
94283 APPLICATION USAGE section for *bg*. For this reason, *jobs* is generally implemented as a shell  
94284 regular built-in.

#### 94285 **EXAMPLES**

94286 None.

#### 94287 **RATIONALE**

94288 Both "*%%*" and "*%+*" are used to refer to the current job. Both forms are of equal validity—the  
94289 "*%%*" mirroring "*\$\$*" and "*%+*" mirroring the output of *jobs*. Both forms reflect historical  
94290 practice of the KornShell and the C shell with job control.

94291 The job control features provided by *bg*, *fg*, and *jobs* are based on the KornShell. The standard  
94292 developers examined the characteristics of the C shell versions of these utilities and found that  
94293 differences exist. Despite widespread use of the C shell, the KornShell versions were selected for  
94294 this volume of POSIX.1-2008 to maintain a degree of uniformity with the rest of the KornShell  
94295 features selected (such as the very popular command line editing features).

94296 The *jobs* utility is not dependent on the job control option, as are the seemingly related *bg* and *fg*  
94297 utilities because *jobs* is useful for examining background jobs, regardless of the condition of job  
94298 control. When the user has invoked a *set +m* command and job control has been turned off, *jobs*  
94299 can still be used to examine the background jobs associated with that current session. Similarly,  
94300 *kill* can then be used to kill background jobs with *kill %<background job number>*.

94301 The output for terminated jobs is left unspecified to accommodate various historical systems.  
94302 The following formats have been witnessed:

- 94303 1. **Killed**(*signal name*)
- 94304 2. *signal name*

94305 3. *signal name*(**coredump**)

94306 4. *signal description*– **core dumped**

94307 Most users should be able to understand these formats, although it means that applications have  
94308 trouble parsing them.

94309 The calculation of job IDs was not described since this would suggest an implementation, which  
94310 may impose unnecessary restrictions.

94311 In an early proposal, a **-n** option was included to “Display the status of jobs that have changed,  
94312 exited, or stopped since the last status report”. It was removed because the shell always writes  
94313 any changed status of jobs before each prompt.

#### 94314 **FUTURE DIRECTIONS**

94315 None.

#### 94316 **SEE ALSO**

94317 [Section 2.12](#) (on page 2371), *bg*, *fg*, *kill*, *wait*

94318 XBD [Section 3.204](#) (on page 66), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

#### 94319 **CHANGE HISTORY**

94320 First released in Issue 4.

#### 94321 **Issue 6**

94322 This utility is marked as part of the User Portability Utilities option.

94323 The JC shading is removed as job control is mandatory in this version.

#### 94324 **Issue 7**

94325 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

**NAME**

join — relational database operator

**SYNOPSIS**

```
join [-a file_number|-v file_number] [-e string] [-o list] [-t char]
    [-1 field] [-2 field] file1 file2
```

**DESCRIPTION**

The *join* utility shall perform an equality join on the files *file1* and *file2*. The joined files shall be written to the standard output.

The join field is a field in each file on which the files are compared. The *join* utility shall write one line in the output for each pair of lines in *file1* and *file2* that have join fields that collate equally. The output line by default shall consist of the join field, then the remaining fields from *file1*, then the remaining fields from *file2*. This format can be changed by using the **-o** option (see below). The **-a** option can be used to add unmatched lines to the output. The **-v** option can be used to output only unmatched lines.

The files *file1* and *file2* shall be ordered in the collating sequence of *sort -b* on the fields on which they shall be joined, by default the first in each line. All selected output shall be written in the same collating sequence.

The default input field separators shall be <blank> characters. In this case, multiple separators shall count as one field separator, and leading separators shall be ignored. The default output field separator shall be a <space>.

The field separator and collating sequence can be changed by using the **-t** option (see below).

If the same key appears more than once in either file, all combinations of the set of remaining fields in *file1* and the set of remaining fields in *file2* are output in the order of the lines encountered.

If the input files are not in the appropriate collating sequence, the results are unspecified.

**OPTIONS**

The *join* utility shall conform to XBD [Section 12.2](#) (on page 216).

The following options shall be supported:

**-a file\_number**

Produce a line for each unpairable line in file *file\_number*, where *file\_number* is 1 or 2, in addition to the default output. If both **-a1** and **-a2** are specified, all unpairable lines shall be output.

**-e string**

Replace empty output fields in the list selected by **-o** with the string *string*.

**-o list**

Construct the output line to comprise the fields specified in *list*, each element of which shall have one of the following two forms:

1. *file\_number.field*, where *file\_number* is a file number and *field* is a decimal integer field number
2. 0 (zero), representing the join field

The elements of *list* shall be either <comma>-separated or <blank>-separated, as specified in Guideline 8 of XBD [Section 12.2](#) (on page 216). The fields specified by *list* shall be written for all selected output lines. Fields selected by *list* that do not appear in the input shall be treated as empty output fields. (See the **-e** option.) Only specifically requested fields shall be written. The application shall ensure that *list* is a single command line argument.

94370        **-t** *char*        Use character *char* as a separator, for both input and output. Every appearance of  
 94371        *char* in a line shall be significant. When this option is specified, the collating  
 94372        sequence shall be the same as *sort* without the **-b** option.

94373        **-v** *file\_number*        Instead of the default output, produce a line only for each unpairable line in  
 94374        *file\_number*, where *file\_number* is 1 or 2. If both **-v1** and **-v2** are specified, all  
 94375        unpairable lines shall be output.  
 94376

94377        **-1** *field*        Join on the *field*th field of file 1. Fields are decimal integers starting with 1.

94378        **-2** *field*        Join on the *field*th field of file 2. Fields are decimal integers starting with 1.

#### 94379 OPERANDS

94380        The following operands shall be supported:

94381        *file1, file2*        A pathname of a file to be joined. If either of the *file1* or *file2* operands is '-', the  
 94382        standard input shall be used in its place.

#### 94383 STDIN

94384        The standard input shall be used only if the *file1* or *file2* operand is '-'. See the INPUT FILES  
 94385        section.

#### 94386 INPUT FILES

94387        The input files shall be text files.

#### 94388 ENVIRONMENT VARIABLES

94389        The following environment variables shall affect the execution of *join*:

94390        *LANG*        Provide a default value for the internationalization variables that are unset or null.  
 94391        (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 94392        variables used to determine the values of locale categories.)

94393        *LC\_ALL*        If set to a non-empty string value, override the values of all the other  
 94394        internationalization variables.

94395        *LC\_COLLATE*

94396        Determine the locale of the collating sequence *join* expects to have been used when  
 94397        the input files were sorted.

94398        *LC\_CTYPE*        Determine the locale for the interpretation of sequences of bytes of text data as  
 94399        characters (for example, single-byte as opposed to multi-byte characters in  
 94400        arguments and input files).

94401        *LC\_MESSAGES*

94402        Determine the locale that should be used to affect the format and contents of  
 94403        diagnostic messages written to standard error.

94404 XSI        *NLSPATH*        Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

#### 94405 ASYNCHRONOUS EVENTS

94406        Default.

#### 94407 STDOUT

94408        The *join* utility output shall be a concatenation of selected character fields. When the **-o** option  
 94409        is not specified, the output shall be:

94410        "%s%s%s\n", <*join field*>, <*other file1 fields*>,  
 94411        <*other file2 fields*>

94412        If the join field is not the first field in a file, the <*other file fields*> for that file shall be:

94413        *<fields preceding join field>, <fields following join field>*

94414        When the **-o** option is specified, the output format shall be:

94415        "*%s\n*", *<concatenation of fields>*

94416        where the concatenation of fields is described by the **-o** option, above.

94417        For either format, each field (except the last) shall be written with its trailing separator character.

94418        If the separator is the default (*<blank>* characters), a single *<space>* shall be written after each  
94419        field (except the last).

#### 94420        **STDERR**

94421        The standard error shall be used only for diagnostic messages.

#### 94422        **OUTPUT FILES**

94423        None.

#### 94424        **EXTENDED DESCRIPTION**

94425        None.

#### 94426        **EXIT STATUS**

94427        The following exit values shall be returned:

94428        0    All input files were output successfully.

94429        >0   An error occurred.

#### 94430        **CONSEQUENCES OF ERRORS**

94431        Default.

#### 94432        **APPLICATION USAGE**

94433        Pathnames consisting of numeric digits or of the form *string.string* should not be specified  
94434        directly following the **-o** list.

94435        If the collating sequence of the current locale does not have a total ordering of all characters (see +  
94436        XBD [Section 7.3.2](#), on page 147), *join* treats fields that collate equally but are not identical as +  
94437        being the same. If this behavior is not desired, it can be avoided by forcing the use of the POSIX +  
94438        locale (although this means re-sorting the input files into the POSIX locale collating sequence.) +

94439        When using *join* to process pathnames, it is recommended that LC\_ALL, or at least LC\_CTYPE +  
94440        and LC\_COLLATE, are set to POSIX or C in the environment, since pathnames can contain byte +  
94441        sequences that do not form valid characters in some locales, in which case the utility's behavior +  
94442        would be undefined. In the POSIX locale each byte is a valid single-byte character, and therefore +  
94443        this problem is avoided.

#### 94444        **EXAMPLES**

94445        The **-o 0** field essentially selects the union of the join fields. For example, given file **phone**:

```
94446        !Name                Phone Number
94447        Don                    +1 123-456-7890
94448        Hal                   +1 234-567-8901
94449        Yasushi               +2 345-678-9012
```

94450        and file **fax**:

```
94451        !Name                Fax Number
94452        Don                   +1 123-456-7899
94453        Keith                  +1 456-789-0122
94454        Yasushi               +2 345-678-9011
```

(where the large expanses of white space are meant to each represent a single <tab>), the command:

```
join -t "<tab>" -a 1 -a 2 -e '(unknown)' -o 0,1.2,2.2 phone fax
```

(where <tab> is a literal <tab> character) would produce:

| !Name   | Phone Number    | Fax Number      |
|---------|-----------------|-----------------|
| Don     | +1 123-456-7890 | +1 123-456-7899 |
| Hal     | +1 234-567-8901 | (unknown)       |
| Keith   | (unknown)       | +1 456-789-0122 |
| Yasushi | +2 345-678-9012 | +2 345-678-9011 |

Multiple instances of the same key will produce combinatorial results. The following:

```
fa:
  a x
  a y
  a z
fb:
  a p
```

will produce:

```
a x p
a y p
a z p
```

And the following:

```
fa:
  a b c
  a d e
fb:
  a w x
  a y z
  a o p
```

will produce:

```
a b c w x
a b c y z
a b c o p
a d e w x
a d e y z
a d e o p
```

## RATIONALE

The **-e** option is only effective when used with **-o** because, unless specific fields are identified using **-o**, *join* is not aware of what fields might be empty. The exception to this is the join field, but identifying an empty join field with the **-e** string is not historical practice and some scripts might break if this were changed.

The 0 field in the **-o** list was adopted from the Tenth Edition version of *join* to satisfy international objections that the *join* in the base documents for IEEE Std 1003.2-1992 did not support the “full join” or “outer join” described in relational database literature. Although it has been possible to include a join field in the output (by default, or by field number using **-o**), the join field could not be included for an unpaired line selected by **-a**. The **-o 0** field essentially

94500 selects the union of the join fields.

94501 This sort of outer join was not possible with the *join* commands in the base documents for  
94502 IEEE Std 1003.2-1992. The **-o 0** field was chosen because it is an upwards-compatible change for  
94503 applications. An alternative was considered: have the join field represent the union of the fields  
94504 in the files (where they are identical for matched lines, and one or both are null for unmatched  
94505 lines). This was not adopted because it would break some historical applications.

94506 The ability to specify *file2* as **-** is not historical practice; it was added for completeness.

94507 The **-v** option is not historical practice, but was considered necessary because it permitted the  
94508 writing of *only* those lines that do not match on the join field, as opposed to the **-a** option, which  
94509 prints both lines that do and do not match. This additional facility is parallel with the **-v** option  
94510 of *grep*.

94511 Some historical implementations have been encountered where a blank line in one of the input  
94512 files was considered to be the end of the file; the description in this volume of POSIX.1-2008 does  
94513 not cite this as an allowable case.

94514 Earlier versions of this standard allowed **-j**, **-j1**, **-j2** options, and a form of the **-o** option that  
94515 allowed the *list* option-argument to be multiple arguments. These forms are no longer specified  
94516 by POSIX.1-2008 but may be present in some implementations.

#### 94517 FUTURE DIRECTIONS

94518 None.

#### 94519 SEE ALSO

94520 *awk*, *comm*, *sort*, *uniq*

94521 XBD [Section 7.3.2](#) (on page 147), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

#### 94522 CHANGE HISTORY

94523 First released in Issue 2.

#### 94524 Issue 6

94525 The obsolescent **-j** options and the multi-argument **-o** option are removed in this version.

94526 The normative text is reworded to avoid use of the term “must” for application requirements.

#### 94527 Issue 7

94528 Austin Group Interpretation 1003.1-2001 #027 is applied.

94529 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



## 94530 NAME

94531 kill — terminate or signal processes

## 94532 SYNOPSIS

94533 kill -s *signal\_name* *pid*...94534 kill -l [*exit\_status*]94535 XSI kill [-*signal\_name*] *pid*...94536 kill [-*signal\_number*] *pid*...

## 94537 DESCRIPTION

94538 The *kill* utility shall send a signal to the process or processes specified by each *pid* operand.94539 For each *pid* operand, the *kill* utility shall perform actions equivalent to the *kill()* function  
94540 defined in the System Interfaces volume of POSIX.1-2008 called with the following arguments:

- 94541 • The value of the *pid* operand shall be used as the *pid* argument.
- 94542 • The *sig* argument is the value specified by the *-s* option, *-signal\_number* option, or the
- 94543 *-signal\_name* option, or by SIGTERM, if none of these options is specified.

## 94544 OPTIONS

94545 XSI The *kill* utility shall conform to XBD [Section 12.2](#) (on page 216), except that in the last two  
 94546 SYNOPSIS forms, the *-signal\_number* and *-signal\_name* options are usually more than a single  
 94547 character.

94548 The following options shall be supported:

94549 **-l** (The letter ell.) Write all values of *signal\_name* supported by the implementation, if  
 94550 no operand is given. If an *exit\_status* operand is given and it is a value of the ' ? '   
 94551 shell special parameter (see [Section 2.5.2](#) (on page 2340) and *wait*) corresponding to  
 94552 a process that was terminated by a signal, the *signal\_name* corresponding to the  
 94553 signal that terminated the process shall be written. If an *exit\_status* operand is  
 94554 given and it is the unsigned decimal integer value of a signal number, the  
 94555 *signal\_name* (the symbolic constant name without the **SIG** prefix defined in the  
 94556 Base Definitions volume of POSIX.1-2008) corresponding to that signal shall be  
 94557 written. Otherwise, the results are unspecified.

94558 **-s *signal\_name***

94559 Specify the signal to send, using one of the symbolic names defined in the  
 94560 **<signal.h>** header. Values of *signal\_name* shall be recognized in a case-independent  
 94561 fashion, without the **SIG** prefix. In addition, the symbolic name 0 shall be  
 94562 recognized, representing the signal value zero. The corresponding signal shall be  
 94563 sent instead of SIGTERM.

94564 XSI **-*signal\_name***94565 Equivalent to *-s signal\_name*.94566 XSI **-*signal\_number***

94567 Specify a non-negative decimal integer, *signal\_number*, representing the signal to be  
 94568 used instead of SIGTERM, as the *sig* argument in the effective call to *kill()*. The  
 94569 correspondence between integer values and the *sig* value used is shown in the  
 94570 following list.

94571 The effects of specifying any *signal\_number* other than those listed below are  
 94572 undefined.



|       |                                                                                                 |         |
|-------|-------------------------------------------------------------------------------------------------|---------|
| 94573 | 0                                                                                               | 0       |
| 94574 | 1                                                                                               | SIGHUP  |
| 94575 | 2                                                                                               | SIGINT  |
| 94576 | 3                                                                                               | SIGQUIT |
| 94577 | 6                                                                                               | SIGABRT |
| 94578 | 9                                                                                               | SIGKILL |
| 94579 | 14                                                                                              | SIGALRM |
| 94580 | 15                                                                                              | SIGTERM |
| 94581 | If the first argument is a negative integer, it shall be interpreted as a <i>-signal_number</i> |         |
| 94582 | option, not as a negative <i>pid</i> operand specifying a process group.                        |         |

**OPERANDS**

The following operands shall be supported:

*pid* One of the following:

1. A decimal integer specifying a process or process group to be signaled. The process or processes selected by positive, negative, and zero values of the *pid* operand shall be as described for the *kill()* function. If process number 0 is specified, all processes in the current process group shall be signaled. For the effects of negative *pid* numbers, see the *kill()* function defined in the System Interfaces volume of POSIX.1-2008. If the first *pid* operand is negative, it should be preceded by "--" to keep it from being interpreted as an option.
2. A job control job ID (see XBD [Section 3.204](#), on page 66) that identifies a background process group to be signaled. The job control job ID notation is applicable only for invocations of *kill* in the current shell execution environment; see [Section 2.12](#) (on page 2371).

*exit\_status* A decimal integer specifying a signal number or the exit status of a process terminated by a signal.

**STDIN**

Not used.

**INPUT FILES**

None.

**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *kill*:

*LANG* Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

*LC\_ALL* If set to a non-empty string value, override the values of all the other internationalization variables.

*LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).

94614 **LC\_MESSAGES**  
 94615 Determine the locale that should be used to affect the format and contents of  
 94616 diagnostic messages written to standard error.

94617 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

94618 **ASYNCHRONOUS EVENTS**  
 94619 Default.

94620 **STDOUT**  
 94621 When the *-l* option is not specified, the standard output shall not be used.  
 94622 When the *-l* option is specified, the symbolic name of each signal shall be written in the  
 94623 following format:  
 94624 "%s%c", <signal\_name>, <separator>  
 94625 where the <signal\_name> is in uppercase, without the **SIG** prefix, and the <separator> shall be  
 94626 either a <newline> or a <space>. For the last signal written, <separator> shall be a <newline>.  
 94627 When both the *-l* option and *exit\_status* operand are specified, the symbolic name of the  
 94628 corresponding signal shall be written in the following format:  
 94629 "%s\n", <signal\_name>

94630 **STDERR**  
 94631 The standard error shall be used only for diagnostic messages.

94632 **OUTPUT FILES**  
 94633 None.

94634 **EXTENDED DESCRIPTION**  
 94635 None.

94636 **EXIT STATUS**  
 94637 The following exit values shall be returned:  
 94638 0 At least one matching process was found for each *pid* operand, and the specified signal was  
 94639 successfully processed for at least one matching process.  
 94640 >0 An error occurred.

94641 **CONSEQUENCES OF ERRORS**  
 94642 Default.

94643 **APPLICATION USAGE**  
 94644 Process numbers can be found by using *ps*.  
 94645 The job control job ID notation is not required to work as expected when *kill* is operating in its  
 94646 own utility execution environment. In either of the following examples:  
 94647 nohup kill %1 &  
 94648 system("kill %1");  
 94649 the *kill* operates in a different environment and does not share the shell's understanding of job  
 94650 numbers.

94651 **EXAMPLES**  
 94652 Any of the commands:  
 94653 kill -9 100 -165  
 94654 kill -s kill 100 -165

```
94655      kill -s KILL 100 -165
```

```
94656      sends the SIGKILL signal to the process whose process ID is 100 and to all processes whose
94657      process group ID is 165, assuming the sending process has permission to send that signal to the
94658      specified processes, and that they exist.
```

```
94659      The System Interfaces volume of POSIX.1-2008 and this volume of POSIX.1-2008 do not require
94660      specific signal numbers for any signal_names. Even the -signal_number option provides symbolic
94661      (although numeric) names for signals. If a process is terminated by a signal, its exit status
94662      indicates the signal that killed it, but the exact values are not specified. The kill -l option,
94663      however, can be used to map decimal signal numbers and exit status values into the name of a
94664      signal. The following example reports the status of a terminated job:
```

```
94665      job
94666      stat=$?
94667      if [ $stat -eq 0 ]
94668      then
94669          echo job completed successfully.
94670      elif [ $stat -gt 128 ]
94671      then
94672          echo job terminated by signal SIG$(kill -l $stat).
94673      else
94674          echo job terminated with error code $stat.
94675      fi
```

```
94676      To send the default signal to a process group (say 123), an application should use a command
94677      similar to one of the following:
```

```
94678      kill -TERM -123
94679      kill -- -123
```

#### 94680 RATIONALE

```
94681      The -l option originated from the C shell, and is also implemented in the KornShell. The C shell
94682      output can consist of multiple output lines because the signal names do not always fit on a
94683      single line on some terminal screens. The KornShell output also included the implementation-
94684      defined signal numbers and was considered by the standard developers to be too difficult for
94685      scripts to parse conveniently. The specified output format is intended not only to accommodate
94686      the historical C shell output, but also to permit an entirely vertical or entirely horizontal listing
94687      on systems for which this is appropriate.
```

```
94688      An early proposal invented the name SIGNULL as a signal_name for signal 0 (used by the System
94689      Interfaces volume of POSIX.1-2008 to test for the existence of a process without sending it a
94690      signal). Since the signal_name 0 can be used in this case unambiguously, SIGNULL has been
94691      removed.
```

```
94692      An early proposal also required symbolic signal_names to be recognized with or without the SIG
94693      prefix. Historical versions of kill have not written the SIG prefix for the -l option and have not
94694      recognized the SIG prefix on signal_names. Since neither applications portability nor ease-of-use
94695      would be improved by requiring this extension, it is no longer required.
```

```
94696      To avoid an ambiguity of an initial negative number argument specifying either a signal number
94697      or a process group, POSIX.1-2008 mandates that it is always considered the former by
94698      implementations that support the XSI option. It also requires that conforming applications
94699      always use the "--" options terminator argument when specifying a process group, unless an
94700      option is also specified.
```

```
94701      The -s option was added in response to international interest in providing some form of kill that
```

94702 meets the Utility Syntax Guidelines.

94703 The job control job ID notation is not required to work as expected when *kill* is operating in its  
94704 own utility execution environment. In either of the following examples:

94705 `nohup kill %1 &`  
94706 `system("kill %1");`

94707 the *kill* operates in a different environment and does not understand how the shell has managed  
94708 its job numbers.

#### 94709 FUTURE DIRECTIONS

94710 None.

#### 94711 SEE ALSO

94712 [Chapter 2](#) (on page 2335), *ps*, *wait*

94713 XBD [Section 3.204](#) (on page 66), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216), [<signal.h>](#)

94714 XSH *kill()*

#### 94715 CHANGE HISTORY

94716 First released in Issue 2.

#### 94717 Issue 6

94718 The obsolescent versions of the SYNOPSIS are turned into non-obsolescent features of the XSI  
94719 option, corresponding to a similar change in the *trap* special built-in.

#### 94720 Issue 7

94721 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

94722 **NAME**

94723 lex — generate programs for lexical tasks (DEVELOPMENT)

94724 **SYNOPSIS**94725 CD `lex [-t] [-n|-v] [file...]`94726 **DESCRIPTION**

94727 The *lex* utility shall generate C programs to be used in lexical processing of character input, and  
 94728 that can be used as an interface to *yacc*. The C programs shall be generated from *lex* source code  
 94729 and conform to the ISO C standard, without depending on any undefined, unspecified, or  
 94730 implementation-defined behavior, except in cases where the code is copied directly from the  
 94731 supplied source, or in cases that are documented by the implementation. Usually, the *lex* utility  
 94732 shall write the program it generates to the file **lex.yy.c**; the state of this file is unspecified if *lex*  
 94733 exits with a non-zero exit status. See the EXTENDED DESCRIPTION section for a complete  
 94734 description of the *lex* input language.

94735 **OPTIONS**94736 The *lex* utility shall conform to XBD [Section 12.2](#) (on page 216), except for Guideline 9.

94737 The following options shall be supported:

- 94738 **-n** Suppress the summary of statistics usually written with the **-v** option. If no table  
 94739 sizes are specified in the *lex* source code and the **-v** option is not specified, then **-n**  
 94740 is implied.
- 94741 **-t** Write the resulting program to standard output instead of **lex.yy.c**.
- 94742 **-v** Write a summary of *lex* statistics to the standard output. (See the discussion of *lex*  
 94743 table sizes in [Definitions in lex](#) (on page 2876).) If the **-t** option is specified and **-n**  
 94744 is not specified, this report shall be written to standard error. If table sizes are  
 94745 specified in the *lex* source code, and if the **-n** option is not specified, the **-v** option  
 94746 may be enabled.

94747 **OPERANDS**

94748 The following operand shall be supported:

- 94749 *file* A pathname of an input file. If more than one such *file* is specified, all files shall be  
 94750 concatenated to produce a single *lex* program. If no *file* operands are specified, or if  
 94751 a *file* operand is '-', the standard input shall be used.

94752 **STDIN**

94753 The standard input shall be used if no *file* operands are specified, or if a *file* operand is '-'. See  
 94754 INPUT FILES.

94755 **INPUT FILES**

94756 The input files shall be text files containing *lex* source code, as described in the EXTENDED  
 94757 DESCRIPTION section.

94758 **ENVIRONMENT VARIABLES**94759 The following environment variables shall affect the execution of *lex*:

- 94760 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 94761 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 94762 variables used to determine the values of locale categories.)
- 94763 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 94764 internationalization variables.

- 94765 **LC\_COLLATE**
- 94766 Determine the locale for the behavior of ranges, equivalence classes, and multi-
- 94767 character collating elements within regular expressions. If this variable is not set to
- 94768 the POSIX locale, the results are unspecified.
- 94769 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as
- 94770 characters (for example, single-byte as opposed to multi-byte characters in
- 94771 arguments and input files), and the behavior of character classes within regular
- 94772 expressions. If this variable is not set to the POSIX locale, the results are
- 94773 unspecified.
- 94774 **LC\_MESSAGES**
- 94775 Determine the locale that should be used to affect the format and contents of
- 94776 diagnostic messages written to standard error.
- 94777 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.
- 94778 **ASYNCHRONOUS EVENTS**
- 94779 Default.
- 94780 **STDOUT**
- 94781 If the **-t** option is specified, the text file of C source code output of *lex* shall be written to
- 94782 standard output.
- 94783 If the **-t** option is not specified:
- 94784 • Implementation-defined informational, error, and warning messages concerning the
  - 94785 contents of *lex* source code input shall be written to either the standard output or standard
  - 94786 error.
  - 94787 • If the **-v** option is specified and the **-n** option is not specified, *lex* statistics shall also be
  - 94788 written to either the standard output or standard error, in an implementation-defined
  - 94789 format. These statistics may also be generated if table sizes are specified with a '%'
  - 94790 operator in the *Definitions* section, as long as the **-n** option is not specified.
- 94791 **STDERR**
- 94792 If the **-t** option is specified, implementation-defined informational, error, and warning messages
- 94793 concerning the contents of *lex* source code input shall be written to the standard error.
- 94794 If the **-t** option is not specified:
- 94795 1. Implementation-defined informational, error, and warning messages concerning the
  - 94796 contents of *lex* source code input shall be written to either the standard output or
  - 94797 standard error.
  - 94798 2. If the **-v** option is specified and the **-n** option is not specified, *lex* statistics shall also be
  - 94799 written to either the standard output or standard error, in an implementation-defined
  - 94800 format. These statistics may also be generated if table sizes are specified with a '%'
  - 94801 operator in the *Definitions* section, as long as the **-n** option is not specified.
- 94802 **OUTPUT FILES**
- 94803 A text file containing C source code shall be written to **lex.yy.c**, or to the standard output if the **-t**
- 94804 option is present.
- 94805 **EXTENDED DESCRIPTION**
- 94806 Each input file shall contain *lex* source code, which is a table of regular expressions with
- 94807 corresponding actions in the form of C program fragments.
- 94808 When **lex.yy.c** is compiled and linked with the *lex* library (using the **-ll** operand with *c99*), the

resulting program shall read character input from the standard input and shall partition it into strings that match the given expressions.

When an expression is matched, these actions shall occur:

- The input string that was matched shall be left in *yytext* as a null-terminated string; *yytext* shall either be an external character array or a pointer to a character string. As explained in [Definitions in lex](#) (on page 2876), the type can be explicitly selected using the **%array** or **%pointer** declarations, but the default is implementation-defined.
- The external **int** *yylen* shall be set to the length of the matching string.
- The expression's corresponding program fragment, or action, shall be executed.

During pattern matching, *lex* shall search the set of patterns for the single longest possible match. Among rules that match the same number of characters, the rule given first shall be chosen.

The general format of *lex* source shall be:

```

Definitions
%%
Rules
%%
UserSubroutines

```

The first "%%" is required to mark the beginning of the rules (regular expressions and actions); the second "%%" is required only if user subroutines follow.

Any line in the *Definitions* section beginning with a <blank> shall be assumed to be a C program fragment and shall be copied to the external definition area of the **lex.yy.c** file. Similarly, anything in the *Definitions* section included between delimiter lines containing only "%{" and "%}" shall also be copied unchanged to the external definition area of the **lex.yy.c** file.

Any such input (beginning with a <blank> or within "%{" and "%}" delimiter lines) appearing at the beginning of the *Rules* section before any rules are specified shall be written to **lex.yy.c** after the declarations of variables for the *yylex()* function and before the first line of code in *yylex()*. Thus, user variables local to *yylex()* can be declared here, as well as application code to execute upon entry to *yylex()*.

The action taken by *lex* when encountering any input beginning with a <blank> or within "%{" and "%}" delimiter lines appearing in the *Rules* section but coming after one or more rules is undefined. The presence of such input may result in an erroneous definition of the *yylex()* function.

C-language code in the input shall not contain C-language trigraphs. The C-language code within "%{" and "%}" delimiter lines shall not contain any lines consisting only of "%}", or only of "%%".



## Definitions in lex

*Definitions* appear before the first "%%" delimiter. Any line in this section not contained between "%{" and "%}" lines and not beginning with a <blank> shall be assumed to define a *lex* substitution string. The format of these lines shall be:

*name substitute*

If a *name* does not meet the requirements for identifiers in the ISO C standard, the result is undefined. The string *substitute* shall replace the string {*name*} when it is used in a rule. The *name* string shall be recognized in this context only when the braces are provided and when it does not appear within a bracket expression or within double-quotes.

In the *Definitions* section, any line beginning with a <percent-sign> ('%') character and followed by an alphanumeric word beginning with either 's' or 'S' shall define a set of start conditions. Any line beginning with a '%' followed by a word beginning with either 'x' or 'X' shall define a set of exclusive start conditions. When the generated scanner is in a %s state, patterns with no state specified shall be also active; in a %x state, such patterns shall not be active. The rest of the line, after the first word, shall be considered to be one or more <blank>-separated names of start conditions. Start condition names shall be constructed in the same way as definition names. Start conditions can be used to restrict the matching of regular expressions to one or more states as described in [Regular Expressions in lex](#) (on page 2877).

Implementations shall accept either of the following two mutually-exclusive declarations in the *Definitions* section:

**%array**      Declare the type of *yytext* to be a null-terminated character array.

**%pointer**    Declare the type of *yytext* to be a pointer to a null-terminated character string.

The default type of *yytext* is implementation-defined. If an application refers to *yytext* outside of the scanner source file (that is, via an **extern**), the application shall include the appropriate **%array** or **%pointer** declaration in the scanner source file.

Implementations shall accept declarations in the *Definitions* section for setting certain internal table sizes. The declarations are shown in the following table.

**Table 4-10** Table Size Declarations in *lex*

| Declaration        | Description                        | Minimum Value |
|--------------------|------------------------------------|---------------|
| <b>%p</b> <i>n</i> | Number of positions                | 2 500         |
| <b>%n</b> <i>n</i> | Number of states                   | 500           |
| <b>%a</b> <i>n</i> | Number of transitions              | 2 000         |
| <b>%e</b> <i>n</i> | Number of parse tree nodes         | 1 000         |
| <b>%k</b> <i>n</i> | Number of packed character classes | 1 000         |
| <b>%o</b> <i>n</i> | Size of the output array           | 3 000         |

In the table, *n* represents a positive decimal integer, preceded by one or more <blank> characters. The exact meaning of these table size numbers is implementation-defined. The implementation shall document how these numbers affect the *lex* utility and how they are related to any output that may be generated by the implementation should limitations be encountered during the execution of *lex*. It shall be possible to determine from this output which of the table size values needs to be modified to permit *lex* to successfully generate tables for the input language. The values in the column Minimum Value represent the lowest values conforming implementations shall provide.



**Rules in lex**

The rules in *lex* source files are a table in which the left column contains regular expressions and the right column contains actions (C program fragments) to be executed when the expressions are recognized.

*ERE action*

*ERE action*

...

The extended regular expression (ERE) portion of a row shall be separated from *action* by one or more <blank> characters. A regular expression containing <blank> characters shall be recognized under one of the following conditions:

- The entire expression appears within double-quotes.
- The <blank> characters appear within double-quotes or square brackets.
- Each <blank> is preceded by a <backslash> character.

**User Subroutines in lex**

Anything in the user subroutines section shall be copied to **lex.yy.c** following *yyllex()*.

**Regular Expressions in lex**

The *lex* utility shall support the set of extended regular expressions (see XBD [Section 9.4](#), on page 188), with the following additions and exceptions to the syntax:

"..." Any string enclosed in double-quotes shall represent the characters within the double-quotes as themselves, except that <backslash>-escapes (which appear in the following table) shall be recognized. Any <backslash>-escape sequence shall be terminated by the closing quote. For example, "\01"1" represents a single string: the octal value 1 followed by the character '1'.

<state>*r*, <state1, state2, ...>*r*

The regular expression *r* shall be matched only when the program is in one of the start conditions indicated by *state*, *state1*, and so on; see [Actions in lex](#) (on page 2879). (As an exception to the typographical conventions of the rest of this volume of POSIX.1-2008, in this case <state> does not represent a metavariable, but the literal angle-bracket characters surrounding a symbol.) The start condition shall be recognized as such only at the beginning of a regular expression.

*r/x*

The regular expression *r* shall be matched only if it is followed by an occurrence of regular expression *x* (*x* is the instance of trailing context, further defined below). The token returned in *yytext* shall only match *r*. If the trailing portion of *r* matches the beginning of *x*, the result is unspecified. The *r* expression cannot include further trailing context or the '\$' (match-end-of-line) operator; *x* cannot include the '^' (match-beginning-of-line) operator, nor trailing context, nor the '\$' operator. That is, only one occurrence of trailing context is allowed in a *lex* regular expression, and the '^' operator only can be used at the beginning of such an expression.

{*name*}

When *name* is one of the substitution symbols from the *Definitions* section, the string, including the enclosing braces, shall be replaced by the *substitute* value. The *substitute* value shall be treated in the extended regular expression as if it were enclosed in parentheses. No substitution shall occur if {*name*} occurs within a bracket expression or within double-quotes.

Within an ERE, a <backslash> character shall be considered to begin an escape sequence as specified in the table in XBD Chapter 5 (on page 121) ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v'). In addition, the escape sequences in the following table shall be recognized.

A literal <newline> cannot occur within an ERE; the escape sequence '\n' can be used to represent a <newline>. A <newline> shall not be matched by a period operator.

**Table 4-11** Escape Sequences in *lex*

| Escape Sequence | Description                                                                                                                                                                                                             | Meaning                                                                                                                                                                                                                         |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \digits         | A <backslash> character followed by the longest sequence of one, two, or three octal-digit characters (01234567). If all of the digits are 0 (that is, representation of the NUL character), the behavior is undefined. | The character whose encoding is represented by the one, two, or three-digit octal integer. Multi-byte characters require multiple, concatenated escape sequences of this type, including the leading <backslash> for each byte. |
| \xdigits        | A <backslash> character followed by the longest sequence of hexadecimal-digit characters (01234567abcdefABCDEF). If all of the digits are 0 (that is, representation of the NUL character), the behavior is undefined.  | The character whose encoding is represented by the hexadecimal integer.                                                                                                                                                         |
| \c              | A <backslash> character followed by any character not described in this table or in the table in XBD Chapter 5 (on page 121) ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v').                                          | The character 'c', unchanged.                                                                                                                                                                                                   |

**Note:** If a '\x' sequence needs to be immediately followed by a hexadecimal digit character, a sequence such as "\x1" "1" can be used, which represents a character containing the value 1, followed by the character '1'.

The order of precedence given to extended regular expressions for *lex* differs from that specified in XBD Section 9.4 (on page 188). The order of precedence for *lex* shall be as shown in the following table, from high to low.

**Note:** The escaped characters entry is not meant to imply that these are operators, but they are included in the table to show their relationships to the true operators. The start condition, trailing context, and anchoring notations have been omitted from the table because of the placement restrictions described in this section; they can only appear at the beginning or ending of an ERE.

Table 4-12 ERE Precedence in *lex*

| Extended Regular Expression       | Precedence              |
|-----------------------------------|-------------------------|
| collation-related bracket symbols | [ = = ] [ : : ] [ . . ] |
| escaped characters                | \<special character>    |
| bracket expression                | [ ]                     |
| quoting                           | " . . . "               |
| grouping                          | ( )                     |
| definition                        | { name }                |
| single-character RE duplication   | * + ?                   |
| concatenation                     |                         |
| interval expression               | { m , n }               |
| alternation                       |                         |

The ERE anchoring operators '*^*' and '*\$*' do not appear in the table. With *lex* regular expressions, these operators are restricted in their use: the '*^*' operator can only be used at the beginning of an entire regular expression, and the '*\$*' operator only at the end. The operators apply to the entire regular expression. Thus, for example, the pattern "*(^abc)|(def\$)*" is undefined; it can instead be written as two separate rules, one with the regular expression "*^abc*" and one with "*def\$*", which share a common action via the special '*|*' action (see below). If the pattern were written "*^abc|def\$*", it would match either "*abc*" or "*def*" on a line by itself.

Unlike the general ERE rules, embedded anchoring is not allowed by most historical *lex* implementations. An example of embedded anchoring would be for patterns such as "*(^| )foo( |\$)*" to match "*foo*" when it exists as a complete word. This functionality can be obtained using existing *lex* features:

```

^foo/[ \n]      |
" foo"/[ \n]    /* Found foo as a separate word. */

```

Note also that '*\$*' is a form of trailing context (it is equivalent to "*/\n*") and as such cannot be used with regular expressions containing another instance of the operator (see the preceding discussion of trailing context).

The additional regular expressions trailing-context operator '*/'*' can be used as an ordinary character if presented within double-quotes, "*/"*"; preceded by a <backslash>, "*\"*"; or within a bracket expression, "*[ / ]*". The start-condition '*<*' and '*>*' operators shall be special only in a start condition at the beginning of a regular expression; elsewhere in the regular expression they shall be treated as ordinary characters.

### Actions in *lex*

The action to be taken when an ERE is matched can be a C program fragment or the special actions described below; the program fragment can contain one or more C statements, and can also include special actions. The empty C statement '*;*' shall be a valid action; any string in the *lex.yy.c* input that matches the pattern portion of such a rule is effectively ignored or skipped. However, the absence of an action shall not be valid, and the action *lex* takes in such a condition is undefined.

The specification for an action, including C statements and special actions, can extend across several lines if enclosed in braces:

```

ERE <one or more blanks> { program statement
                           program statement }

```

95015 The program statements shall not contain unbalanced curly brace preprocessing tokens.

95016 The default action when a string in the input to a **lex.yy.c** program is not matched by any  
 95017 expression shall be to copy the string to the output. Because the default behavior of a program  
 95018 generated by *lex* is to read the input and copy it to the output, a minimal *lex* source program that  
 95019 has just "%%" shall generate a C program that simply copies the input to the output unchanged.

95020 Four special actions shall be available:

95021 | ECHO; REJECT; BEGIN

95022 | The action ' | ' means that the action for the next rule is the action for this rule.  
 95023 Unlike the other three actions, ' | ' cannot be enclosed in braces or be  
 95024 <semicolon>-terminated; the application shall ensure that it is specified alone, with  
 95025 no other actions.

95026 **ECHO;** Write the contents of the string *yytext* on the output.

95027 **REJECT;** Usually only a single expression is matched by a given string in the input.  
 95028 **REJECT** means "continue to the next expression that matches the current input",  
 95029 and shall cause whatever rule was the second choice after the current rule to be  
 95030 executed for the same input. Thus, multiple rules can be matched and executed for  
 95031 one input string or overlapping input strings. For example, given the regular  
 95032 expressions "xyz" and "xy" and the input "xyz", usually only the regular  
 95033 expression "xyz" would match. The next attempted match would start after **z**. If  
 95034 the last action in the "xyz" rule is **REJECT**, both this rule and the "xy" rule  
 95035 would be executed. The **REJECT** action may be implemented in such a fashion that  
 95036 flow of control does not continue after it, as if it were equivalent to a **goto** to  
 95037 another part of *yylex()*. The use of **REJECT** may result in somewhat larger and  
 95038 slower scanners.

95039 **BEGIN** The action:

95040 BEGIN *newstate*;

95041 switches the state (start condition) to *newstate*. If the string *newstate* has not been  
 95042 declared previously as a start condition in the *Definitions* section, the results are  
 95043 unspecified. The initial state is indicated by the digit '0' or the token **INITIAL**.

95044 The functions or macros described below are accessible to user code included in the *lex* input. It  
 95045 is unspecified whether they appear in the C code output of *lex*, or are accessible only through the  
 95046 **-ll** operand to *c99* (the *lex* library).

95047 **int yylex(void)**  
 95048 Performs lexical analysis on the input; this is the primary function generated by the *lex*  
 95049 utility. The function shall return zero when the end of input is reached; otherwise, it shall  
 95050 return non-zero values (tokens) determined by the actions that are selected.

95051 **int yymore(void)**  
 95052 When called, indicates that when the next input string is recognized, it is to be appended to  
 95053 the current value of *yytext* rather than replacing it; the value in *yyleng* shall be adjusted  
 95054 accordingly.

95055 **int yyless(int n)**  
 95056 Retains *n* initial characters in *yytext*, NUL-terminated, and treats the remaining characters as  
 95057 if they had not been read; the value in *yyleng* shall be adjusted accordingly.

**int input(void)**

Returns the next character from the input, or zero on end-of-file. It shall obtain input from the stream pointer *yyin*, although possibly via an intermediate buffer. Thus, once scanning has begun, the effect of altering the value of *yyin* is undefined. The character read shall be removed from the input stream of the scanner without any processing by the scanner.

**int unput(int c)**

Returns the character 'c' to the input; *yytext* and *yylen* are undefined until the next expression is matched. The result of using *unput()* for more characters than have been input is unspecified.

The following functions shall appear only in the *lex* library accessible through the `-ll` operand; they can therefore be redefined by a conforming application:

**int yywrap(void)**

Called by *yylex()* at end-of-file; the default *yywrap()* shall always return 1. If the application requires *yylex()* to continue processing with another source of input, then the application can include a function *yywrap()*, which associates another file with the external variable **FILE \* yyin** and shall return a value of zero.

**int main(int argc, char \*argv[])**

Calls *yylex()* to perform lexical analysis, then exits. The user code can contain *main()* to perform application-specific operations, calling *yylex()* as applicable.

Except for *input()*, *unput()*, and *main()*, all external and static names generated by *lex* shall begin with the prefix **yy** or **YY**.

**EXIT STATUS**

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

**CONSEQUENCES OF ERRORS**

Default.

**APPLICATION USAGE**

Conforming applications are warned that in the *Rules* section, an ERE without an action is not acceptable, but need not be detected as erroneous by *lex*. This may result in compilation or runtime errors.

The purpose of *input()* is to take characters off the input stream and discard them as far as the lexical analysis is concerned. A common use is to discard the body of a comment once the beginning of a comment is recognized.

The *lex* utility is not fully internationalized in its treatment of regular expressions in the *lex* source code or generated lexical analyzer. It would seem desirable to have the lexical analyzer interpret the regular expressions given in the *lex* source according to the environment specified when the lexical analyzer is executed, but this is not possible with the current *lex* technology. Furthermore, the very nature of the lexical analyzers produced by *lex* must be closely tied to the lexical requirements of the input language being described, which is frequently locale-specific anyway. (For example, writing an analyzer that is used for French text is not automatically useful for processing other languages.)

**EXAMPLES**

The following is an example of a *lex* program that implements a rudimentary scanner for a Pascal-like syntax:

```
%{
/* Need this for the call to atof() below. */
#include <math.h>
/* Need this for printf(), fopen(), and stdin below. */
#include <stdio.h>
}%

DIGIT      [0-9]
ID         [a-z][a-z0-9]*

%%

{DIGIT}+ {
    printf("An integer: %s (%d)\n", yytext,
        atoi(yytext));
}

{DIGIT}+"."{DIGIT}* {
    printf("A float: %s (%g)\n", yytext,
        atof(yytext));
}

if|then|begin|end|procedure|function      {
    printf("A keyword: %s\n", yytext);
}

{ID}    printf("An identifier: %s\n", yytext);

"+"|"-"|"*"|"/"      printf("An operator: %s\n", yytext);

"{ "[^]\n}"          /* Eat up one-line comments. */

[ \t\n]+             /* Eat up white space. */

.    printf("Unrecognized character: %s\n", yytext);

%%

int main(int argc, char *argv[])
{
    ++argv, --argc; /* Skip over program name. */
    if (argc > 0)
        yyin = fopen(argv[0], "r");
    else
        yyin = stdin;

    yylex();
}
```

**RATIONALE**

Even though the `-c` option and references to the C language are retained in this description, *lex* may be generalized to other languages, as was done at one time for EFL, the Extended FORTRAN Language. Since the *lex* input specification is essentially language-independent, versions of this utility could be written to produce Ada, Modula-2, or Pascal code, and there are known historical implementations that do so.



The current description of *lex* bypasses the issue of dealing with internationalized EREs in the *lex* source code or generated lexical analyzer. If it follows the model used by *awk* (the source code is assumed to be presented in the POSIX locale, but input and output are in the locale specified by the environment variables), then the tables in the lexical analyzer produced by *lex* would interpret EREs specified in the *lex* source in terms of the environment variables specified when *lex* was executed. The desired effect would be to have the lexical analyzer interpret the EREs given in the *lex* source according to the environment specified when the lexical analyzer is executed, but this is not possible with the current *lex* technology.

The description of octal and hexadecimal-digit escape sequences agrees with the ISO C standard use of escape sequences.

Earlier versions of this standard allowed for implementations with bytes other than eight bits, but this has been modified in this version.

There is no detailed output format specification. The observed behavior of *lex* under four different historical implementations was that none of these implementations consistently reported the line numbers for error and warning messages. Furthermore, there was a desire that *lex* be allowed to output additional diagnostic messages. Leaving message formats unspecified avoids these formatting questions and problems with internationalization.

Although the `%x` specifier for *exclusive* start conditions is not historical practice, it is believed to be a minor change to historical implementations and greatly enhances the usability of *lex* programs since it permits an application to obtain the expected functionality with fewer statements.

The `%array` and `%pointer` declarations were added as a compromise between historical systems. The System V-based *lex* copies the matched text to a *yytext* array. The *flex* program, supported in BSD and GNU systems, uses a pointer. In the latter case, significant performance improvements are available for some scanners. Most historical programs should require no change in porting from one system to another because the string being referenced is null-terminated in both cases. (The method used by *flex* in its case is to null-terminate the token in place by remembering the character that used to come right after the token and replacing it before continuing on to the next scan.) Multi-file programs with external references to *yytext* outside the scanner source file should continue to operate on their historical systems, but would require one of the new declarations to be considered strictly portable.

The description of EREs avoids unnecessary duplication of ERE details because their meanings within a *lex* ERE are the same as that for the ERE in this volume of POSIX.1-2008.

The reason for the undefined condition associated with text beginning with a <blank> or within "`%{`" and "`%}`" delimiter lines appearing in the *Rules* section is historical practice. Both the BSD and System V *lex* copy the indented (or enclosed) input in the *Rules* section (except at the beginning) to unreachable areas of the *yylex()* function (the code is written directly after a *break* statement). In some cases, the System V *lex* generates an error message or a syntax error, depending on the form of indented input.

The intention in breaking the list of functions into those that may appear in `lex.yy.c` *versus* those that only appear in `libl.a` is that only those functions in `libl.a` can be reliably redefined by a conforming application.

The descriptions of standard output and standard error are somewhat complicated because historical *lex* implementations chose to issue diagnostic messages to standard output (unless `-t` was given). POSIX.1-2008 allows this behavior, but leaves an opening for the more expected behavior of using standard error for diagnostics. Also, the System V behavior of writing the statistics when any table sizes are given is allowed, while BSD-derived systems can avoid it. The

programmer can always precisely obtain the desired results by using either the `-t` or `-n` options.

The OPERANDS section does not mention the use of `-` as a synonym for standard input; not all historical implementations support such usage for any of the *file* operands.

A description of the *translation table* was deleted from early proposals because of its relatively low usage in historical applications.

The change to the definition of the *input()* function that allows buffering of input presents the opportunity for major performance gains in some applications.

The following examples clarify the differences between *lex* regular expressions and regular expressions appearing elsewhere in this volume of POSIX.1-2008. For regular expressions of the form `"r/x"`, the string matching *r* is always returned; confusion may arise when the beginning of *x* matches the trailing portion of *r*. For example, given the regular expression `"a*b/cc"` and the input `"aaabcc"`, *yytext* would contain the string `"aaab"` on this match. But given the regular expression `"x*/xy"` and the input `"xxxxy"`, the token `xxx`, not `xx`, is returned by some implementations because `xxx` matches `"x*"`.

In the rule `"ab*/bc"`, the `"b*"` at the end of *r* extends *r*'s match into the beginning of the trailing context, so the result is unspecified. If this rule were `"ab/bc"`, however, the rule matches the text `"ab"` when it is followed by the text `"bc"`. In this latter case, the matching of *r* cannot extend into the beginning of *x*, so the result is specified.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*c99*, *ed*, *yacc*

XBD [Chapter 5](#) (on page 121), [Chapter 8](#) (on page 173), [Chapter 9](#) (on page 181), [Section 12.2](#) (on page 216)

#### CHANGE HISTORY

First released in Issue 2.

#### Issue 6

This utility is marked as part of the C-Language Development Utilities option.

The obsolescent `-c` option is removed.

The normative text is reworded to avoid use of the term “must” for application requirements.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/14 is applied, removing text describing behavior on systems with bytes consisting of more than eight bits.

#### Issue 7

Austin Group Interpretation 1003.1-2001 #190 is applied, clarifying the requirements for generated code to conform to the ISO C standard.

Austin Group Interpretation 1003.1-2001 #191 is applied, clarifying the handling of C-language trigraphs and curly brace preprocessing tokens.

SD5-XCU-ERN-6 is applied, clarifying that Guideline 9 of the Utility Syntax Guidelines does not apply.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



95231 **NAME**95232 link — call *link()* function95233 **SYNOPSIS**95234 XSI `link file1 file2`95235 **DESCRIPTION**95236 The *link* utility shall perform the function call:95237 `link(file1, file2);`95238 A user may need appropriate privileges to invoke the *link* utility.95239 **OPTIONS**

95240 None.

95241 **OPERANDS**

95242 The following operands shall be supported:

95243 *file1* The pathname of an existing file.95244 *file2* The pathname of the new directory entry to be created.95245 **STDIN**

95246 Not used.

95247 **INPUT FILES**

95248 Not used.

95249 **ENVIRONMENT VARIABLES**95250 The following environment variables shall affect the execution of *link*:95251 *LANG* Provide a default value for the internationalization variables that are unset or null.  
95252 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
95253 variables used to determine the values of locale categories.)95254 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
95255 internationalization variables.95256 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
95257 characters (for example, single-byte as opposed to multi-byte characters in  
95258 arguments).95259 *LC\_MESSAGES*  
95260 Determine the locale that should be used to affect the format and contents of  
95261 diagnostic messages written to standard error.95262 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.95263 **ASYNCHRONOUS EVENTS**

95264 Default.

95265 **STDOUT**

95266 None.

95267 **STDERR**

95268 The standard error shall be used only for diagnostic messages.

95269 **OUTPUT FILES**

95270 None.

95271 **EXTENDED DESCRIPTION**

95272 None.

95273 **EXIT STATUS**

95274 The following exit values shall be returned:

95275 0 Successful completion.

95276 &gt;0 An error occurred.

95277 **CONSEQUENCES OF ERRORS**

95278 Default.

95279 **APPLICATION USAGE**

95280 None.

95281 **EXAMPLES**

95282 None.

95283 **RATIONALE**

95284 None.

95285 **FUTURE DIRECTIONS**

95286 None.

95287 **SEE ALSO**95288 *ln, unlink*95289 XBD *Chapter 8* (on page 173)95290 XSH *link()*95291 **CHANGE HISTORY**

95292 First released in Issue 5.

95293 **NAME**95294 `ln` — link files95295 **SYNOPSIS**95296 `ln [-fs] [-L|-P] source_file target_file`95297 `ln [-fs] [-L|-P] source_file... target_dir`95298 **DESCRIPTION**

95299 In the first synopsis form, the `ln` utility shall create a new directory entry (link) at the destination  
 95300 path specified by the `target_file` operand. If the `-s` option is specified, a symbolic link shall be  
 95301 created for the file specified by the `source_file` operand. This first synopsis form shall be assumed  
 95302 when the final operand does not name an existing directory; if more than two operands are  
 95303 specified and the final is not an existing directory, an error shall result.

95304 In the second synopsis form, the `ln` utility shall create a new directory entry (link), or if the `-s`  
 95305 option is specified a symbolic link, for each file specified by a `source_file` operand, at a  
 95306 destination path in the existing directory named by `target_dir`.

95307 If the last operand specifies an existing file of a type not specified by the System Interfaces  
 95308 volume of POSIX.1-2008, the behavior is implementation-defined.

95309 The corresponding destination path for each `source_file` shall be the concatenation of the target  
 95310 directory pathname, a `<slash>` character if the target directory pathname did not end in a  
 95311 `<slash>`, and the last pathname component of the `source_file`. The second synopsis form shall be  
 95312 assumed when the final operand names an existing directory.

95313 For each `source_file`:

95314 1. If the destination path exists and was created by a previous step, it is unspecified whether  
 95315 `ln` shall write a diagnostic message to standard error, do nothing more with the current  
 95316 `source_file`, and go on to any remaining `source_files`; or will continue processing the current  
 95317 `source_file`. If the destination path exists:

95318 a. If the `-f` option is not specified, `ln` shall write a diagnostic message to standard  
 95319 error, do nothing more with the current `source_file`, and go on to any remaining  
 95320 `source_files`.

95321 b. If the destination path names the same directory entry as the current `source_file` `ln` |  
 95322 shall write a diagnostic message to standard error, do nothing more with the  
 95323 current `source_file`, and go on to any remaining `source_files`.

95324 c. Actions shall be performed equivalent to the `unlink()` function defined in the |  
 95325 System Interfaces volume of POSIX.1-2008, called using the destination path as the  
 95326 `path` argument. If this fails for any reason, `ln` shall write a diagnostic message to  
 95327 standard error, do nothing more with the current `source_file`, and go on to any  
 95328 remaining `source_files`.

95329 2. If the `-s` option is specified, actions shall be performed equivalent to the `symlink()`  
 95330 function with `source_file` as the `path1` argument and the destination path as the `path2`  
 95331 argument. The `ln` utility shall do nothing more with `source_file` and shall go on to any  
 95332 remaining files.

95333 3. If `source_file` is a symbolic link:

95334 a. If the `-P` option is in effect, actions shall be performed equivalent to the `linkat()`  
 95335 function with `source_file` as the `path1` argument, the destination path as the `path2`  
 95336 argument, `AT_FDCWD` as the `fd1` and `fd2` arguments, and zero as the `flag`  
 95337 argument.

- b. If the `-L` option is in effect, actions shall be performed equivalent to the `linkat()` function with *source\_file* as the *path1* argument, the destination path as the *path2* argument, `AT_FDCWD` as the *fd1* and *fd2* arguments, and `AT_SYMLINK_FOLLOW` as the *flag* argument.

The *ln* utility shall do nothing more with *source\_file* and shall go on to any remaining files.

4. Actions shall be performed equivalent to the `link()` function defined in the System Interfaces volume of POSIX.1-2008 using *source\_file* as the *path1* argument, and the destination path as the *path2* argument.

## OPTIONS

The *ln* utility shall conform to XBD [Section 12.2](#) (on page 216).

The following options shall be supported:

- `-f` Force existing destination pathnames to be removed to allow the link.
- `-L` For each *source\_file* operand that names a file of type symbolic link, create a (hard) link to the file referenced by the symbolic link.
- `-P` For each *source\_file* operand that names a file of type symbolic link, create a (hard) link to the symbolic link itself.
- `-s` Create symbolic links instead of hard links. If the `-s` option is specified, the `-L` and `-P` options shall be silently ignored.

Specifying more than one of the mutually-exclusive options `-L` and `-P` shall not be considered an error. The last option specified shall determine the behavior of the utility (unless the `-s` option causes it to be ignored).

If the `-s` option is not specified and neither a `-L` nor a `-P` option is specified, it is implementation-defined which of the `-L` and `-P` options will be used as the default.

## OPERANDS

The following operands shall be supported:

- source\_file* A pathname of a file to be linked. If the `-s` option is specified, no restrictions on the type of file or on its existence shall be made. If the `-s` option is not specified, whether a directory can be linked is implementation-defined.
- target\_file* The pathname of the new directory entry to be created.
- target\_dir* A pathname of an existing directory in which the new directory entries are created.

## STDIN

Not used.

## INPUT FILES

None.

## ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of *ln*:

- LANG* Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)
- LC\_ALL* If set to a non-empty string value, override the values of all the other internationalization variables.

|       |                               |                                                                                                                                                                                                                                                                                                                                                   |
|-------|-------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 95379 | <b>LC_CTYPE</b>               | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).                                                                                                                                                                         |
| 95380 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 95381 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 95382 | <b>LC_MESSAGES</b>            |                                                                                                                                                                                                                                                                                                                                                   |
| 95383 |                               | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                                                                                                                      |
| 95384 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 95385 | XSI <b>NLSPATH</b>            | Determine the location of message catalogs for the processing of <b>LC_MESSAGES</b> .                                                                                                                                                                                                                                                             |
| 95386 | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                                                                                                                                                                                                                   |
| 95387 |                               | Default.                                                                                                                                                                                                                                                                                                                                          |
| 95388 | <b>STDOUT</b>                 |                                                                                                                                                                                                                                                                                                                                                   |
| 95389 |                               | Not used.                                                                                                                                                                                                                                                                                                                                         |
| 95390 | <b>STDERR</b>                 |                                                                                                                                                                                                                                                                                                                                                   |
| 95391 |                               | The standard error shall be used only for diagnostic messages.                                                                                                                                                                                                                                                                                    |
| 95392 | <b>OUTPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                   |
| 95393 |                               | None.                                                                                                                                                                                                                                                                                                                                             |
| 95394 | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                                                                                                                                                                                                                   |
| 95395 |                               | None.                                                                                                                                                                                                                                                                                                                                             |
| 95396 | <b>EXIT STATUS</b>            |                                                                                                                                                                                                                                                                                                                                                   |
| 95397 |                               | The following exit values shall be returned:                                                                                                                                                                                                                                                                                                      |
| 95398 | 0                             | All the specified files were linked successfully.                                                                                                                                                                                                                                                                                                 |
| 95399 | >0                            | An error occurred.                                                                                                                                                                                                                                                                                                                                |
| 95400 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                                                                                                                                                                                                                   |
| 95401 |                               | Default.                                                                                                                                                                                                                                                                                                                                          |
| 95402 | <b>APPLICATION USAGE</b>      |                                                                                                                                                                                                                                                                                                                                                   |
| 95403 |                               | None.                                                                                                                                                                                                                                                                                                                                             |
| 95404 | <b>EXAMPLES</b>               |                                                                                                                                                                                                                                                                                                                                                   |
| 95405 |                               | None.                                                                                                                                                                                                                                                                                                                                             |
| 95406 | <b>RATIONALE</b>              |                                                                                                                                                                                                                                                                                                                                                   |
| 95407 |                               | The <b>CONSEQUENCES OF ERRORS</b> section does not require <i>ln -f a b</i> to remove <i>b</i> if a subsequent link operation would fail.                                                                                                                                                                                                         |
| 95408 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 95409 |                               | Some historic versions of <i>ln</i> (including the one specified by the SVID) unlink the destination file, if it exists, by default. If the mode does not permit writing, these versions prompt for confirmation before attempting the unlink. In these versions the <i>-f</i> option causes <i>ln</i> not to attempt to prompt for confirmation. |
| 95410 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 95411 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 95412 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 95413 |                               | This allows <i>ln</i> to succeed in creating links when the target file already exists, even if the file itself is not writable (although the directory must be). Early proposals specified this functionality.                                                                                                                                   |
| 95414 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 95415 |                               | This volume of POSIX.1-2008 does not allow the <i>ln</i> utility to unlink existing destination paths by default for the following reasons:                                                                                                                                                                                                       |
| 95416 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 95417 |                               | • The <i>ln</i> utility has historically been used to provide locking for shell applications, a usage that is incompatible with <i>ln</i> unlinking the destination path by default. There was no corresponding technical advantage to adding this functionality.                                                                                 |
| 95418 |                               |                                                                                                                                                                                                                                                                                                                                                   |
| 95419 |                               |                                                                                                                                                                                                                                                                                                                                                   |

- This functionality gave *ln* the ability to destroy the link structure of files, which changes the historical behavior of *ln*.
- This functionality is easily replicated with a combination of *rm* and *ln*.
- It is not historical practice in many systems; BSD and BSD-derived systems do not support this behavior. Unfortunately, whichever behavior is selected can cause scripts written expecting the other behavior to fail.
- It is preferable that *ln* perform in the same manner as the *link()* function, which does not permit the target to exist already.

This volume of POSIX.1-2008 retains the *-f* option to provide support for shell scripts depending on the SVID semantics. It seems likely that shell scripts would not be written to handle prompting by *ln* and would therefore have specified the *-f* option.

The *-f* option is an undocumented feature of many historical versions of the *ln* utility, allowing linking to directories. These versions require modification.

Early proposals of this volume of POSIX.1-2008 also required a *-i* option, which behaved like the *-i* options in *cp* and *mv*, prompting for confirmation before unlinking existing files. This was not historical practice for the *ln* utility and has been omitted.

The *-L* and *-P* options allow for implementing both common behaviors of the *ln* utility. Earlier versions of this standard did not specify these options and required the behavior now described for the *-L* option. Many systems by default or as an alternative provided a non-conforming *ln* utility with the behavior now described for the *-P* option. Since applications could not rely on *ln* following links in practice, the *-L* and *-P* options were added to specify the desired behavior for the application.

The *-L* and *-P* options are ignored when *-s* is specified in order to allow an alias to be created to alter the default behavior when creating hard links (for example, *alias ln='ln -L'*). They serve no purpose when *-s* is specified, since *source\_file* is then just a string to be used as the contents of the created symbolic link and need not exist as a file.

The specification ensures that *ln a a* with or without the *-f* option will not unlink the file *a*. Earlier versions of this standard were unclear in this case.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*chmod*, *find*, *pax*, *rm*

XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

XSH *link()*, *unlink()*

## CHANGE HISTORY

First released in Issue 2.

### Issue 6

The *ln* utility is updated to include symbolic link processing as defined in the IEEE P1003.2b draft standard.

### Issue 7

Austin Group Interpretations 1003.1-2001 #164, #168, and #169 are applied.

SD5-XCU-ERN-27 is applied, adding a new paragraph to the RATIONALE.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

95463  
95464  
95465

The **-L** and **-P** options are added to make it implementation-defined whether the *ln* utility follows symbolic links.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0096 [136] is applied.

95466 **NAME**

95467 locale — get locale-specific information

95468 **SYNOPSIS**

95469 locale [-a|-m]

95470 locale [-ck] name...

95471 **DESCRIPTION**

95472 The *locale* utility shall write information about the current locale environment, or all public  
 95473 locales, to the standard output. For the purposes of this section, a *public locale* is one provided by  
 95474 the implementation that is accessible to the application.

95475 When *locale* is invoked without any arguments, it shall summarize the current locale  
 95476 environment for each locale category as determined by the settings of the environment variables  
 95477 defined in XBD [Chapter 7](#) (on page 135).

95478 When invoked with operands, it shall write values that have been assigned to the keywords in  
 95479 the locale categories, as follows:

- 95480 • Specifying a keyword name shall select the named keyword and the category containing  
 95481 that keyword.
- 95482 • Specifying a category name shall select the named category and all keywords in that  
 95483 category.

95484 **OPTIONS**95485 The *locale* utility shall conform to XBD [Section 12.2](#) (on page 216).

95486 The following options shall be supported:

- 95487 **-a** Write information about all available public locales. The available locales shall  
 95488 include **POSIX**, representing the POSIX locale. The manner in which the  
 95489 implementation determines what other locales are available is implementation-  
 95490 defined.
- 95491 **-c** Write the names of selected locale categories; see the STDOUT section. The **-c**  
 95492 option increases readability when more than one category is selected (for example,  
 95493 via more than one keyword name or via a category name). It is valid both with  
 95494 and without the **-k** option.
- 95495 **-k** Write the names and values of selected keywords. The implementation may omit  
 95496 values for some keywords; see the OPERANDS section.
- 95497 **-m** Write names of available charmaps; see XBD [Section 6.1](#) (on page 125).

95498 **OPERANDS**

95499 The following operand shall be supported:

- 95500 **name** The name of a locale category as defined in XBD [Chapter 7](#) (on page 135), the name  
 95501 of a keyword in a locale category, or the reserved name **charmap**. The named  
 95502 category or keyword shall be selected for output. If a single *name* represents both a  
 95503 locale category name and a keyword name in the current locale, the results are  
 95504 unspecified. Otherwise, both category and keyword names can be specified as  
 95505 *name* operands, in any sequence. It is implementation-defined whether any  
 95506 keyword values are written for the categories *LC\_CTYPE* and *LC\_COLLATE*.



95507 **STDIN**

95508 Not used.

95509 **INPUT FILES**

95510 None.

95511 **ENVIRONMENT VARIABLES**95512 The following environment variables shall affect the execution of *locale*:

95513 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 95514 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 95515 variables used to determine the values of locale categories.)

95516 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 95517 internationalization variables.

95518 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 95519 characters (for example, single-byte as opposed to multi-byte characters in  
 95520 arguments and input files).

95521 *LC\_MESSAGES*

95522 Determine the locale that should be used to affect the format and contents of  
 95523 diagnostic messages written to standard error.

95524 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

95525 XSI The application shall ensure that the *LANG*, *LC\_\**, and *NLSPATH* environment variables specify  
 95526 the current locale environment to be written out; they shall be used if the *-a* option is not  
 95527 specified.

95528 **ASYNCHRONOUS EVENTS**

95529 Default.

95530 **STDOUT**95531 The *LANG* variable shall be written first using the format:

95532 "LANG=%s\n", &lt;value&gt;

95533 If *LANG* is not set or is an empty string, the value is the empty string.

95534 If *locale* is invoked without any options or operands, the names and values of the *LC\_\**  
 95535 environment variables described in this volume of POSIX.1-2008 shall be written to the standard  
 95536 output, one variable per line, and each line using the following format. Only those variables set  
 95537 in the environment and not overridden by *LC\_ALL* shall be written using this format:

95538 "%s=%s\n", &lt;variable\_name&gt;, &lt;value&gt;

95539 The names of those *LC\_\** variables associated with locale categories defined in this volume of  
 95540 POSIX.1-2008 that are not set in the environment or are overridden by *LC\_ALL* shall be written  
 95541 in the following format:

95542 "%s=\"%s\"\\n", &lt;variable\_name&gt;, &lt;implied value&gt;

95543 The <implied value> shall be the name of the locale that has been selected for that category by the  
 95544 implementation, based on the values in *LANG* and *LC\_ALL*, as described in XBD [Chapter 8](#) (on  
 95545 page 173).

95546 The <value> and <implied value> shown above shall be properly quoted for possible later reentry  
 95547 to the shell. The <value> shall not be quoted using double-quotes (so that it can be distinguished  
 95548 by the user from the <implied value> case, which always requires double-quotes).

The `LC_ALL` variable shall be written last, using the first format shown above. If it is not set, it shall be written as:

```
"LC_ALL=\n"
```

If any arguments are specified:

1. If the `-a` option is specified, the names of all the public locales shall be written, each in the following format:

```
"%s\n", <locale name>
```

2. If the `-c` option is specified, the names of all selected categories shall be written, each in the following format:

```
"%s\n", <category name>
```

If keywords are also selected for writing (see following items), the category name output shall precede the keyword output for that category.

If the `-c` option is not specified, the names of the categories shall not be written; only the keywords, as selected by the `<name>` operand, shall be written.

3. If the `-k` option is specified, the names and values of selected keywords shall be written. If a value is non-numeric and is not a compound keyword value, it shall be written in the following format:

```
"%s=\"%s\"\n", <keyword name>, <keyword value>
```

If a value is a non-numeric compound keyword value, it shall either be written in the format:

```
"%s=\"%s\"\n", <keyword name>, <keyword value>
```

where the `<keyword value>` is a single string of values separated by `<semicolon>` characters, or it shall be written in the format:

```
"%s=%s\n", <keyword name>, <keyword value>
```

where the `<keyword value>` is encoded as a set of strings, each enclosed in double-quotation-marks, separated by `<semicolon>` characters.

If the keyword was **charmap**, the name of the charmap (if any) that was specified via the `localedef -f` option when the locale was created shall be written, with the word **charmap** as `<keyword name>`.

If a value is numeric, it shall be written in one of the following formats:

```
"%s=%d\n", <keyword name>, <keyword value>
```

```
"%s=%c%o\n", <keyword name>, <escape character>, <keyword value>
```

```
"%s=%cx%x\n", <keyword name>, <escape character>, <keyword value>
```

where the `<escape character>` is that identified by the **escape\_char** keyword in the current locale; see XBD [Section 7.3](#) (on page 136).

Compound keyword values (list entries) shall be separated in the output by `<semicolon>` characters. When included in keyword values, the `<semicolon>`, `<backslash>`, double-quote, and any control character shall be preceded (escaped) with the escape character.

95587 4. If the **-k** option is not specified, selected keyword values shall be written, each in the  
 95588 following format:

95589 "%s\n", <keyword value>

95590 If the keyword was **charmap**, the name of the charmap (if any) that was specified via the  
 95591 *localedef* **-f** option when the locale was created shall be written.

95592 5. If the **-m** option is specified, then a list of all available charmaps shall be written, each in  
 95593 the format:

95594 "%s\n", <charmap>

95595 where <charmap> is in a format suitable for use as the option-argument to the *localedef* **-f**  
 95596 option.

#### 95597 **STDERR**

95598 The standard error shall be used only for diagnostic messages.

#### 95599 **OUTPUT FILES**

95600 None.

#### 95601 **EXTENDED DESCRIPTION**

95602 None.

#### 95603 **EXIT STATUS**

95604 The following exit values shall be returned:

95605 0 All the requested information was found and output successfully.

95606 >0 An error occurred.

#### 95607 **CONSEQUENCES OF ERRORS**

95608 Default.

#### 95609 **APPLICATION USAGE**

95610 If the *LANG* environment variable is not set or set to an empty value, or one of the *LC\_\**  
 95611 environment variables is set to an unrecognized value, the actual locales assumed (if any) are  
 95612 implementation-defined as described in XBD [Chapter 8](#) (on page 173).

95613 Implementations are not required to write out the actual values for keywords in the categories  
 95614 *LC\_CTYPE* and *LC\_COLLATE*; however, they must write out the categories (allowing an  
 95615 application to determine, for example, which character classes are available).

#### 95616 **EXAMPLES**

95617 In the following examples, the assumption is that locale environment variables are set as  
 95618 follows:

95619 *LANG*=locale\_x

95620 *LC\_COLLATE*=locale\_y

95621 The command *locale* would result in the following output:

95622 *LANG*=locale\_x

95623 *LC\_CTYPE*="locale\_x"

95624 *LC\_COLLATE*=locale\_y

95625 *LC\_TIME*="locale\_x"

95626 *LC\_NUMERIC*="locale\_x"

95627 *LC\_MONETARY*="locale\_x"

95628 *LC\_MESSAGES*="locale\_x"

95629 *LC\_ALL*=

The order of presentation of the categories is not specified by this volume of POSIX.1-2008.

The command:

```
LC_ALL=POSIX locale -ck decimal_point
```

would produce:

```
LC_NUMERIC
```

```
decimal_point="."
```

The following command shows an application of *locale* to determine whether a user-supplied response is affirmative:

```
printf 'Prompt for response: '
read response
if printf "%s\n" "$response" | grep -- -Eq "${locale yesexpr}"
then
    affirmative processing goes here
else
    non-affirmative processing goes here
fi
```

## RATIONALE

The output for categories *LC\_CTYPE* and *LC\_COLLATE* has been made implementation-defined because there is a questionable value in having a shell script receive an entire array of characters. It is also difficult to return a logical collation description, short of returning a complete *localedef* source.

The **-m** option was included to allow applications to query for the existence of charmaps. The output is a list of the charmaps (implementation-supplied and user-supplied, if any) on the system.

The **-c** option was included for readability when more than one category is selected (for example, via more than one keyword name or via a category name). It is valid both with and without the **-k** option.

The **charmap** keyword, which returns the name of the charmap (if any) that was used when the current locale was created, was included to allow applications needing the information to retrieve it.

According to XBD [Section 6.1](#) (on page 125), the standard requires that all supported locales must have the same encoding for <period> and <slash>, because these two characters are used within the locale-independent pathname resolution sequence. Therefore, it would be an error if *locale -a* listed both ASCII and EBCDIC-based locales, since those two encodings do not share the same representation for either <period> or <slash>. Any system that supports both environments would be expected to provide two POSIX locales, one in either codeset, where only the locales appropriate to the current environment can be visible at a time. In an XSI-compliant implementation, the *dd* utility is the only portable means for performing conversions between the two character sets.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*localedef*

XBD [Section 6.1](#) (on page 125), [Chapter 7](#) (on page 135), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

95675 **CHANGE HISTORY**

95676 First released in Issue 4.

95677 **Issue 5**

95678 The FUTURE DIRECTIONS section is added.

95679 **Issue 6**

95680 The normative text is reworded to avoid use of the term “must” for application requirements.

95681 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/30 is applied, correcting an editorial error  
95682 in the STDOUT section.95683 **Issue 7**95684 Austin Group Interpretations 1003.1-2001 #017, #021, and #088 are applied, clarifying the  
95685 standard output for the **-k** option when *LANG* is not set or is an empty string.

95686 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

95687 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0097 [291] is applied.

95688 **NAME**

95689 localedef — define locale environment

95690 **SYNOPSIS**95691 localedef [-c] [-f *charmap*] [-i *sourcefile*] [-u *code\_set\_name*] *name*95692 **DESCRIPTION**

95693 The *localedef* utility shall convert source definitions for locale categories into a format usable by  
 95694 the functions and utilities whose operational behavior is determined by the setting of the locale  
 95695 environment variables defined in XBD [Chapter 7](#) (on page 135). It is implementation-defined  
 95696 whether users have the capability to create new locales, in addition to those supplied by the  
 95697 implementation. If the symbolic constant POSIX2\_LOCALEDEF is defined, the system supports  
 95698 XSI the creation of new locales. On XSI-conformant systems, the symbolic constant  
 95699 POSIX2\_LOCALEDEF shall be defined.

95700 The utility shall read source definitions for one or more locale categories belonging to the same  
 95701 locale from the file named in the -i option (if specified) or from standard input.

95702 The *name* operand identifies the target locale. The utility shall support the creation of *public*, or  
 95703 generally accessible locales, as well as *private*, or restricted-access locales. Implementations may  
 95704 restrict the capability to create or modify public locales to users with appropriate privileges.

95705 Each category source definition shall be identified by the corresponding environment variable  
 95706 name and terminated by an **END** *category-name* statement. The following categories shall be  
 95707 supported. In addition, the input may contain source for implementation-defined categories.

95708 *LC\_CTYPE* Defines character classification and case conversion.

95709 *LC\_COLLATE*  
 95710 Defines collation rules.

95711 *LC\_MONETARY*  
 95712 Defines the format and symbols used in formatting of monetary information.

95713 *LC\_NUMERIC*  
 95714 Defines the decimal delimiter, grouping, and grouping symbol for non-monetary  
 95715 numeric editing.

95716 *LC\_TIME* Defines the format and content of date and time information.

95717 *LC\_MESSAGES*  
 95718 Defines the format and values of affirmative and negative responses.

95719 **OPTIONS**

95720 The *localedef* utility shall conform to XBD [Section 12.2](#) (on page 216).

95721 The following options shall be supported:

95722 **-c** Create permanent output even if warning messages have been issued.

95723 **-f *charmap*** Specify the pathname of a file containing a mapping of character symbols and  
 95724 collating element symbols to actual character encodings. The format of the  
 95725 *charmap* is described in XBD [Section 6.4](#) (on page 129). The application shall ensure  
 95726 that this option is specified if symbolic names (other than collating symbols  
 95727 defined in a **collating-symbol** keyword) are used. If the -f option is not present, an  
 95728 implementation-defined character mapping shall be used.

95729 **-i *inputfile*** The pathname of a file containing the source definitions. If this option is not  
 95730 present, source definitions shall be read from standard input. The format of the  
 95731 *inputfile* is described in XBD [Section 7.3](#) (on page 136).

95732            **-u** *code\_set\_name*  
 95733                    Specify the name of a codeset used as the target mapping of character symbols and  
 95734                    collating element symbols whose encoding values are defined in terms of the  
 95735                    ISO/IEC 10646-1:2000 standard position constant values.

## 95736 OPERANDS

95737            The following operand shall be supported:

95738            *name*            Identifies the locale; see XBD [Chapter 7](#) (on page 135) for a description of the use of  
 95739                    this name. If the name contains one or more <slash> characters, *name* shall be  
 95740                    interpreted as a pathname where the created locale definitions shall be stored. If  
 95741                    *name* does not contain any <slash> characters, the interpretation of the name is  
 95742                    implementation-defined and the locale shall be public. The ability to create public  
 95743                    locales in this way may be restricted to users with appropriate privileges. (As a  
 95744                    consequence of specifying one *name*, although several categories can be processed  
 95745                    in one execution, only categories belonging to the same locale can be processed.)

## 95746 STDIN

95747            Unless the **-i** option is specified, the standard input shall be a text file containing one or more  
 95748                    locale category source definitions, as described in XBD [Section 7.3](#) (on page 136). When lines are  
 95749                    continued using the escape character mechanism, there is no limit to the length of the  
 95750                    accumulated continued line.

## 95751 INPUT FILES

95752            The character set mapping file specified as the *charmap* option-argument is described in XBD  
 95753                    [Section 6.4](#) (on page 129). If a locale category source definition contains a **copy** statement, as  
 95754                    defined in XBD [Chapter 7](#) (on page 135), and the **copy** statement names a valid, existing locale,  
 95755                    then *localedef* shall behave as if the source definition had contained a valid category source  
 95756                    definition for the named locale.

## 95757 ENVIRONMENT VARIABLES

95758            The following environment variables shall affect the execution of *localedef*:

95759            *LANG*            Provide a default value for the internationalization variables that are unset or null.  
 95760                    (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 95761                    variables used to determine the values of locale categories.)

95762            *LC\_ALL*            If set to a non-empty string value, override the values of all the other  
 95763                    internationalization variables.

95764            *LC\_COLLATE*  
 95765                    (This variable has no affect on *localedef*; the POSIX locale is used for this category.)

95766            *LC\_CTYPE*           Determine the locale for the interpretation of sequences of bytes of text data as  
 95767                    characters (for example, single-byte as opposed to multi-byte characters in  
 95768                    arguments and input files). This variable has no affect on the processing of *localedef*  
 95769                    input data; the POSIX locale is used for this purpose, regardless of the value of this  
 95770                    variable.

95771            *LC\_MESSAGES*  
 95772                    Determine the locale that should be used to affect the format and contents of  
 95773                    diagnostic messages written to standard error.

95774    XSI            *NLSPATH*           Determine the location of message catalogs for the processing of *LC\_MESSAGES*.



95775 **ASYNCHRONOUS EVENTS**

95776 Default.

95777 **STDOUT**

95778 The utility shall report all categories successfully processed, in an unspecified format.

95779 **STDERR**

95780 The standard error shall be used only for diagnostic messages.

95781 **OUTPUT FILES**95782 The format of the created output is unspecified. If the *name* operand does not contain a <slash>,  
95783 the existence of an output file for the locale is unspecified.95784 **EXTENDED DESCRIPTION**95785 When the **-u** option is used, the *code\_set\_name* option-argument shall be interpreted as an  
95786 implementation-defined name of a codeset to which the ISO/IEC 10646-1:2000 standard  
95787 position constant values shall be converted via an implementation-defined method. Both the  
95788 ISO/IEC 10646-1:2000 standard position constant values and other formats (decimal,  
95789 hexadecimal, or octal) shall be valid as encoding values within the *charmap* file. The codeset  
95790 represented by the implementation-defined name can be any codeset that is supported by the  
95791 implementation.95792 When conflicts occur between the *charmap* specification of <*code\_set\_name*>, <*mb\_cur\_max*>, or  
95793 <*mb\_cur\_min*> and the implementation-defined interpretation of these respective items for the  
95794 codeset represented by the **-u** option-argument *code\_set\_name*, the result is unspecified.95795 When conflicts occur between the *charmap* encoding values specified for symbolic names of  
95796 characters of the portable character set and the implementation-defined assignment of character  
95797 encoding values, the result is unspecified.95798 If a non-printable character in the *charmap* has a width specified that is not **-1**, the result will be  
95799 undefined.95800 **EXIT STATUS**

95801 The following exit values shall be returned:

- 95802 0 No errors occurred and the locales were successfully created.
- 
- 95803 1 Warnings occurred and the locales were successfully created.
- 
- 95804 2 The locale specification exceeded implementation limits or the coded character set or sets
- 
- 95805 used were not supported by the implementation, and no locale was created.
- 
- 95806 3 The capability to create new locales is not supported by the implementation.
- 
- 95807 >3 Warnings or errors occurred and no output was created.

95808 **CONSEQUENCES OF ERRORS**

95809 If an error is detected, no permanent output shall be created.

95810 If warnings occur, permanent output shall be created if the **-c** option was specified. The  
95811 following conditions shall cause warning messages to be issued:

- 95812 • If a symbolic name not found in the
- charmap*
- file is used for the descriptions of the
- 
- 95813
- LC\_CTYPE*
- or
- LC\_COLLATE*
- categories (for other categories, this shall be an error
- 
- 95814 condition).
- 
- 95815 • If the number of operands to the
- order**
- keyword exceeds the {
- COLL\_WEIGHTS\_MAX*
- }
- 
- 95816 limit.



- If optional keywords not supported by the implementation are present in the source.

Other implementation-defined conditions may also cause warnings.

## APPLICATION USAGE

The *charmap* definition is optional, and is contained outside the locale definition. This allows both completely self-defined source files, and generic sources (applicable to more than one codeset). To aid portability, all *charmap* definitions must use the same symbolic names for the portable character set. As explained in XBD [Section 6.4](#) (on page 129), it is implementation-defined whether or not users or applications can provide additional character set description files. Therefore, the `-f` option might be operable only when an implementation-defined *charmap* is named.

## EXAMPLES

None.

## RATIONALE

The output produced by the *localedef* utility is implementation-defined. The *name* operand is used to identify the specific locale. (As a consequence, although several categories can be processed in one execution, only categories belonging to the same locale can be processed.)

## FUTURE DIRECTIONS

None.

## SEE ALSO

[locale](#)

XBD [Section 6.4](#) (on page 129), [Chapter 7](#) (on page 135), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

## CHANGE HISTORY

First released in Issue 4.

### Issue 6

The `-u` option is added, as specified in the IEEE P1003.2b draft standard.

The normative text is reworded to avoid use of the term “must” for application requirements.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/15 is applied, rewording text in the OPERANDS section describing the ability to create public locales.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/16 is applied, making the text consistent with the descriptions of **WIDTH** and **WIDTH\_DEFAULT** in the Base Definitions volume of POSIX.1-2008.

### Issue 7

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

95851 **NAME**

95852       logger — log messages

95853 **SYNOPSIS**95854       logger *string*...95855 **DESCRIPTION**

95856       The *logger* utility saves a message, in an unspecified manner and format, containing the *string*  
 95857       operands provided by the user. The messages are expected to be evaluated later by personnel  
 95858       performing system administration tasks.

95859       It is implementation-defined whether messages written in locales other than the POSIX locale  
 95860       are effective.

95861 **OPTIONS**

95862       None.

95863 **OPERANDS**

95864       The following operand shall be supported:

95865       *string*       One of the string arguments whose contents are concatenated together, in the order  
 95866       specified, separated by single <space> characters.

95867 **STDIN**

95868       Not used.

95869 **INPUT FILES**

95870       None.

95871 **ENVIRONMENT VARIABLES**95872       The following environment variables shall affect the execution of *logger*:

95873       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 95874       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 95875       variables used to determine the values of locale categories.)

95876       *LC\_ALL*     If set to a non-empty string value, override the values of all the other  
 95877       internationalization variables.

95878       *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 95879       characters (for example, single-byte as opposed to multi-byte characters in  
 95880       arguments).

95881       *LC\_MESSAGES*

95882       Determine the locale that should be used to affect the format and contents of  
 95883       diagnostic messages written to standard error. (This means diagnostics from *logger*  
 95884       to the user or application, not diagnostic messages that the user is sending to the  
 95885       system administrator.)

95886       *XSI*       *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

95887 **ASYNCHRONOUS EVENTS**

95888       Default.

95889 **STDOUT**

95890       Not used.

**95891 STDERR**

95892 The standard error shall be used only for diagnostic messages.

**95893 OUTPUT FILES**

95894 Unspecified.

**95895 EXTENDED DESCRIPTION**

95896 None.

**95897 EXIT STATUS**

95898 The following exit values shall be returned:

95899 0 Successful completion.

95900 >0 An error occurred.

**95901 CONSEQUENCES OF ERRORS**

95902 Default.

**95903 APPLICATION USAGE**

95904 This utility allows logging of information for later use by a system administrator or programmer  
95905 in determining why non-interactive utilities have failed. The locations of the saved messages,  
95906 their format, and retention period are all unspecified. There is no method for a conforming  
95907 application to read messages, once written.

**95908 EXAMPLES**

95909 A batch application, running non-interactively, tries to read a configuration file and fails; it may  
95910 attempt to notify the system administrator with:

95911 `logger myname: unable to read file foo. [timestamp]`

**95912 RATIONALE**

95913 The standard developers believed strongly that some method of alerting administrators to errors  
95914 was necessary. The obvious example is a batch utility, running non-interactively, that is unable to  
95915 read its configuration files or that is unable to create or write its results file. However, the  
95916 standard developers did not wish to define the format or delivery mechanisms as they have  
95917 historically been (and will probably continue to be) very system-specific, as well as involving  
95918 functionality clearly outside the scope of this volume of POSIX.1-2008.

95919 The text with *LC\_MESSAGES* about diagnostic messages means diagnostics from *logger* to the  
95920 user or application, not diagnostic messages that the user is sending to the system administrator.

95921 Multiple *string* arguments are allowed, similar to *echo*, for ease-of-use.

95922 Like the utilities *mailx* and *lp*, *logger* is admittedly difficult to test. This was not deemed sufficient  
95923 justification to exclude these utilities from this volume of POSIX.1-2008. It is also arguable that  
95924 they are, in fact, testable, but that the tests themselves are not portable.

**95925 FUTURE DIRECTIONS**

95926 None.

**95927 SEE ALSO**

95928 *lp*, *mailx*, *write*

95929 XBD Chapter 8 (on page 173)

**95930 CHANGE HISTORY**

95931 First released in Issue 4.

95932 **Issue 7**  
95933

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

95934 **NAME**

95935 logname — return the user's login name

95936 **SYNOPSIS**

95937 logname

95938 **DESCRIPTION**

95939 The *logname* utility shall write the user's login name to standard output. The login name shall be  
 95940 the string that would be returned by the *getlogin()* function defined in the System Interfaces  
 95941 volume of POSIX.1-2008. Under the conditions where the *getlogin()* function would fail, the  
 95942 *logname* utility shall write a diagnostic message to standard error and exit with a non-zero exit  
 95943 status.

95944 **OPTIONS**

95945 None.

95946 **OPERANDS**

95947 None.

95948 **STDIN**

95949 Not used.

95950 **INPUT FILES**

95951 None.

95952 **ENVIRONMENT VARIABLES**95953 The following environment variables shall affect the execution of *logname*:

95954 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 95955 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 95956 variables used to determine the values of locale categories.)

95957 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 95958 internationalization variables.

95959 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 95960 characters (for example, single-byte as opposed to multi-byte characters in  
 95961 arguments).

95962 *LC\_MESSAGES*

95963 Determine the locale that should be used to affect the format and contents of  
 95964 diagnostic messages written to standard error.

95965 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

95966 **ASYNCHRONOUS EVENTS**

95967 Default.

95968 **STDOUT**95969 The *logname* utility output shall be a single line consisting of the user's login name:

95970 "%s\n", &lt;login name&gt;

95971 **STDERR**

95972 The standard error shall be used only for diagnostic messages.

95973 **OUTPUT FILES**

95974 None.

95975 **EXTENDED DESCRIPTION**

95976 None.

95977 **EXIT STATUS**

95978 The following exit values shall be returned:

95979 0 Successful completion.

95980 &gt;0 An error occurred.

95981 **CONSEQUENCES OF ERRORS**

95982 Default.

95983 **APPLICATION USAGE**95984 The *logname* utility explicitly ignores the *LOGNAME* environment variable because environment  
95985 changes could produce erroneous results.95986 **EXAMPLES**

95987 None.

95988 **RATIONALE**95989 The **passwd** file is not listed as required because the implementation may have other means of  
95990 mapping login names.95991 **FUTURE DIRECTIONS**

95992 None.

95993 **SEE ALSO**95994 *id*, *who*

95995 XBD Chapter 8 (on page 173)

95996 XSH *getlogin()*95997 **CHANGE HISTORY**

95998 First released in Issue 2.

95999 **NAME**96000 `lp` — send files to a printer96001 **SYNOPSIS**96002 `lp [-c] [-d dest] [-n copies] [-msw] [-o option]... [-t title] [file...]`96003 **DESCRIPTION**

96004 The *lp* utility shall copy the input files to an output destination in an unspecified manner. The  
 96005 default output destination should be to a hardcopy device, such as a printer or microfilm  
 96006 recorder, that produces non-volatile, human-readable documents. If such a device is not  
 96007 available to the application, or if the system provides no such device, the *lp* utility shall exit with  
 96008 a non-zero exit status.

96009 The actual writing to the output device may occur some time after the *lp* utility successfully  
 96010 exits. During the portion of the writing that corresponds to each input file, the implementation  
 96011 shall guarantee exclusive access to the device.

96012 The *lp* utility shall associate a unique *request ID* with each request.

96013 Normally, a banner page is produced to separate and identify each print job. This page may be  
 96014 suppressed by implementation-defined conditions, such as an operator command or one of the  
 96015 `-o option` values.

96016 **OPTIONS**96017 The *lp* utility shall conform to XBD [Section 12.2](#) (on page 216).

96018 The following options shall be supported:

96019 **-c** Exit only after further access to any of the input files is no longer required. The  
 96020 application can then safely delete or modify the files without affecting the output  
 96021 operation. Normally, files are not copied, but are linked whenever possible. If the  
 96022 `-c` option is not given, then the user should be careful not to remove any of the  
 96023 files before the request has been printed in its entirety. It should also be noted that  
 96024 in the absence of the `-c` option, any changes made to the named files after the  
 96025 request is made but before it is printed may be reflected in the printed output. On  
 96026 some implementations, `-c` may be on by default.

96027 **-d *dest*** Specify a string that names the destination (*dest*). If *dest* is a printer, the request  
 96028 shall be printed only on that specific printer. If *dest* is a class of printers, the request  
 96029 shall be printed on the first available printer that is a member of the class. Under  
 96030 certain conditions (printer unavailability, file space limitation, and so on), requests  
 96031 for specific destinations need not be accepted. Destination names vary between  
 96032 systems.

96033 If `-d` is not specified, and neither the *LPDEST* nor *PRINTER* environment variable  
 96034 is set, an unspecified destination is used. The `-d dest` option shall take precedence  
 96035 over *LPDEST*, which in turn shall take precedence over *PRINTER*. Results are  
 96036 undefined when *dest* contains a value that is not a valid destination name.

96037 **-m** Send mail (see [mailx](#)) after the files have been printed. By default, no mail is sent  
 96038 upon normal completion of the print request.

96039 **-n *copies*** Write *copies* number of copies of the files, where *copies* is a positive decimal integer.  
 96040 The methods for producing multiple copies and for arranging the multiple copies  
 96041 when multiple *file* operands are used are unspecified, except that each file shall be  
 96042 output as an integral whole, not interleaved with portions of other files.

- 96043        **-o option**        Specify printer-dependent or class-dependent *options*. Several such *options* may be  
96044                                collected by specifying the **-o** option more than once.
- 96045        **-s**                Suppress messages from *lp*.
- 96046        **-t title**            Write *title* on the banner page of the output.
- 96047        **-w**                Write a message on the user's terminal after the files have been printed. If the user  
96048                                is not logged in, then mail shall be sent instead.

## 96049 OPERANDS

96050        The following operand shall be supported:

- 96051        *file*             A pathname of a file to be output. If no *file* operands are specified, or if a *file*  
96052                                operand is '-', the standard input shall be used. If a *file* operand is used, but the  
96053                                **-c** option is not specified, the process performing the writing to the output device  
96054                                may have user and group permissions that differ from that of the process invoking  
96055                                *lp*.

## 96056 STDIN

96057        The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '-'.  
96058        See the INPUT FILES section.

## 96059 INPUT FILES

96060        The input files shall be text files.

## 96061 ENVIRONMENT VARIABLES

96062        The following environment variables shall affect the execution of *lp*:

- 96063        *LANG*             Provide a default value for the internationalization variables that are unset or null.  
96064                                (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
96065                                variables used to determine the values of locale categories.)
- 96066        *LC\_ALL*            If set to a non-empty string value, override the values of all the other  
96067                                internationalization variables.
- 96068        *LC\_CTYPE*        Determine the locale for the interpretation of sequences of bytes of text data as  
96069                                characters (for example, single-byte as opposed to multi-byte characters in  
96070                                arguments and input files).
- 96071        *LC\_MESSAGES*     Determine the locale that should be used to affect the format and contents of  
96072                                diagnostic messages written to standard error and informative messages written to  
96073                                standard output.
- 96075        *LC\_TIME*            Determine the format and contents of date and time strings displayed in the *lp*  
96076                                banner page, if any.
- 96077        *LPDEST*            Determine the destination. If the *LPDEST* environment variable is not set, the  
96078                                *PRINTER* environment variable shall be used. The **-d dest** option takes precedence  
96079                                over *LPDEST*. Results are undefined when **-d** is not specified and *LPDEST*  
96080                                contains a value that is not a valid destination name.
- 96081        XSI        *NLSPATH*            Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 96082        *PRINTER*           Determine the output device or destination. If the *LPDEST* and *PRINTER*  
96083                                environment variables are not set, an unspecified output device is used. The **-d**  
96084                                *dest* option and the *LPDEST* environment variable shall take precedence over  
96085                                *PRINTER*. Results are undefined when **-d** is not specified, *LPDEST* is unset, and  
96086                                *PRINTER* contains a value that is not a valid device or destination name.



96087           TZ           Determine the timezone used to calculate date and time strings displayed in the *lp*  
96088                           banner page, if any. If *TZ* is unset or null, an unspecified default timezone shall be  
96089                           used.

## 96090 ASYNCHRONOUS EVENTS

96091           Default.

## 96092 STDOUT

96093           The *lp* utility shall write a *request ID* to the standard output, unless *-s* is specified. The format of  
96094           the message is unspecified. The request ID can be used on systems supporting the historical  
96095           *cancel* and *lpstat* utilities.

## 96096 STDERR

96097           The standard error shall be used only for diagnostic messages.

## 96098 OUTPUT FILES

96099           None.

## 96100 EXTENDED DESCRIPTION

96101           None.

## 96102 EXIT STATUS

96103           The following exit values shall be returned:

96104           0   All input files were processed successfully.

96105           >0 No output device was available, or an error occurred.

## 96106 CONSEQUENCES OF ERRORS

96107           Default.

## 96108 APPLICATION USAGE

96109           The *pr* and *fold* utilities can be used to achieve reasonable formatting for the implementation's  
96110           default page size.

96111           A conforming application can use one of the *file* operands only with the *-c* option or if the file is  
96112           publicly readable and guaranteed to be available at the time of printing. This is because  
96113           POSIX.1-2008 gives the implementation the freedom to queue up the request for printing at  
96114           some later time by a different process that might not be able to access the file.

## 96115 EXAMPLES

96116           1. To print file *file*:

96117                 `lp -c file`

96118           2. To print multiple files with headers:

96119                 `pr file1 file2 | lp`

## 96120 RATIONALE

96121           The *lp* utility was designed to be a basic version of a utility that is already available in many  
96122           historical implementations. The standard developers considered that it should be implementable  
96123           simply as:

96124                 `cat "$@" > /dev/lp`

96125           after appropriate processing of options, if that is how the implementation chose to do it and if  
96126           exclusive access could be granted (so that two users did not write to the device simultaneously).  
96127           Although in the future the standard developers may add other options to this utility, it should  
96128           always be able to execute with no options or operands and send the standard input to an

unspecified output device.

This volume of POSIX.1-2008 makes no representations concerning the format of the printed output, except that it must be “human-readable” and “non-volatile”. Thus, writing by default to a disk or tape drive or a display terminal would not qualify. (Such destinations are not prohibited when `-d dest`, `LPDEST`, or `PRINTER` are used, however.)

This volume of POSIX.1-2008 is worded such that a “print job” consisting of multiple input files, possibly in multiple copies, is guaranteed to print so that any one file is not intermixed with another, but there is no statement that all the files or copies have to print out together.

The `-c` option may imply a spooling operation, but this is not required. The utility can be implemented to wait until the printer is ready and then wait until it is finished. Because of that, there is no attempt to define a queuing mechanism (priorities, classes of output, and so on).

On some historical systems, the request ID reported on the `STDOUT` can be used to later cancel or find the status of a request using utilities not defined in this volume of POSIX.1-2008.

Although the historical System V `lp` and BSD `lpr` utilities have provided similar functionality, they used different names for the environment variable specifying the destination printer. Since the name of the utility here is `lp`, `LPDEST` (used by the System V `lp` utility) was given precedence over `PRINTER` (used by the BSD `lpr` utility). Since environments of users frequently contain one or the other environment variable, the `lp` utility is required to recognize both. If this was not done, many applications would send output to unexpected output devices when users moved from system to system.

Some have commented that `lp` has far too little functionality to make it worthwhile. Requests have proposed additional options or operands or both that added functionality. The requests included:

- Wording *requiring* the output to be “hardcopy”
- A requirement for multiple printers
- Options for supporting various page-description languages

Given that a compliant system is not required to even have a printer, placing further restrictions upon the behavior of the printer is not useful. Since hardcopy format is so application-dependent, it is difficult, if not impossible, to select a reasonable subset of functionality that should be required on all compliant systems.

The term *unspecified* is used in this section in lieu of *implementation-defined* as most known implementations would not be able to make definitive statements in their conformance documents; the existence and usage of printers is very dependent on how the system administrator configures each individual system.

Since the default destination, device type, queuing mechanisms, and acceptable forms of input are all unspecified, usage guidelines for what a conforming application can do are as follows:

- Use the command in a pipeline, or with `-c`, so that there are no permission problems and the files can be safely deleted or modified.
- Limit output to text files of reasonable line lengths and printable characters and include no device-specific formatting information, such as a page description language. The meaning of “reasonable” in this context can only be answered as a quality-of-implementation issue, but it should be apparent from historical usage patterns in the industry and the locale. The `pr` and `fold` utilities can be used to achieve reasonable formatting for the default page size of the implementation.

Alternatively, the application can arrange its installation in such a way that it requires the system administrator or operator to provide the appropriate information on *lp* options and environment variable values.

At a minimum, having this utility in this volume of POSIX.1-2008 tells the industry that conforming applications require a means to print output and provides at least a command name and *LPDEST* routing mechanism that can be used for discussions between vendors, application developers, and users. The use of “should” in the DESCRIPTION of *lp* clearly shows the intent of the standard developers, even if they cannot mandate that all systems (such as laptops) have printers.

This volume of POSIX.1-2008 does not specify what the ownership of the process performing the writing to the output device may be. If *-c* is not used, it is unspecified whether the process performing the writing to the output device has permission to read *file* if there are any restrictions in place on who may read *file* until after it is printed. Also, if *-c* is not used, the results of deleting *file* before it is printed are unspecified.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*mailx*

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

## CHANGE HISTORY

First released in Issue 2.

### Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the DESCRIPTION, the requirement to associate a unique request ID, and the normal generation of a banner page is added.
- In the OPTIONS section:
  - The *-d dest* description is expanded, but references to *lpstat* are removed.
  - The *-m*, *-o*, *-s*, *-t*, and *-w* options are added.
- In the ENVIRONMENT VARIABLES section, *LC\_TIME* may now affect the execution.
- The STDOUT section is added.

The normative text is reworded to avoid use of the term “must” for application requirements.

The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

### Issue 7

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

**NAME**

ls — list directory contents

**SYNOPSIS**

```
ls [-ikqrs] [-g lno] [-A|-a] [-C|-m|-x|-l] \
  [-F|-p] [-H|-L] [-R|-d] [-S|-f|-t] [-c|-u] [file...]
```

**DESCRIPTION**

For each operand that names a file of a type other than directory or symbolic link to a directory, *ls* shall write the name of the file as well as any requested, associated information. For each operand that names a file of type directory, *ls* shall write the names of files contained within the directory as well as any requested, associated information. Filenames beginning with a <period> ( ' . ' ) and any associated information shall not be written out unless explicitly referenced, the *-A* or *-a* option is supplied, or an implementation-defined condition causes them to be written. If one or more of the *-d*, *-F*, or *-l* options are specified, and neither the *-H* nor the *-L* option is specified, for each operand that names a file of type symbolic link to a directory, *ls* shall write the name of the file as well as any requested, associated information. If none of the *-d*, *-F*, or *-l* options are specified, or the *-H* or *-L* options are specified, for each operand that names a file of type symbolic link to a directory, *ls* shall write the names of files contained within the directory as well as any requested, associated information. In each case where the names of files contained within a directory are written, if the directory contains any symbolic links then *ls* shall evaluate the file information and file type to be those of the symbolic link itself, unless the *-L* option is specified.

If no operands are specified, *ls* shall behave as if a single operand of dot ( ' . ' ) had been specified. If more than one operand is specified, *ls* shall write non-directory operands first; it shall sort directory and non-directory operands separately according to the collating sequence in the current locale.

Whenever *ls* sorts filenames or pathnames according to the collating sequence in the current locale, if this collating sequence does not have a total ordering of all characters (see XBD [Section 7.3.2](#), on page 147), then any filenames or pathnames that collate equally should be further compared byte-by-byte using the collating sequence for the POSIX locale.

The *ls* utility shall detect infinite loops; that is, entering a previously visited directory that is an ancestor of the last file encountered. When it detects an infinite loop, *ls* shall write a diagnostic message to standard error and shall either recover its position in the hierarchy or terminate.

**OPTIONS**

The *ls* utility shall conform to XBD [Section 12.2](#) (on page 216).

The following options shall be supported:

- A** Write out all directory entries, including those whose names begin with a <period> ( ' . ' ) but excluding the entries dot and dot-dot (if they exist).
- C** Write multi-text-column output with entries sorted down the columns, according to the collating sequence. The number of text columns and the column separator characters are unspecified, but should be adapted to the nature of the output device. This option disables long format output.
- F** Do not follow symbolic links named as operands unless the *-H* or *-L* options are specified. Write a <slash> ( ' / ' ) immediately after each pathname that is a directory, an <asterisk> ( ' \* ' ) after each that is executable, a <vertical-line> ( ' | ' ) after each that is a FIFO, and an at-sign ( ' @ ' ) after each that is a symbolic link. For other file types, other symbols may be written.

|       |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|-------|-----|-----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 96254 |     | <b>-H</b> | Evaluate the file information and file type for symbolic links specified on the command line to be those of the file referenced by the link, and not the link itself; however, <i>ls</i> shall write the name of the link itself and not the file referenced by the link.                                                                                                                                                                                                                                      |
| 96255 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96256 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96257 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96258 |     | <b>-L</b> | Evaluate the file information and file type for all symbolic links (whether named on the command line or encountered in a file hierarchy) to be those of the file referenced by the link, and not the link itself; however, <i>ls</i> shall write the name of the link itself and not the file referenced by the link. When <b>-L</b> is used with <b>-l</b> , write the contents of symbolic links in the long format (see the STDOUT section).                                                               |
| 96259 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96260 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96261 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96262 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96263 |     | <b>-R</b> | Recursively list subdirectories encountered. When a symbolic link to a directory is encountered, the directory shall not be recursively listed unless the <b>-L</b> option is specified. The use of <b>-R</b> with <b>-d</b> or <b>-f</b> produces unspecified results.                                                                                                                                                                                                                                        |
| 96264 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96265 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96266 |     | <b>-S</b> | Sort with the primary key being file size (in decreasing order) and the secondary key being filename in the collating sequence (in increasing order).                                                                                                                                                                                                                                                                                                                                                          |
| 96267 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96268 |     | <b>-a</b> | Write out all directory entries, including those whose names begin with a <period> ( ' . ' ).                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 96269 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96270 |     | <b>-c</b> | Use time of last modification of the file status information (see XBD <sys/stat.h>) instead of last modification of the file itself for sorting ( <b>-t</b> ) or writing ( <b>-l</b> ).                                                                                                                                                                                                                                                                                                                        |
| 96271 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96272 |     | <b>-d</b> | Do not follow symbolic links named as operands unless the <b>-H</b> or <b>-L</b> options are specified. Do not treat directories differently than other types of files. The use of <b>-d</b> with <b>-R</b> or <b>-f</b> produces unspecified results.                                                                                                                                                                                                                                                         |
| 96273 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96274 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96275 |     | <b>-f</b> | List the entries in directory operands in the order they appear in the directory. The behavior for non-directory operands is unspecified. This option shall turn on <b>-a</b> . When <b>-f</b> is specified, any occurrences of the <b>-r</b> , <b>-S</b> , and <b>-t</b> options shall be ignored and any occurrences of the <b>-A</b> , <b>-g</b> , <b>-l</b> , <b>-n</b> , <b>-o</b> , and <b>-s</b> options may be ignored. The use of <b>-f</b> with <b>-R</b> or <b>-d</b> produces unspecified results. |
| 96276 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96277 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96278 | XSI |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96279 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96280 | XSI | <b>-g</b> | Turn on the <b>-l</b> (ell) option, but disable writing the file's owner name or number. Disable the <b>-C</b> , <b>-m</b> , and <b>-x</b> options.                                                                                                                                                                                                                                                                                                                                                            |
| 96281 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96282 |     | <b>-i</b> | For each file, write the file's file serial number (see <i>stat()</i> in the System Interfaces volume of POSIX.1-2008).                                                                                                                                                                                                                                                                                                                                                                                        |
| 96283 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96284 |     | <b>-k</b> | Set the block size for the <b>-s</b> option and the per-directory block count written for the <b>-l</b> , <b>-n</b> , <b>-s</b> , <b>-g</b> , and <b>-o</b> options (see the STDOUT section) to 1 024 bytes.                                                                                                                                                                                                                                                                                                   |
| 96285 | XSI |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96286 |     | <b>-l</b> | (The letter ell.) Do not follow symbolic links named as operands unless the <b>-H</b> or <b>-L</b> options are specified. Write out in long format (see the STDOUT section). Disable the <b>-C</b> , <b>-m</b> , and <b>-x</b> options.                                                                                                                                                                                                                                                                        |
| 96287 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96288 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96289 |     | <b>-m</b> | Stream output format; list pathnames across the page, separated by a <comma> character followed by a <space> character. Use a <newline> character as the list terminator and after the separator sequence when there is not room on a line for the next list entry. This option disables long format output.                                                                                                                                                                                                   |
| 96290 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96291 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96292 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96293 |     | <b>-n</b> | Turn on the <b>-l</b> (ell) option, but when writing the file's owner or group, write the file's numeric UID or GID rather than the user or group name, respectively. Disable the <b>-C</b> , <b>-m</b> , and <b>-x</b> options.                                                                                                                                                                                                                                                                               |
| 96294 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96295 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96296 | XSI | <b>-o</b> | Turn on the <b>-l</b> (ell) option, but disable writing the file's group name or number. Disable the <b>-C</b> , <b>-m</b> , and <b>-x</b> options.                                                                                                                                                                                                                                                                                                                                                            |
| 96297 |     |           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

|       |           |                                                                                                                                      |
|-------|-----------|--------------------------------------------------------------------------------------------------------------------------------------|
| 96298 | <b>-p</b> | Write a <slash> ( ' / ' ) after each filename if that file is a directory.                                                           |
| 96299 | <b>-q</b> | Force each instance of non-printable filename characters and <tab> characters to be                                                  |
| 96300 |           | written as the <question-mark> ( ' ? ' ) character. Implementations may provide this                                                 |
| 96301 |           | option by default if the output is to a terminal device.                                                                             |
| 96302 | <b>-r</b> | Reverse the order of the sort to get reverse collating sequence oldest first, or                                                     |
| 96303 |           | smallest file size first depending on the other options given.                                                                       |
| 96304 | <b>-s</b> | Indicate the total number of file system blocks consumed by each file displayed. If                                                  |
| 96305 |           | the <b>-k</b> option is also specified, the block size shall be 1 024 bytes; otherwise, the                                          |
| 96306 |           | block size is implementation-defined.                                                                                                |
| 96307 | <b>-t</b> | Sort with the primary key being time modified (most recently modified first) and                                                     |
| 96308 |           | the secondary key being filename in the collating sequence. For a symbolic link,                                                     |
| 96309 |           | the time used as the sort key is that of the symbolic link itself, unless <i>ls</i> is                                               |
| 96310 |           | evaluating its file information to be that of the file referenced by the link (see the                                               |
| 96311 |           | <b>-H</b> and <b>-L</b> options).                                                                                                    |
| 96312 | <b>-u</b> | Use time of last access (see XBD <a href="#">&lt;sys/stat.h&gt;</a> ) instead of last modification of the file                       |
| 96313 |           | for sorting ( <b>-t</b> ) or writing ( <b>-l</b> ).                                                                                  |
| 96314 | <b>-x</b> | The same as <b>-C</b> , except that the multi-text-column output is produced with entries                                            |
| 96315 |           | sorted across, rather than down, the columns. This option disables long format                                                       |
| 96316 |           | output.                                                                                                                              |
| 96317 | <b>-1</b> | (The numeric digit one.) Force output to be one entry per line. This option does                                                     |
| 96318 | XSI       | not disable long format output. (Long format output is enabled by <b>-g</b> , <b>-l</b> (ell), <b>-n</b> ,                           |
| 96319 | XSI       | and <b>-o</b> ; and disabled by <b>-C</b> , <b>-m</b> , and <b>-x</b> .)                                                             |
| 96320 | XSI       | If an option that enables long format output ( <b>-g</b> , <b>-l</b> (ell), <b>-n</b> , and <b>-o</b> ) is given with an option that |
| 96321 |           | disables long format output ( <b>-C</b> , <b>-m</b> , and <b>-x</b> ), this shall not be considered an error. The last of            |
| 96322 |           | these options specified shall determine whether long format output is written.                                                       |
| 96323 |           | If <b>-R</b> , <b>-d</b> , or <b>-f</b> are specified, the results of specifying these mutually-exclusive options are                |
| 96324 |           | specified by the descriptions of these options above. If more than one of any of the other options                                   |
| 96325 |           | shown in the SYNOPSIS section in mutually-exclusive sets are given, this shall not be considered                                     |
| 96326 |           | an error; the last option specified in each set shall determine the output.                                                          |
| 96327 |           | Note that if <b>-t</b> is specified, <b>-c</b> and <b>-u</b> are not only mutually-exclusive with each other, they are               |
| 96328 |           | also mutually-exclusive with <b>-S</b> when determining sort order. But even if <b>-S</b> is specified after all                     |
| 96329 |           | occurrences of <b>-c</b> , <b>-t</b> , and <b>-u</b> , the last use of <b>-c</b> or <b>-u</b> determines the timestamp printed when  |
| 96330 |           | producing long format output.                                                                                                        |

## OPERANDS

The following operand shall be supported:

|             |                                                                                                                                 |
|-------------|---------------------------------------------------------------------------------------------------------------------------------|
| <i>file</i> | A pathname of a file to be written. If the file specified is not found, a diagnostic message shall be output on standard error. |
|-------------|---------------------------------------------------------------------------------------------------------------------------------|

## STDIN

Not used.

## INPUT FILES

None.



**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *ls*:

**COLUMNS** Determine the user's preferred column position width for writing multiple text-column output. If this variable contains a string representing a decimal integer, the *ls* utility shall calculate how many pathname text columns to write (see **-C**) based on the width provided. If **COLUMNS** is not set or invalid, an implementation-defined number of column positions shall be assumed, based on the implementation's knowledge of the output device. The column width chosen to write the names of files in any given directory shall be constant. Filenames shall not be truncated to fit into the multiple text-column output.

**LANG** Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

**LC\_ALL** If set to a non-empty string value, override the values of all the other internationalization variables.

**LC\_COLLATE** Determine the locale for character collation information in determining the pathname collation sequence.

**LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments) and which characters are defined as printable (character class **print**).

**LC\_MESSAGES** Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

**LC\_TIME** Determine the format and contents for date and time strings written by *ls*.

**NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

**TZ** Determine the timezone for date and time strings written by *ls*. If **TZ** is unset or null, an unspecified default timezone shall be used.

**ASYNCHRONOUS EVENTS**

Default.

**STDOUT**

The default format shall be to list one entry per line to standard output; the exceptions are to terminals or when one of the **-C**, **-m**, or **-x** options is specified. If the output is to a terminal, the format is implementation-defined.

When **-m** is specified, the format used for the last element of the list shall be:

"%s\n", <filename>

The format used for each other element of the list shall be:

"%s,%s", <filename>, <separator>

where, if there is not room for the next element of the list to fit within the current line length, <separator> is a string containing an optional <space> character and a mandatory <newline> character; otherwise it is a single <space> character.

If the **-i** option is specified, the file's file serial number (see XBD [<sys/stat.h>](#)) shall be written in the following format before any other output for the corresponding entry:

96382        %u ", <file serial number>

96383        If the **-l** option is specified, the following information shall be written for files other than  
96384        character special and block special files:

96385        "%s %u %s %s %u %s %s\n", <file mode>, <number of links>,  
96386        <owner name>, <group name>, <size>, <date and time>,  
96387        <pathname>

96388        If the **-l** option is specified, the following information shall be written for character special and  
96389        block special files:

96390        "%s %u %s %s %s %s %s\n", <file mode>, <number of links>,  
96391        <owner name>, <group name>, <device info>, <date and time>,  
96392        <pathname>

96393        In both cases if the file is a symbolic link and the **-L** option is also specified, this information  
96394        shall be for the file resolved from the symbolic link, except that the <pathname> field shall  
96395        contain the pathname of the symbolic link itself. If the file is a symbolic link and the **-L** option is  
96396        not specified, this information shall be about the link itself and the <pathname> field shall be of  
96397        the form:

96398        "%s -> %s", <pathname of link>, <contents of link>

96399        XSI        The **-n**, **-g**, and **-o** options use the same format as **-l**, but with omitted items and their  
96400        associated <blank> characters. See the OPTIONS section.

96401        In both the preceding **-l** forms, if <owner name> or <group name> cannot be determined, or if **-n**  
96402        is given, they shall be replaced with their associated numeric values using the format %u.

96403        The <size> field shall contain the value that would be returned for the file in the *st\_size* field of  
96404        **struct stat** (see XBD <sys/stat.h>). Note that for some file types this value is unspecified.

96405        The <device info> field shall contain implementation-defined information associated with the  
96406        device in question.

96407        The <date and time> field shall contain the appropriate date and timestamp of when the file was  
96408        last modified. In the POSIX locale, the field shall be the equivalent of the output of the following  
96409        *date* command:

96410        date "+%b %e %H:%M"

96411        if the file has been modified in the last six months, or:

96412        date "+%b %e %Y"

96413        (where two <space> characters are used between %e and %Y) if the file has not been modified in  
96414        the last six months or if the modification date is in the future, except that, in both cases, the final  
96415        <newline> produced by *date* shall not be included and the output shall be as if the *date*  
96416        command were executed at the time of the last modification date of the file rather than the  
96417        current time. When the *LC\_TIME* locale category is not set to the POSIX locale, a different format  
96418        and order of presentation of this field may be used.

96419        If the pathname was specified as a *file* operand, it shall be written as specified.

96420        XSI        The file mode written under the **-l**, **-n**, **-g**, and **-o** options shall consist of the following format:

96421        "%c%s%s%s", <entry type>, <owner permissions>,  
96422        <group permissions>, <other permissions>,  
96423        <optional alternate access method flag>



96424 The *<optional alternate access method flag>* shall be the empty string if there is no alternate or  
 96425 additional access control method associated with the file; otherwise, it shall be a string  
 96426 containing a single printable character that is not a <blank>.

96427 The *<entry type>* character shall describe the type of file, as follows:

- 96428 d Directory.
- 96429 b Block special file.
- 96430 c Character special file.
- 96431 l (ell) Symbolic link.
- 96432 p FIFO.
- 96433 – Regular file.

96434 Implementations may add other characters to this list to represent other implementation-defined  
 96435 file types.

96436 The next three fields shall be three characters each:

96437 *<owner permissions>*

96438 Permissions for the file owner class (see XBD [Section 4.5](#), on page 108).

96439 *<group permissions>*

96440 Permissions for the file group class.

96441 *<other permissions>*

96442 Permissions for the file other class.

96443 Each field shall have three character positions:

- 96444 1. If 'r', the file is readable; if '–', the file is not readable.
- 96445 2. If 'w', the file is writable; if '–', the file is not writable.
- 96446 3. The first of the following that applies:
  - 96447 S If in *<owner permissions>*, the file is not executable and set-user-ID mode is set. If in
  - 96448 *<group permissions>*, the file is not executable and set-group-ID mode is set.
  - 96449 s If in *<owner permissions>*, the file is executable and set-user-ID mode is set. If in
  - 96450 *<group permissions>*, the file is executable and set-group-ID mode is set.
  - 96451 XSI T If in *<other permissions>* and the file is a directory, search permission is not granted to
  - 96452 others, and the restricted deletion flag is set.
  - 96453 XSI t If in *<other permissions>* and the file is a directory, search permission is granted to
  - 96454 others, and the restricted deletion flag is set.
  - 96455 x The file is executable or the directory is searchable.
  - 96456 – None of the attributes of 'S', 's', 'T', 't', or 'x' applies.

96457 Implementations may add other characters to this list for the third character position.  
 96458 Such additions shall, however, be written in lowercase if the file is executable or  
 96459 searchable, and in uppercase if it is not.

96460 XSI If any of the **–l**, **–n**, **–s**, **–g**, or **–o** options is specified, each list of files within the directory shall be  
 96461 preceded by a status line indicating the number of file system blocks occupied by files in the  
 96462 directory in 512-byte units if the **–k** option is not specified, or 1024-byte units if the **–k** option is  
 96463 specified, rounded up to the next integral number of units, if necessary. In the POSIX locale, the

96464 format shall be:

96465 "total %u\n", <number of units in the directory>

96466 If more than one directory, or a combination of non-directory files and directories are written,  
96467 either as a result of specifying multiple operands, or the **-R** option, each list of files within a  
96468 directory shall be preceded by:

96469 "\n%s:\n", <directory name>

96470 If this string is the first thing to be written, the first <newline> shall not be written. This output  
96471 shall precede the number of units in the directory.

96472 If the **-s** option is given, each file shall be written with the number of blocks used by the file.  
96473 XSI Along with **-C**, **-l**, **-m**, or **-x**, the number and a <space> shall precede the filename; with **-l**, **-n**,  
96474 **-g**, or **-o**, they shall precede each line describing a file.

#### 96475 **STDERR**

96476 The standard error shall be used only for diagnostic messages.

#### 96477 **OUTPUT FILES**

96478 None.

#### 96479 **EXTENDED DESCRIPTION**

96480 None.

#### 96481 **EXIT STATUS**

96482 The following exit values shall be returned:

96483 0 Successful completion.

96484 >0 An error occurred.

#### 96485 **CONSEQUENCES OF ERRORS**

96486 Default.

#### 96487 **APPLICATION USAGE**

96488 Many implementations use the <equals-sign> (' = ') to denote sockets bound to the file system  
96489 for the **-F** option. Similarly, many historical implementations use the 's' character to denote  
96490 sockets as the entry type characters for the **-l** option.

96491 It is difficult for an application to use every part of the file modes field of *ls -l* in a portable  
96492 manner. Certain file types and executable bits are not guaranteed to be exactly as shown, as  
96493 implementations may have extensions. Applications can use this field to pass directly to a user  
96494 printout or prompt, but actions based on its contents should generally be deferred, instead, to  
96495 the *test* utility.

96496 The output of *ls* (with the **-l** and related options) contains information that logically could be  
96497 used by utilities such as *chmod* and *touch* to restore files to a known state. However, this  
96498 information is presented in a format that cannot be used directly by those utilities or be easily  
96499 translated into a format that can be used. A character has been added to the end of the  
96500 permissions string so that applications at least have an indication that they may be working in  
96501 an area they do not understand instead of assuming that they can translate the permissions  
96502 string into something that can be used. Future versions or related documents may define one or  
96503 more specific characters to be used based on different standard additional or alternative access  
96504 control mechanisms.

96505 As with many of the utilities that deal with filenames, the output of *ls* for multiple files or in one  
96506 of the long listing formats must be used carefully on systems where filenames can contain  
96507 embedded white space. Systems and system administrators should institute policies and user

training to limit the use of such filenames.

The number of disk blocks occupied by the file that it reports varies depending on underlying file system type, block size units reported, and the method of calculating the number of blocks. On some file system types, the number is the actual number of blocks occupied by the file (counting indirect blocks and ignoring holes in the file); on others it is calculated based on the file size (usually making an allowance for indirect blocks, but ignoring holes).

#### EXAMPLES

An example of a small directory tree being fully listed with *ls -laRF a* in the POSIX locale:

```
total 11
drwxr-xr-x  3 fox      prog          64 Jul  4 12:07 ./
drwxrwxrwx  4 fox      prog        3264 Jul  4 12:09 ../
drwxr-xr-x  2 fox      prog          48 Jul  4 12:07 b/
-rwxr--r--  1 fox      prog        572 Jul  4 12:07 foo*

a/b:
total 4
drwxr-xr-x  2 fox      prog          48 Jul  4 12:07 ./
drwxr-xr-x  3 fox      prog          64 Jul  4 12:07 ../
-rw-r--r--  1 fox      prog        700 Jul  4 12:07 bar
```

#### RATIONALE

Some historical implementations of the *ls* utility show all entries in a directory except dot and dot-dot when a superuser invokes *ls* without specifying the *-a* option. When “normal” users invoke *ls* without specifying *-a*, they should not see information about any files with names beginning with a <period> unless they were named as *file* operands.

Implementations are expected to traverse arbitrary depths when processing the *-R* option. The only limitation on depth should be based on running out of physical storage for keeping track of untraversed directories.

The *-1* (one) option was historically found in BSD and BSD-derived implementations only. It is required in this volume of POSIX.1-2008 so that conforming applications might ensure that output is one entry per line, even if the output is to a terminal.

The *-S* option was added in Issue 7, but had been provided by several implementations for many years. The description given in the standard documents historic practice, but does not match much of the documentation that described its behavior. Historical documentation typically described it as something like:

*-S*            Sort by size (largest size first) instead of by name. Special character devices (listed last) are sorted by name.

even though the file type was never considered when sorting the output. Character special files do typically sort close to the end of the list because their file size on most implementations is zero. But they are sorted alphabetically with any other files that happen to have the same file size (zero), not sorted separately and added to the end.

This volume of POSIX.1-2008 is frequently silent about what happens when mutually-exclusive options are specified. Except for *-R*, *-d*, and *-f*, the *ls* utility is required to accept multiple options from each mutually-exclusive option set without treating them as errors and to use the behavior specified by the last option given in each mutually-exclusive set. Since *ls* is one of the most aliased commands, it is important that the implementation perform intuitively. For example, if the alias were:

```
alias ls="ls -C"
```

and the user typed `ls -1` (one), single-text-column output should result, not an error.

The `-g`, `-l` (ell), `-n`, and `-o` options are not mutually-exclusive options. They all enable long format output. They work together to determine whether the file's owner is written (no if `-g` is present), file's group is written (no if `-o` is present), and if the file's group or owner is written whether it is written as the name (default) or a string representation of the UID or GID number (if `-n` is present). The `-C`, `-m`, `-x`, and `-1` (one) are mutually-exclusive options and the first three of these disable long format output. The `-1` (one) option does not directly change whether or not long format output is enabled, but by overriding `-C`, `-m`, and `-x`, it can re-enable long format output that had been disabled by one of these options.

Earlier versions of this standard did not describe the BSD `-A` option (like `-a`, but dot and dot-dot are not written out). It has been added due to widespread implementation.

Implementations may make `-q` the default for terminals to prevent trojan horse attacks on terminals with special escape sequences. This is not required because:

- Some control characters may be useful on some terminals; for example, a system might write them as `"\001"` or `"^A"`.
- Special behavior for terminals is not relevant to applications portability.

An early proposal specified that the *<optional alternate access method flag>* had to be `'+'` if there was an alternate access method used on the file or `<space>` if there was not. This was changed to be `<space>` if there is not and a single printable character if there is. This was done for three reasons:

1. There are historical implementations using characters other than `'+'`.
2. There are implementations that vary this character used in that position to distinguish between various alternate access methods in use.
3. The standard developers did not want to preclude future specifications that might need a way to specify more than one alternate access method.

Nonetheless, implementations providing a single alternate access method are encouraged to use `'+'`.

Earlier versions of this standard did not have the `-k` option, which meant that the `-s` option could not be used portably as its block size was implementation-defined, and the units used to specify the number of blocks occupied by files in a directory in an `ls -l` listing were fixed as 512-byte units. The `-k` option has been added to provide a way for the `-s` option to be used portably, and for consistency it also changes the aforementioned units from 512-byte to 1024-byte.

The *<date and time>* field in the `-l` format is specified only for the POSIX locale. As noted, the format can be different in other locales. No mechanism for defining this is present in this volume of POSIX.1-2008, as the appropriate vehicle is a messaging system; that is, the format should be specified as a "message".

## FUTURE DIRECTIONS

Allowing `-f` to ignore the `-A`, `-g`, `-l`, `-n`, `-o`, and `-s` options may be removed in a future version.

A future version of this standard may require that if the collating sequence for the current locale does not have a total ordering of all characters, any filenames or pathnames that collate equally are further compared byte-by-byte using the collating sequence for the POSIX locale.

**SEE ALSO***chmod, find*

XBD [Section 7.3.2](#) (on page 147), [Section 4.5](#) (on page 108), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216), [<sys/stat.h>](#)

XSH *fstatat()***CHANGE HISTORY**

First released in Issue 2.

**Issue 5**

A second FUTURE DIRECTION is added.

**Issue 6**

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- In the **-F** option, other symbols are allowed for other file types.

Treatment of symbolic links is added, as defined in the IEEE P1003.2b draft standard.

The Open Group Base Resolution bwg2001-010 is applied, adding the **T** and **t** fields as part of the **XSI** option.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #101 is applied, clarifying the optional alternate access method flag in the **STDOUT** section.

Austin Group Interpretation 1003.1-2001 #128 is applied, clarifying the **DESCRIPTION** and the definition of the **-R** option.

Austin Group Interpretation 1003.1-2001 #129 is applied, clarifying the behavior of *ls* when no operands are specified.

Austin Group Interpretation 1003.1-2001 #198 is applied, clarifying the requirements for the **-H** option.

SD5-XCU-ERN-50 is applied, adding the **-A** option.SD5-XCU-ERN-97 is applied, updating the **SYNOPSIS**.

The **-S** option is added from The Open Group Technical Standard, 2006, Extended API Set Part 1.

The **-f**, **-m**, **-n**, **-p**, **-s**, and **-x** options are moved from the **XSI** option to the **Base**.

The description of the **-f**, **-s**, and **-t** options are revised and the **-k** option is added.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0098 [424], XCU/TC1-2008/0099 [424], XCU/TC1-2008/0100 [424], XCU/TC1-2008/0101 [424], XCU/TC1-2008/0102 [424], XCU/TC1-2008/0103 [424], XCU/TC1-2008/0104 [424], XCU/TC1-2008/0105 [423,424], XCU/TC1-2008/0106 [424], XCU/TC1-2008/0107 [424], XCU/TC1-2008/0108 [424], XCU/TC1-2008/0109 [424], XCU/TC1-2008/0110 [424], XCU/TC1-2008/0111 [423], XCU/TC1-2008/0112 [117], XCU/TC1-2008/0113 [117], XCU/TC1-2008/0114 [117], XCU/TC1-2008/0115 [424], and XCU/TC1-2008/0116 [424] are applied.

96634 **NAME**

96635 m4 — macro processor

96636 **SYNOPSIS**96637 m4 [-s] [-D *name*[=*val*]]... [-U *name*]... *file*...96638 **DESCRIPTION**96639 The *m4* utility is a macro processor that shall read one or more text files, process them according  
96640 to their included macro statements, and write the results to standard output.96641 **OPTIONS**96642 The *m4* utility shall conform to XBD [Section 12.2](#) (on page 216), except that the order of the **-D**  
96643 and **-U** options shall be significant, and options can be interspersed with operands.

96644 The following options shall be supported:

96645 **-s** Enable line synchronization output for the *c99* preprocessor phase (that is, **#line**  
96646 directives).96647 **-D *name*[=*val*]**96648 Define *name* to *val* or to null if *=val* is omitted.96649 **-U *name*** Undefine *name*.96650 **OPERANDS**

96651 The following operand shall be supported:

96652 *file* A pathname of a text file to be processed. If no *file* is given, or if it is '-', the  
96653 standard input shall be read.96654 **STDIN**96655 The standard input shall be a text file that is used if no *file* operand is given, or if it is '-'.96656 **INPUT FILES**96657 The input file named by the *file* operand shall be a text file.96658 **ENVIRONMENT VARIABLES**96659 The following environment variables shall affect the execution of *m4*:96660 **LANG** Provide a default value for the internationalization variables that are unset or null.  
96661 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
96662 variables used to determine the values of locale categories.)96663 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
96664 internationalization variables.96665 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
96666 characters (for example, single-byte as opposed to multi-byte characters in  
96667 arguments and input files).96668 **LC\_MESSAGES**96669 Determine the locale that should be used to affect the format and contents of  
96670 diagnostic messages written to standard error.96671 **XSI NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.96672 **ASYNCHRONOUS EVENTS**

96673 Default.



**STDOUT**

The standard output shall be the same as the input files, after being processed for macro expansion.

**STDERR**

The standard error shall be used to display strings with the **errprint** macro, macro tracing enabled by the **traceon** macro, the defined text for macros written by the **dumpdef** macro, or for diagnostic messages.

**OUTPUT FILES**

None.

**EXTENDED DESCRIPTION**

The *m4* utility shall compare each token from the input against the set of built-in and user-defined macros. If the token matches the name of a macro, then the token shall be replaced by the macro's defining text, if any, and rescanned for matching macro names. Once no portion of the token matches the name of a macro, it shall be written to standard output. Macros may have arguments, in which case the arguments shall be substituted into the defining text before it is rescanned.

Macro calls have the form:

```
name(arg1, arg2, ..., argn)
```

Macro names shall consist of letters, digits, and underscores, where the first character is not a digit. Tokens not of this form shall not be treated as macros.

The application shall ensure that the <left-parenthesis> immediately follows the name of the macro. If a token matching the name of a macro is not followed by a <left-parenthesis>, it is handled as a use of that macro without arguments.

If a macro name is followed by a <left-parenthesis>, its arguments are the <comma>-separated tokens between the <left-parenthesis> and the matching <right-parenthesis>. Unquoted white-space characters preceding each argument shall be ignored. All other characters, including trailing white-space characters, are retained. <comma> characters enclosed between <left-parenthesis> and <right-parenthesis> characters do not delimit arguments.

Arguments are positionally defined and referenced. The string "\$1" in the defining text shall be replaced by the first argument. Systems shall support at least nine arguments; only the first nine can be referenced, using the strings "\$1" to "\$9", inclusive. The string "\$0" is replaced with the name of the macro. The string "\$#" is replaced by the number of arguments as a string. The string "\$\*" is replaced by a list of all of the arguments, separated by <comma> characters. The string "\$@" is replaced by a list of all of the arguments separated by <comma> characters, and each argument is quoted using the current left and right quoting strings. The string "\${" produces unspecified behavior.

If fewer arguments are supplied than are in the macro definition, the omitted arguments are taken to be null. It is not an error if more arguments are supplied than are in the macro definition.

No special meaning is given to any characters enclosed between matching left and right quoting strings, but the quoting strings are themselves discarded. By default, the left quoting string consists of a grave accent (backquote) and the right quoting string consists of an acute accent (single-quote); see also the **changequote** macro.

Comments are written but not scanned for matching macro names; by default, the begin-comment string consists of the <number-sign> character and the end-comment string consists of a <newline>. See also the **changecom** and **dnl** macros.

96720 The *m4* utility shall make available the following built-in macros. They can be redefined, but  
 96721 once this is done the original meaning is lost. Their values shall be null unless otherwise stated.  
 96722 In the descriptions below, the term *defining text* refers to the value of the macro: the second  
 96723 argument to the **define** macro, among other things. Except for the first argument to the **eval**  
 96724 macro, all numeric arguments to built-in macros shall be interpreted as decimal values. The  
 96725 string values produced as the defining text of the **decr**, **divnum**, **incr**, **index**, **len**, and **sysval**  
 96726 built-in macros shall be in the form of a decimal-constant as defined in the C language.

96727 **changecom** The **changecom** macro shall set the begin-comment and end-comment strings.  
 96728 With no arguments, the comment mechanism shall be disabled. With a single non-  
 96729 null argument, that argument shall become the begin-comment and the <newline>  
 96730 shall become the end-comment string. With two non-null arguments, the first  
 96731 argument shall become the begin-comment string and the second argument shall  
 96732 become the end-comment string. The behavior is unspecified if either argument is  
 96733 provided but null. Systems shall support comment strings of at least five  
 96734 characters.

96735 **changequote** The **changequote** macro shall set the begin-quote and end-quote strings. With no  
 96736 arguments, the quote strings shall be set to the default values (that is, ' '). The  
 96737 behavior is unspecified if there is a single argument or either argument is null.  
 96738 With two non-null arguments, the first argument shall become the begin-quote  
 96739 string and the second argument shall become the end-quote string. Systems shall  
 96740 support quote strings of at least five characters.

96741 **decr** The defining text of the **decr** macro shall be its first argument decremented by 1. It  
 96742 shall be an error to specify an argument containing any non-numeric characters.  
 96743 The behavior is unspecified if **decr** is not immediately followed by a <left-  
 96744 parenthesis>.

96745 **define** The second argument shall become the defining text of the macro whose name is  
 96746 the first argument. It is unspecified whether the **define** macro deletes all prior  
 96747 definitions of the macro named by its first argument or preserves all but the  
 96748 current definition of the macro. The behavior is unspecified if **define** is not  
 96749 immediately followed by a <left-parenthesis>.

96750 **defn** The defining text of the **defn** macro shall be the quoted definition (using the  
 96751 current quoting strings) of its arguments. The behavior is unspecified if **defn** is not  
 96752 immediately followed by a <left-parenthesis>.

96753 **divert** The *m4* utility maintains nine temporary buffers, numbered 1 to 9, inclusive.  
 96754 When the last of the input has been processed, any output that has been placed in  
 96755 these buffers shall be written to standard output in buffer-numerical order. The  
 96756 **divert** macro shall divert future output to the buffer specified by its argument.  
 96757 Specifying no argument or an argument of 0 shall resume the normal output  
 96758 process. Output diverted to a stream with a negative number shall be discarded.  
 96759 Behavior is implementation-defined if a stream number larger than 9 is specified. It  
 96760 shall be an error to specify an argument containing any non-numeric characters.

96761 **divnum** The defining text of the **divnum** macro shall be the number of the current output  
 96762 stream as a string.

96763 **dnl** The **dnl** macro shall cause *m4* to discard all input characters up to and including  
 96764 the next <newline>.



|       |                 |                                                                                                                                                                                                                                                                                                                                   |
|-------|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 96765 | <b>dumpdef</b>  | The <b>dumpdef</b> macro shall write the defined text to standard error for each of the macros specified as arguments, or, if no arguments are specified, for all macros.                                                                                                                                                         |
| 96766 |                 |                                                                                                                                                                                                                                                                                                                                   |
| 96767 | <b>errprint</b> | The <b>errprint</b> macro shall write its arguments to standard error. The behavior is unspecified if <b>errprint</b> is not immediately followed by a <left-parenthesis>.                                                                                                                                                        |
| 96768 |                 |                                                                                                                                                                                                                                                                                                                                   |
| 96769 | <b>eval</b>     | The <b>eval</b> macro shall evaluate its first argument as an arithmetic expression, using signed integer arithmetic with at least 32-bit precision. At least the following C-language operators shall be supported, with precedence, associativity, and behavior as described in <a href="#">Section 1.1.2.1</a> (on page 2321): |
| 96770 |                 |                                                                                                                                                                                                                                                                                                                                   |
| 96771 |                 |                                                                                                                                                                                                                                                                                                                                   |
| 96772 |                 |                                                                                                                                                                                                                                                                                                                                   |
| 96773 |                 | ( )                                                                                                                                                                                                                                                                                                                               |
| 96774 |                 | unary +                                                                                                                                                                                                                                                                                                                           |
| 96775 |                 | unary -                                                                                                                                                                                                                                                                                                                           |
| 96776 |                 | ~                                                                                                                                                                                                                                                                                                                                 |
| 96777 |                 | !                                                                                                                                                                                                                                                                                                                                 |
| 96778 |                 | binary *                                                                                                                                                                                                                                                                                                                          |
| 96779 |                 | /                                                                                                                                                                                                                                                                                                                                 |
| 96780 |                 | %                                                                                                                                                                                                                                                                                                                                 |
| 96781 |                 | binary +                                                                                                                                                                                                                                                                                                                          |
| 96782 |                 | binary -                                                                                                                                                                                                                                                                                                                          |
| 96783 |                 | <<                                                                                                                                                                                                                                                                                                                                |
| 96784 |                 | >>                                                                                                                                                                                                                                                                                                                                |
| 96785 |                 | <                                                                                                                                                                                                                                                                                                                                 |
| 96786 |                 | <=                                                                                                                                                                                                                                                                                                                                |
| 96787 |                 | >                                                                                                                                                                                                                                                                                                                                 |
| 96788 |                 | >=                                                                                                                                                                                                                                                                                                                                |
| 96789 |                 | = =                                                                                                                                                                                                                                                                                                                               |
| 96790 |                 | !=                                                                                                                                                                                                                                                                                                                                |
| 96791 |                 | binary &                                                                                                                                                                                                                                                                                                                          |
| 96792 |                 | ^                                                                                                                                                                                                                                                                                                                                 |
| 96793 |                 |                                                                                                                                                                                                                                                                                                                                   |
| 96794 |                 | &&                                                                                                                                                                                                                                                                                                                                |
| 96795 |                 |                                                                                                                                                                                                                                                                                                                                   |
| 96796 |                 | Systems shall support octal and hexadecimal numbers as in the ISO C standard.                                                                                                                                                                                                                                                     |
| 96797 |                 | The second argument, if specified, shall set the radix for the result; if the argument                                                                                                                                                                                                                                            |
| 96798 |                 | is blank or unspecified, the default is 10. Behavior is unspecified if the radix falls                                                                                                                                                                                                                                            |
| 96799 |                 | outside the range 2 to 36, inclusive. The third argument, if specified, sets the                                                                                                                                                                                                                                                  |
| 96800 |                 | minimum number of digits in the result. Behavior is unspecified if the third                                                                                                                                                                                                                                                      |
| 96801 |                 | argument is less than zero. It shall be an error to specify the second or third                                                                                                                                                                                                                                                   |
| 96802 |                 | argument containing any non-numeric characters. The behavior is unspecified if                                                                                                                                                                                                                                                    |
| 96803 |                 | <b>eval</b> is not immediately followed by a <left-parenthesis>.                                                                                                                                                                                                                                                                  |
| 96804 | <b>ifdef</b>    | If the first argument to the <b>ifdef</b> macro is defined, the defining text shall be the                                                                                                                                                                                                                                        |
| 96805 |                 | second argument. Otherwise, the defining text shall be the third argument, if                                                                                                                                                                                                                                                     |
| 96806 |                 | specified, or the null string, if not. The behavior is unspecified if <b>ifdef</b> is not                                                                                                                                                                                                                                         |
| 96807 |                 | immediately followed by a <left-parenthesis>.                                                                                                                                                                                                                                                                                     |
| 96808 | <b>ifelse</b>   | The <b>ifelse</b> macro takes three or more arguments. If the first two arguments                                                                                                                                                                                                                                                 |
| 96809 |                 | compare as equal strings (after macro expansion of both arguments), the defining                                                                                                                                                                                                                                                  |
| 96810 |                 | text shall be the third argument. If the first two arguments do not compare as equal                                                                                                                                                                                                                                              |
| 96811 |                 | strings and there are three arguments, the defining text shall be null. If the first two                                                                                                                                                                                                                                          |
| 96812 |                 | arguments do not compare as equal strings and there are four or five arguments,                                                                                                                                                                                                                                                   |

|       |                    |                                                                                                    |
|-------|--------------------|----------------------------------------------------------------------------------------------------|
| 96813 |                    | the defining text shall be the fourth argument. If the first two arguments do not                  |
| 96814 |                    | compare as equal strings and there are six or more arguments, the first three                      |
| 96815 |                    | arguments shall be discarded and processing shall restart with the remaining                       |
| 96816 |                    | arguments. The behavior is unspecified if <b>ifelse</b> is not immediately followed by a           |
| 96817 |                    | <left-parenthesis>.                                                                                |
| 96818 | <b>include</b>     | The defining text for the <b>include</b> macro shall be the contents of the file named by          |
| 96819 |                    | the first argument. It shall be an error if the file cannot be read. The behavior is               |
| 96820 |                    | unspecified if <b>include</b> is not immediately followed by a <left-parenthesis>.                 |
| 96821 | <b>incr</b>        | The defining text of the <b>incr</b> macro shall be its first argument incremented by 1. It        |
| 96822 |                    | shall be an error to specify an argument containing any non-numeric characters.                    |
| 96823 |                    | The behavior is unspecified if <b>incr</b> is not immediately followed by a <left-                 |
| 96824 |                    | parenthesis>.                                                                                      |
| 96825 | <b>index</b>       | The defining text of the <b>index</b> macro shall be the first character position (as a            |
| 96826 |                    | string) in the first argument where a string matching the second argument begins                   |
| 96827 |                    | (zero origin), or -1 if the second argument does not occur. The behavior is                        |
| 96828 |                    | unspecified if <b>index</b> is not immediately followed by a <left-parenthesis>.                   |
| 96829 | <b>len</b>         | The defining text of the <b>len</b> macro shall be the length (as a string) of the first           |
| 96830 |                    | argument. The behavior is unspecified if <b>len</b> is not immediately followed by a               |
| 96831 |                    | <left-parenthesis>.                                                                                |
| 96832 | <b>m4exit</b>      | Exit from the <i>m4</i> utility. If the first argument is specified, it shall be the exit code. If |
| 96833 |                    | no argument is specified, the exit code shall be zero. It shall be an error to specify             |
| 96834 |                    | an argument containing any non-numeric characters. If the first argument is zero                   |
| 96835 |                    | or no argument is specified, and an error has previously occurred (for example, a                  |
| 96836 |                    | <i>file</i> operand that could not be opened), it is unspecified whether the exit status is        |
| 96837 |                    | zero or non-zero.                                                                                  |
| 96838 | <b>m4wrap</b>      | The first argument shall be processed when EOF is reached. If the <b>m4wrap</b> macro              |
| 96839 |                    | is used multiple times, the arguments specified shall be processed in the order in                 |
| 96840 |                    | which the <b>m4wrap</b> macros were processed. The behavior is unspecified if <b>m4wrap</b>        |
| 96841 |                    | is not immediately followed by a <left-parenthesis>.                                               |
| 96842 | OB <b>maketemp</b> | The defining text shall be the first argument, with any trailing 'X' characters                    |
| 96843 |                    | replaced with the current process ID as a string. The behavior is unspecified if                   |
| 96844 |                    | <b>maketemp</b> is not immediately followed by a <left-parenthesis>.                               |
| 96845 | <b>mkstemp</b>     | The defining text shall be as if it were the resulting pathname after a successful call            |
| 96846 |                    | to the <i>mkstemp</i> () function defined in the System Interfaces volume of POSIX.1-2008          |
| 96847 |                    | called with the first argument to the macro invocation. If a file is created, that file            |
| 96848 |                    | shall be closed. If a file could not be created, the <i>m4</i> utility shall write a diagnostic    |
| 96849 |                    | message to standard error and shall continue processing input but its final exit                   |
| 96850 |                    | status shall be non-zero; the defining text of the macro shall be the empty string.                |
| 96851 |                    | The behavior is unspecified if <b>mkstemp</b> is not immediately followed by a <left-              |
| 96852 |                    | parenthesis>.                                                                                      |
| 96853 | <b>popdef</b>      | The <b>popdef</b> macro shall delete the current definition of its arguments, replacing            |
| 96854 |                    | that definition with the previous one. If there is no previous definition, the macro               |
| 96855 |                    | is undefined. The behavior is unspecified if <b>popdef</b> is not immediately followed by          |
| 96856 |                    | a <left-parenthesis>.                                                                              |
| 96857 | <b>pushdef</b>     | The <b>pushdef</b> macro shall be equivalent to the <b>define</b> macro with the exception that    |
| 96858 |                    | it shall preserve any current definition for future retrieval using the <b>popdef</b> macro.       |
| 96859 |                    | The behavior is unspecified if <b>pushdef</b> is not immediately followed by a <left-              |

|       |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|-------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 96860 |                 | parenthesis>.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 96861 | <b>shift</b>    | The defining text for the <b>shift</b> macro shall be a comma-separated list of its arguments except the first one. Each argument shall be quoted using the current quoting strings. The behavior is unspecified if <b>shift</b> is not immediately followed by a <left-parenthesis>.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96862 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96863 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96864 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96865 | <b>sinclude</b> | The <b>sinclude</b> macro shall be equivalent to the <b>include</b> macro, except that it shall not be an error if the file is inaccessible. The behavior is unspecified if <b>sinclude</b> is not immediately followed by a <left-parenthesis>.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 96866 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96867 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96868 | <b>substr</b>   | The defining text for the <b>substr</b> macro shall be the substring of the first argument beginning at the zero-offset character position specified by the second argument. The third argument, if specified, shall be the number of characters to select; if not specified, the characters from the starting point to the end of the first argument shall become the defining text. It shall not be an error to specify a starting point beyond the end of the first argument and the defining text shall be null. It shall be an error to specify an argument containing any non-numeric characters. The behavior is unspecified if <b>substr</b> is not immediately followed by a <left-parenthesis>.                                                                            |
| 96869 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96870 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96871 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96872 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96873 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96874 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96875 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96876 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96877 | <b>syscmd</b>   | The <b>syscmd</b> macro shall interpret its first argument as a shell command line. The defining text shall be the string result of that command. The string result shall not be rescanned for macros while setting the defining text. No output redirection shall be performed by the <i>m4</i> utility. The exit status value from the command can be retrieved using the <b>sysval</b> macro. The behavior is unspecified if <b>syscmd</b> is not immediately followed by a <left-parenthesis>.                                                                                                                                                                                                                                                                                   |
| 96878 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96879 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96880 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96881 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96882 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96883 | <b>sysval</b>   | The defining text of the <b>sysval</b> macro shall be the exit value of the utility last invoked by the <b>syscmd</b> macro (as a string).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 96884 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96885 | <b>traceon</b>  | The <b>traceon</b> macro shall enable tracing for the macros specified as arguments, or, if no arguments are specified, for all macros. The trace output shall be written to standard error in an unspecified format.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 96886 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96887 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96888 | <b>traceoff</b> | The <b>traceoff</b> macro shall disable tracing for the macros specified as arguments, or, if no arguments are specified, for all macros.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 96889 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96890 | <b>translit</b> | The defining text of the <b>translit</b> macro shall be the first argument with every character that occurs in the second argument replaced with the corresponding character from the third argument. If no replacement character is specified for some source character because the second argument is longer than the third argument, that character shall be deleted from the first argument in <b>translit</b> 's defining text. The behavior is unspecified if the '-' character appears within the second or third argument anywhere besides the first or last character. The behavior is unspecified if the same character appears more than once in the second argument. The behavior is unspecified if <b>translit</b> is not immediately followed by a <left-parenthesis>. |
| 96891 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96892 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96893 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96894 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96895 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96896 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96897 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96898 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96899 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96900 | <b>undefine</b> | The <b>undefine</b> macro shall delete all definitions (including those preserved using the <b>pushdef</b> macro) of the macros named by its arguments. The behavior is unspecified if <b>undefine</b> is not immediately followed by a <left-parenthesis>.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 96901 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96902 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96903 | <b>undivert</b> | The <b>undivert</b> macro shall cause immediate output of any text in temporary buffers named as arguments, or all temporary buffers if no arguments are specified. Buffers can be undiverted into other temporary buffers. Undiverting shall discard the contents of the temporary buffer. The behavior is unspecified if an argument                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 96904 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96905 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 96906 |                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |

96907 contains any non-numeric characters.

## 96908 EXIT STATUS

96909 The following exit values shall be returned:

96910 0 Successful completion.

96911 >0 An error occurred

96912 If the **m4exit** macro is used, the exit value can be specified by the input file.

## 96913 CONSEQUENCES OF ERRORS

96914 Default.

## 96915 APPLICATION USAGE

96916 The **defn** macro is useful for renaming macros, especially built-ins.

96917 Since **eval** defers to the ISO C standard, some operations have undefined behavior. In some  
 96918 implementations, division or remainder by zero cause a fatal signal, even if the division occurs  
 96919 on the short-circuited branch of "&&" or "||". Any operation that overflows in signed  
 96920 arithmetic produces undefined behavior. Likewise, using the **shift** operators with a shift amount  
 96921 that is not positive and smaller than the precision is undefined, as is shifting a negative number  
 96922 to the right. Historically, not all implementations obeyed C-language precedence rules: '~' and  
 96923 '!' were lower than '=='; '==' and '!=' were not lower than '<'; and '|' was not lower  
 96924 than '^'; the liberal use of "(" can force the desired precedence even with these non-  
 96925 compliant implementations. Furthermore, some traditional implementations treated '^' as an  
 96926 exponentiation operator, although most implementations now use "\*\*\*" as an extension for this  
 96927 purpose.

96928 When a macro has been multiply defined via the **pushdef** macro, it is unspecified whether the  
 96929 **define** macro will alter only the most recent definition (as though by **popdef** and **pushdef**), or  
 96930 replace the entire stack of definitions with a single definition (as though by **undefine** and  
 96931 **pushdef**). An application desiring particular behavior for the **define** macro in this case can  
 96932 redefine it accordingly.

96933 Applications should use the **mkstemp** macro instead of the obsolescent **maketemp** macro for  
 96934 creating temporary files.

## 96935 EXAMPLES

96936 If the file **m4src** contains the lines:

```
96937 The value of 'VER' is "VER".
96938 ifdef('VER', ``VER'' is defined to be VER., VER is not defined.)
96939 ifelse(VER, 1, ``VER'' is 'VER'.)
96940 ifelse(VER, 2, ``VER'' is 'VER'., ``VER'' is not 2.)
96941 end
```

96942 then the command

96943 m4 m4src

96944 or the command:

96945 m4 -U VER m4src

96946 produces the output:

```
96947 The value of VER is "VER".
96948 VER is not defined.
96949 VER is not 2.
```

96950           end  
96951           The command:  
96952           m4 -D VER m4src  
96953           produces the output:  
96954           The value of VER is "".  
96955           VER is defined to be .  
96956           VER is not 2.  
96957           end  
96958           The command:  
96959           m4 -D VER=1 m4src  
96960           produces the output:  
96961           The value of VER is "1".  
96962           VER is defined to be 1.  
96963           VER is 1.  
96964           VER is not 2.  
96965           end  
96966           The command:  
96967           m4 -D VER=2 m4src  
96968           produces the output:  
96969           The value of VER is "2".  
96970           VER is defined to be 2.  
96971           VER is 2.  
96972           end

#### 96973 **RATIONALE**

96974           Historic System V-based behavior treated "\${" in a macro definition as two literal characters.  
96975           However, this sequence is left unspecified so that implementations may offer extensions such as  
96976           "\${11}" meaning the eleventh positional parameter. Macros can still be defined with  
96977           appropriate uses of nested quoting to result in a literal "\${" in the output after rescanning  
96978           removes the nested quotes.

96979           In the **translit** built-in, historic System V-based behavior treated '–' as a literal; GNU behavior  
96980           treats it as a range. This version of the standard allows either behavior.

#### 96981 **FUTURE DIRECTIONS**

96982           None.

#### 96983 **SEE ALSO**

96984           c99

96985           XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

#### 96986 **CHANGE HISTORY**

96987           First released in Issue 2.

**Issue 5**

The phrase “the defined text for macros written by the **dumpdef** macro” is added to the description of **STDERR**, and the description of **dumpdef** is updated to indicate that output is written to standard error. The description of **eval** is updated to indicate that the list of excluded C operators excludes unary **'&'** and **'.'**. In the description of **ifdef**, the phrase “and it is not defined to be zero” is deleted.

**Issue 6**

In the EXTENDED DESCRIPTION, the **eval** text is updated to include a **'&'** character in the excepted list.

The EXTENDED DESCRIPTION of **divert** is updated to clarify that there are only nine diversion buffers.

The normative text is reworded to avoid use of the term “must” for application requirements.

The Open Group Base Resolution bwg2000-006 is applied.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/31 is applied, replacing the EXAMPLES section.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #117 is applied, marking the **maketemp** macro obsolescent and adding a new **mkstemp** macro.

Austin Group Interpretation 1003.1-2001 #207 is applied, clarifying the handling of white-space characters that precede or trail any macro arguments.

SD5-XCU-ERN-6 is applied, clarifying that Guideline 9 of the Utility Syntax Guidelines does not apply (options can be interspersed with operands).

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

SD5-XCU-ERN-99 is applied, clarifying the definition of the **divert** macro in the EXTENDED DESCRIPTION.

SD5-XCU-ERN-100 is applied, clarifying the definition of the **syscmd** macro in the EXTENDED DESCRIPTION.

SD5-XCU-ERN-101 is applied, clarifying the definition of the **undivert** macro in the EXTENDED DESCRIPTION.

SD5-XCU-ERN-111 is applied to the EXTENDED DESCRIPTION, clarifying that the string **"\${"** produces unspecified behavior.

SD5-XCU-ERN-112 is applied, updating the **changequote** macro.

SD5-XCU-ERN-118 is applied, clarifying the definition of the **define** macro in the EXTENDED DESCRIPTION and APPLICATION USAGE sections.

SD5-XCU-ERN-119 is applied, clarifying the definition of the **translit** macro in the EXTENDED DESCRIPTION and RATIONALE sections.

SD5-XCU-ERN-130, SD5-XCU-ERN-131, and SD5-XCU-ERN-137 are applied.

The **m4** utility is moved from the XSI option to the Base.

97026  
97027

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0117 [241], XCU/TC1-2008/0118 [242,431], XCU/TC1-2008/0119 [242,431], and XCU/TC1-2008/0120 [325,430] are applied.



97028 **NAME**

97029 mailx — process messages

97030 **SYNOPSIS**97031 **Send Mode**97032 mailx [-s *subject*] *address*...97033 **Receive Mode**

97034 UP mailx -e

97035 mailx [-HiNn] [-F] [-u *user*]97036 mailx -f [-HiNn] [-F] [*file*]97037 **DESCRIPTION**

97038 The *mailx* utility provides a message sending and receiving facility. It has two major modes,  
97039 selected by the options used: Send Mode and Receive Mode.

97040 On systems that do not support the User Portability Utilities option, an application using *mailx*  
97041 shall have the ability to send messages in an unspecified manner (Send Mode). Unless the first  
97042 character of one or more lines is <tilde> ( '~ '), all characters in the input message shall appear in  
97043 the delivered message, but additional characters may be inserted in the message before it is  
97044 retrieved.

97045 UP On systems supporting the User Portability Utilities option, mail-receiving capabilities and other  
97046 interactive features, Receive Mode, described below, also shall be enabled.

97047 **Send Mode**

97048 Send Mode can be used by applications or users to send messages from the text in standard  
97049 input.

97050 UP **Receive Mode**

97051 Receive Mode is more oriented towards interactive users. Mail can be read and sent in this  
97052 interactive mode.

97053 When reading mail, *mailx* provides commands to facilitate saving, deleting, and responding to  
97054 messages. When sending mail, *mailx* allows editing, reviewing, and other modification of the  
97055 message as it is entered.

97056 Incoming mail shall be stored in one or more unspecified locations for each user, collectively  
97057 called the system *mailbox* for that user. When *mailx* is invoked in Receive Mode, the system  
97058 mailbox shall be the default place to find new mail. As messages are read, they shall be marked  
97059 to be moved to a secondary file for storage, unless specific action is taken. This secondary file is  
97060 called the **mbox** and is normally located in the directory referred to by the *HOME* environment  
97061 variable (see *MBOX* in the ENVIRONMENT VARIABLES section for a description of this file).  
97062 Messages shall remain in this file until explicitly removed. When the -f option is used to read  
97063 mail messages from secondary files, messages shall be retained in those files unless specifically  
97064 removed. All three of these locations—system mailbox, **mbox**, and secondary file—are referred  
97065 to in this section as simply “mailboxes”, unless more specific identification is required.



## OPTIONS

The *mailx* utility shall conform to XBD [Section 12.2](#) (on page 216).

The following options shall be supported. (Only the `-s subject` option shall be required on all systems. The other options are required only on systems supporting the User Portability Utilities option.)

**-e** Test for the presence of mail in the system mailbox. The *mailx* utility shall write nothing and exit with a successful return code if there is mail to read.

**-f** Read messages from the file named by the *file* operand instead of the system mailbox. (See also **folder**.) If no *file* operand is specified, read messages from **mbox** instead of the system mailbox.

**-F** Record the message in a file named after the first recipient. The name is the login-name portion of the address found first on the **To:** line in the mail header. Overrides the **record** variable, if set (see [Internal Variables in mailx](#), on page 2939).

**-H** Write a header summary only.

**-i** Ignore interrupts. (See also **ignore**.)

**-n** Do not initialize from the system default start-up file. See the EXTENDED DESCRIPTION section.

**-N** Do not write an initial header summary.

**-s subject** Set the **Subject** header field to *subject*. All characters in the *subject* string shall appear in the delivered message. The results are unspecified if *subject* is longer than {LINE\_MAX} – 10 bytes or contains a <newline>.

**-u user** Read the system mailbox of the login name *user*. This shall only be successful if the invoking user has appropriate privileges to read the system mailbox of that user.

## OPERANDS

The following operands shall be supported:

*address* Addressee of message. When **-n** is specified and no user start-up files are accessed (see the EXTENDED DESCRIPTION section), the user or application shall ensure this is an address to pass to the mail delivery system. Any system or user start-up files may enable aliases (see **alias** under [Commands in mailx](#), on page 2942) that may modify the form of *address* before it is passed to the mail delivery system.

*file* A pathname of a file to be read instead of the system mailbox when **-f** is specified. The meaning of the *file* option-argument shall be affected by the contents of the **folder** internal variable; see [Internal Variables in mailx](#) (on page 2939).

## STDIN

When *mailx* is invoked in Send Mode (the first synopsis line), standard input shall be the message to be delivered to the specified addresses. When in Receive Mode, user commands shall be accepted from *stdin*. If the User Portability Utilities option is not supported, standard input lines beginning with a <tilde> ('~') character produce unspecified results.

If the User Portability Utilities option is supported, then in both Send and Receive Modes, standard input lines beginning with the escape character (usually <tilde> ('~')) shall affect processing as described in [Command Escapes in mailx](#) (on page 2951).

## 97108 INPUT FILES

97109 When *mailx* is used as described by this volume of POSIX.1-2008, the *file* option-argument (see  
 97110 the **-f** option) and the **mbox** shall be text files containing mail messages, formatted as described  
 97111 in the OUTPUT FILES section. The nature of the system mailbox is unspecified; it need not be a  
 97112 file.

## 97113 ENVIRONMENT VARIABLES

97114 UP Some of the functionality described in this section shall be provided on implementations that  
 97115 support the User Portability Utilities option as described in the text, and is not further shaded  
 97116 for this option.

97117 The following environment variables shall affect the execution of *mailx*:

97118 **DEAD** Determine the pathname of the file in which to save partial messages in case of  
 97119 interrupts or delivery errors. The default shall be **dead.letter** in the directory  
 97120 named by the *HOME* variable. The behavior of *mailx* in saving partial messages is  
 97121 unspecified if the User Portability Utilities option is not supported and *DEAD* is  
 97122 not defined with the value **/dev/null**.

97123 **EDITOR** Determine the name of a utility to invoke when the **edit** (see [Commands in mailx](#),  
 97124 on page 2942) or **~e** (see [Command Escapes in mailx](#), on page 2951) command is  
 97125 XSI used. The default editor is unspecified. [On XSI-conformant systems it is \*ed\*.](#) The  
 97126 effects of this variable are unspecified if the User Portability Utilities option is not  
 97127 supported.

97128 **HOME** Determine the pathname of the user's home directory.

97129 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 97130 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 97131 variables used to determine the values of locale categories.)

97132 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 97133 internationalization variables.

97134 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 97135 characters (for example, single-byte as opposed to multi-byte characters in  
 97136 arguments and input files) and the handling of case-insensitive address and  
 97137 header-field comparisons.

97138 **LC\_TIME** This variable may determine the format and contents of the date and time strings  
 97139 written by *mailx*. This volume of POSIX.1-2008 specifies the effects of this variable  
 97140 only for systems supporting the User Portability Utilities option.

## 97141 LC\_MESSAGES

97142 Determine the locale that should be used to affect the format and contents of  
 97143 diagnostic messages written to standard error and informative messages written to  
 97144 standard output.

97145 **LISTER** Determine a string representing the command for writing the contents of the  
 97146 **folder** directory to standard output when the **folders** command is given (see  
 97147 **folders** in [Commands in mailx](#), on page 2942). Any string acceptable as a  
 97148 *command\_string* operand to the *sh -c* command shall be valid. If this variable is null  
 97149 or not set, the output command shall be *ls*. The effects of this variable are  
 97150 unspecified if the User Portability Utilities option is not supported.

97151 **MAILRC** Determine the pathname of the user start-up file. The default shall be **.mailrc** in the  
 97152 directory referred to by the *HOME* environment variable. The behavior of *mailx* is  
 97153 unspecified if the User Portability Utilities option is not supported and *MAILRC* is

|       |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|-------|----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 97154 |                            | not defined with the value <code>/dev/null</code> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 97155 | <i>MBOX</i>                | Determine a pathname of the file to save messages from the system mailbox that have been read. The <b>exit</b> command shall override this function, as shall saving the message explicitly in another file. The default shall be <b>mbox</b> in the directory named by the <i>HOME</i> variable. The effects of this variable are unspecified if the User Portability Utilities option is not supported.                                                                                                                                                                                                                                                                                                                                                                    |
| 97156 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97157 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97158 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97159 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97160 | XSI                        | <i>NLSPATH</i> Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 97161 | <i>PAGER</i>               | Determine a string representing an output filtering or pagination command for writing the output to the terminal. Any string acceptable as a <i>command_string</i> operand to the <i>sh -c</i> command shall be valid. When standard output is a terminal device, the message output shall be piped through the command if the <i>mailx</i> internal variable <b>cr</b> t is set to a value less the number of lines in the message; see <a href="#">Internal Variables in mailx</a> (on page 2939). If the <i>PAGER</i> variable is null or not set, the paginator shall be either <i>more</i> or another paginator utility documented in the system documentation. The effects of this variable are unspecified if the User Portability Utilities option is not supported. |
| 97162 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97163 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97164 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97165 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97166 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97167 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97168 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97169 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97170 | <i>SHELL</i>               | Determine the name of a preferred command interpreter. The default shall be <i>sh</i> . The effects of this variable are unspecified if the User Portability Utilities option is not supported.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97171 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97172 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97173 | <i>TERM</i>                | If the internal variable <b>screen</b> is not specified, determine the name of the terminal type to indicate in an unspecified manner the number of lines in a screenful of headers. If <i>TERM</i> is not set or is set to null, an unspecified default terminal type shall be used and the value of a screenful is unspecified. The effects of this variable are unspecified if the User Portability Utilities option is not supported.                                                                                                                                                                                                                                                                                                                                    |
| 97174 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97175 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97176 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97177 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97178 | <i>TZ</i>                  | This variable may determine the timezone used to calculate date and time strings written by <i>mailx</i> . If <i>TZ</i> is unset or null, an unspecified default timezone shall be used.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 97179 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97180 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97181 | <i>VISUAL</i>              | Determine a pathname of a utility to invoke when the <b>visual</b> command (see <a href="#">Commands in mailx</a> , on page 2942) or <b>~v</b> command-escape (see <a href="#">Command Escapes in mailx</a> , on page 2951) is used. If this variable is null or not set, the full-screen editor shall be <i>vi</i> . The effects of this variable are unspecified if the User Portability Utilities option is not supported.                                                                                                                                                                                                                                                                                                                                                |
| 97182 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97183 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97184 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97185 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97186 | <b>ASYNCHRONOUS EVENTS</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97187 |                            | When <i>mailx</i> is in Send Mode and standard input is not a terminal, it shall take the standard action for all signals.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 97188 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97189 | UP                         | In <b>Receive Mode</b> , or in Send Mode when standard input is a terminal, if a SIGINT signal is received:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 97190 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97191 | UP                         | 1. If in command mode, the current command, if there is one, shall be aborted, and a command-mode prompt shall be written.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 97192 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97193 |                            | 2. If in input mode:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 97194 | UP                         | a. If <b>ignore</b> is set, <i>mailx</i> shall write "@\n", discard the current input line, and continue processing, bypassing the message-abort mechanism described in item 2b.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 97195 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 97196 |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

- 97197 UP                   b. If the interrupt was received while sending mail, either when in **Receive Mode** or  
 97198 **in Send Mode**, a message shall be written, and another subsequent interrupt, with  
 97199 no other intervening characters typed, shall be required to abort the mail message.  
 97200 UP                   If in **Receive Mode** and another interrupt is received, a command-mode prompt  
 97201 shall be written. If in **Send Mode** and another interrupt is received, *mailx* shall  
 97202 terminate with a non-zero status.

97203                   In both cases listed in item b, if the message is not empty:

- 97204 UP                   i. If **save** is enabled and the file named by *DEAD* can be created, the message  
 97205 shall be written to the file named by *DEAD*. If the file exists, the message  
 97206 shall be written to replace the contents of the file.
- 97207 UP                   ii. If **save** is not enabled, or the file named by *DEAD* cannot be created, the  
 97208 message shall not be saved.

97209                   The *mailx* utility shall take the standard action for all other signals.

## 97210 **STDOUT**

97211                   In command and input modes, all output, including prompts and messages, shall be written to  
 97212 standard output.

## 97213 **STDERR**

97214                   The standard error shall be used only for diagnostic messages.

## 97215 **OUTPUT FILES**

97216                   Various *mailx* commands and command escapes can create or add to files, including the **mbox**,  
 97217 the dead-letter file, and secondary mailboxes. When *mailx* is used as described in this volume of  
 97218 POSIX.1-2008, these files shall be text files, formatted as follows:

97219                   line beginning with **From<space>**  
 97220                   [one or more *header-lines*; see [Commands in mailx](#) (on page 2942)]  
 97221                   empty line  
 97222                   [zero or more *body lines*  
 97223                   empty line]  
 97224                   [line beginning with **From<space>...**]

97225                   where each message begins with the **From <space>** line shown, preceded by the beginning of  
 97226 the file or an empty line. (The **From <space>** line is considered to be part of the message header,  
 97227 but not one of the header-lines referred to in [Commands in mailx](#) (on page 2942); thus, it shall  
 97228 not be affected by the **discard**, **ignore**, or **retain** commands.) The formats of the remainder of the  
 97229 **From <space>** line and any additional header lines are unspecified, except that none shall be  
 97230 empty. The format of a message body line is also unspecified, except that no line following an  
 97231 empty line shall start with **From <space>**; *mailx* shall modify any such user-entered message  
 97232 body lines (following an empty line and beginning with **From <space>**) by adding one or more  
 97233 characters to precede the 'F'; it may add these characters to **From <space>** lines that are not  
 97234 preceded by an empty line.

97235                   When a message from the system mailbox or entered by the user is not a text file, it is  
 97236 implementation-defined how such a message is stored in files written by *mailx*.

## 97237 **EXTENDED DESCRIPTION**

97238 UP                   The functionality in the entire EXTENDED DESCRIPTION section shall be provided on  
 97239 implementations supporting the User Portability Utilities option. The functionality described in  
 97240 this section shall be provided on implementations that support the User Portability Utilities  
 97241 option (and the rest of this section is not further shaded for this option).

The *mailx* utility need not support for all character encodings in all circumstances. For example, inter-system mail may be restricted to 7-bit data by the underlying network, 8-bit data need not be portable to non-internationalized systems, and so on. Under these circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991 standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used.

When *mailx* is invoked using one of the Receive Mode synopsis forms, it shall write a page of header-summary lines (if `-N` was not specified and there are messages, see below), followed by a prompt indicating that *mailx* can accept regular commands (see [Commands in mailx](#), on page 2942); this is termed *command mode*. The page of header-summary lines shall contain the first new message if there are new messages, or the first unread message if there are unread messages, or the first message. When *mailx* is invoked using the Send Mode synopsis and standard input is a terminal, if no subject is specified on the command line and the **asksub** variable is set, a prompt for the subject shall be written. At this point, *mailx* shall be in input mode. This input mode shall also be entered when using one of the Receive Mode synopsis forms and a reply or new message is composed using the **reply**, **Reply**, **followup**, **Followup**, or **mail** commands and standard input is a terminal. When the message is typed and the end of the message is encountered, the message shall be passed to the mail delivery software. Commands can be entered by beginning a line with the escape character (by default, `<tilde>` (`'~'`)) followed by a single command letter and optional arguments. See [Commands in mailx](#) (on page 2942) for a summary of these commands. It is unspecified what effect these commands will have if standard input is not a terminal when a message is entered using either the Send Mode synopsis, or the Read Mode commands **reply**, **Reply**, **followup**, **Followup**, or **mail**.

**Note:** For notational convenience, this section uses the default escape character, `<tilde>`, in all references and examples.

At any time, the behavior of *mailx* shall be governed by a set of environmental and internal variables. These are flags and valued parameters that can be set and cleared via the *mailx* **set** and **unset** commands.

Regular commands are of the form:

`[command] [msglist] [argument ...]`

If no *command* is specified in command mode, **next** shall be assumed. In input mode, commands shall be recognized by the escape character, and lines not treated as commands shall be taken as input for the message.

In command mode, each message shall be assigned a sequential number, starting with 1.

All messages have a state that shall affect how they are displayed in the header summary and how they are retained or deleted upon termination of *mailx*. There is at any time the notion of a *current* message, which shall be marked by a `'>'` at the beginning of a line in the header summary. When *mailx* is invoked using one of the Receive Mode synopsis forms, the current message shall be the first new message, if there is a new message, or the first unread message if there is an unread message, or the first message if there are any messages, or unspecified if there are no messages in the mailbox. Each command that takes an optional list of messages (*msglist*) or an optional single message (*message*) on which to operate shall leave the current message set to the highest-numbered message of the messages specified, unless the command deletes messages, in which case the current message shall be set to the first undeleted message (that is, a message not in the deleted state) after the highest-numbered message deleted by the command, if one exists, or the first undeleted message before the highest-numbered message deleted by the command, if one exists, or to an unspecified value if there are no remaining undeleted messages. All messages shall be in one of the following states:



|       |                  |                                                                                                         |
|-------|------------------|---------------------------------------------------------------------------------------------------------|
| 97289 | <i>new</i>       | The message is present in the system mailbox and has not been viewed by the user                        |
| 97290 |                  | or moved to any other state. Messages in state <i>new</i> when <i>mailx</i> quits shall be              |
| 97291 |                  | retained in the system mailbox.                                                                         |
| 97292 | <i>unread</i>    | The message has been present in the system mailbox for more than one invocation                         |
| 97293 |                  | of <i>mailx</i> and has not been viewed by the user or moved to any other state.                        |
| 97294 |                  | Messages in state <i>unread</i> when <i>mailx</i> quits shall be retained in the system mailbox.        |
| 97295 | <i>read</i>      | The message has been processed by one of the following commands: <b>~f, ~m, ~F, ~M,</b>                 |
| 97296 |                  | <b>copy, mbox, next, pipe, print, Print, top, type, Type, undelete.</b> The <b>delete, dp,</b>          |
| 97297 |                  | and <b>dt</b> commands may also cause the next message to be marked as <i>read</i> ,                    |
| 97298 |                  | depending on the value of the <b>autoprint</b> variable. Messages that are in the system                |
| 97299 |                  | mailbox and in state <i>read</i> when <i>mailx</i> quits shall be saved in the <b>mbox</b> , unless the |
| 97300 |                  | internal variable <b>hold</b> was set. Messages that are in the <b>mbox</b> or in a secondary           |
| 97301 |                  | mailbox and in state <i>read</i> when <i>mailx</i> quits shall be retained in their current             |
| 97302 |                  | location.                                                                                               |
| 97303 | <i>deleted</i>   | The message has been processed by one of the following commands: <b>delete, dp, dt.</b>                 |
| 97304 |                  | Messages in state <i>deleted</i> when <i>mailx</i> quits shall be deleted. Deleted messages shall       |
| 97305 |                  | be ignored until <i>mailx</i> quits or changes mailboxes or they are specified to the                   |
| 97306 |                  | undelete command; for example, the message specification <i>/string</i> shall only                      |
| 97307 |                  | search the subject lines of messages that have not yet been deleted, unless the                         |
| 97308 |                  | command operating on the list of messages is <b>undelete</b> . No deleted message or                    |
| 97309 |                  | deleted message header shall be displayed by any <i>mailx</i> command other than                        |
| 97310 |                  | <b>undelete</b> .                                                                                       |
| 97311 | <i>preserved</i> | The message has been processed by a <b>preserve</b> command. When <i>mailx</i> quits, the               |
| 97312 |                  | message shall be retained in its current location.                                                      |
| 97313 | <i>saved</i>     | The message has been processed by one of the following commands: <b>save</b> or <b>write</b> .          |
| 97314 |                  | If the current mailbox is the system mailbox, and the internal variable <b>keepsave</b> is              |
| 97315 |                  | set, messages in the state <i>saved</i> shall be saved to the file designated by the <i>MBOX</i>        |
| 97316 |                  | variable (see the ENVIRONMENT VARIABLES section). If the current mailbox is                             |
| 97317 |                  | the system mailbox, messages in the state <i>saved</i> shall be deleted from the current                |
| 97318 |                  | mailbox, when the <b>quit</b> or <b>file</b> command is used to exit the current mailbox.               |
| 97319 |                  | The header-summary line for each message shall indicate the state of the message.                       |
| 97320 |                  | Many commands take an optional list of messages ( <i>msglist</i> ) on which to operate, which defaults  |
| 97321 |                  | to the current message. A <i>msglist</i> is a list of message specifications separated by <blank>       |
| 97322 |                  | characters, which can include:                                                                          |
| 97323 | <i>n</i>         | Message number <i>n</i> .                                                                               |
| 97324 | <b>+</b>         | The next undeleted message, or the next deleted message for the <b>undelete</b> command.                |
| 97325 | <b>–</b>         | The next previous undeleted message, or the next previous deleted message for the                       |
| 97326 |                  | <b>undelete</b> command.                                                                                |
| 97327 | <b>.</b>         | The current message.                                                                                    |
| 97328 | <b>^</b>         | The first undeleted message, or the first deleted message for the <b>undelete</b> command.              |
| 97329 | <b>\$</b>        | The last message.                                                                                       |
| 97330 | <b>*</b>         | All messages.                                                                                           |

|       |         |                                                                                      |
|-------|---------|--------------------------------------------------------------------------------------|
| 97331 | n-m     | An inclusive range of message numbers.                                               |
| 97332 | address | All messages from <i>address</i> ; any address as shown in a header summary shall be |
| 97333 |         | matchable in this form.                                                              |
| 97334 | /string | All messages with <i>string</i> in the subject line (case ignored).                  |
| 97335 | :c      | All messages of type <i>c</i> , where <i>c</i> shall be one of:                      |
| 97336 | d       | Deleted messages.                                                                    |
| 97337 | n       | New messages.                                                                        |
| 97338 | o       | Old messages (any not in state <i>read</i> or <i>new</i> ).                          |
| 97339 | r       | Read messages.                                                                       |
| 97340 | u       | Unread messages.                                                                     |

97341 Other commands take an optional message (*message*) on which to operate, which defaults to the  
97342 current message. All of the forms allowed for *msglist* are also allowed for *message*, but if more  
97343 than one message is specified, only the first shall be operated on.

97344 Other arguments are usually arbitrary strings whose usage depends on the command involved.

97345 **Start-Up in mailx**

97346 At start-up time, *mailx* shall take the following steps in sequence:

1. Establish all variables at their stated default values.
2. Process command line options, overriding corresponding default values.
3. Import any of the *DEAD*, *EDITOR*, *MBOX*, *LISTER*, *PAGER*, *SHELL*, or *VISUAL* variables that are present in the environment, overriding the corresponding default values.
4. Read *mailx* commands from an unspecified system start-up file, unless the **-n** option is given, to initialize any internal *mailx* variables and aliases.
5. Process the user start-up file of *mailx* commands named in the user *MAILRC* variable.

97354 Most regular *mailx* commands are valid inside start-up files, the most common use being to set  
97355 up initial display options and alias lists. The following commands shall be invalid in a start-up  
97356 file: **!**, **edit**, **hold**, **mail**, **preserve**, **reply**, **Reply**, **shell**, **visual**, **Copy**, **followup**, and **Followup**.  
97357 Any errors in a start-up file shall either cause *mailx* to terminate with a diagnostic message and a  
97358 non-zero status or to continue after writing a diagnostic message, ignoring the remainder of the  
97359 lines in the file.

97360 A blank line in a start-up file shall be ignored.

## 97361 Internal Variables in mailx

97362 The following variables are internal *mailx* variables. Each internal variable can be set via the  
97363 *mailx* **set** command at any time. The **unset** and **set no name** commands can be used to erase  
97364 variables.

97365 In the following list, variables shown as:

```
97366      variable
```

97367 represent Boolean values. Variables shown as:

```
97368      variable=value
```

|       |                         |                                                                                                               |
|-------|-------------------------|---------------------------------------------------------------------------------------------------------------|
| 97369 |                         | shall be assigned string or numeric values. For string values, the rules in <a href="#">Commands in mailx</a> |
| 97370 |                         | (on page 2942) concerning filenames and quoting shall also apply.                                             |
| 97371 |                         | The defaults specified here may be changed by the unspecified system start-up file unless the                 |
| 97372 |                         | user specifies the <b>-n</b> option.                                                                          |
| 97373 | <b>allnet</b>           | All network names whose login name components match shall be treated as                                       |
| 97374 |                         | identical. This shall cause the <i>msglist</i> message specifications to behave similarly.                    |
| 97375 |                         | The default shall be <b>noallnet</b> . See also the <b>alternates</b> command and the <b>metoo</b>            |
| 97376 |                         | variable.                                                                                                     |
| 97377 | <b>append</b>           | Append messages to the end of the <b>mbox</b> file upon termination instead of placing                        |
| 97378 |                         | them at the beginning. The default shall be <b>noappend</b> . This variable shall not                         |
| 97379 |                         | affect the <b>save</b> command when saving to <b>mbox</b> .                                                   |
| 97380 | <b>ask, asksub</b>      | Prompt for a subject line on outgoing mail if one is not specified on the command                             |
| 97381 |                         | line with the <b>-s</b> option. The <b>ask</b> and <b>asksub</b> forms are synonyms; the system shall         |
| 97382 |                         | refer to <b>asksub</b> and <b>noasksub</b> in its messages, but shall accept <b>ask</b> and <b>noask</b> as   |
| 97383 |                         | user input to mean <b>asksub</b> and <b>noasksub</b> . It shall not be possible to set both <b>ask</b>        |
| 97384 |                         | and <b>noasksub</b> , or <b>noask</b> and <b>asksub</b> . The default shall be <b>asksub</b> , but no         |
| 97385 |                         | prompting shall be done if standard input is not a terminal.                                                  |
| 97386 | <b>askbcc</b>           | Prompt for the blind copy list. The default shall be <b>noaskbcc</b> .                                        |
| 97387 | <b>askcc</b>            | Prompt for the copy list. The default shall be <b>noaskcc</b> .                                               |
| 97388 | <b>autoprint</b>        | Enable automatic writing of messages after <b>delete</b> and <b>undelete</b> commands. The                    |
| 97389 |                         | default shall be <b>noautoprint</b> .                                                                         |
| 97390 | <b>bang</b>             | Enable the special-case treatment of <exclamation-mark> characters ('!') in                                   |
| 97391 |                         | escape command lines; see the <b>escape</b> command and <a href="#">Command Escapes in mailx</a>              |
| 97392 |                         | (on page 2951). The default shall be <b>nobang</b> , disabling the expansion of '!' in the                    |
| 97393 |                         | <i>command</i> argument to the <b>~!</b> command and the <b>~!command</b> escape.                             |
| 97394 | <b>cmd=command</b>      |                                                                                                               |
| 97395 |                         | Set the default command to be invoked by the <b>pipe</b> command. The default shall be                        |
| 97396 |                         | <b>nocmd</b> .                                                                                                |
| 97397 | <b>crt=number</b>       | Pipe messages having more than <i>number</i> lines through the command specified by                           |
| 97398 |                         | the value of the <i>PAGER</i> variable. The default shall be <b>nocrt</b> . If it is set to null, the         |
| 97399 |                         | value used is implementation-defined.                                                                         |
| 97400 | XSI <b>debug</b>        | Enable verbose diagnostics for debugging. Messages are not delivered. The                                     |
| 97401 |                         | default shall be <b>nodebug</b> .                                                                             |
| 97402 | <b>dot</b>              | When <b>dot</b> is set, a <period> on a line by itself during message input from a                            |
| 97403 |                         | terminal shall also signify end-of-file (in addition to normal end-of-file). The                              |
| 97404 |                         | default shall be <b>nodot</b> . If <b>ignoreeof</b> is set (see below), a setting of <b>nodot</b> shall be    |
| 97405 |                         | ignored and the <period> is the only method to terminate input mode.                                          |
| 97406 | <b>escape=c</b>         | Set the command escape character to be the character 'c'. By default, the                                     |
| 97407 |                         | command escape character shall be <tilde>. If <b>escape</b> is unset, <tilde> shall be                        |
| 97408 |                         | used; if it is set to null, command escaping shall be disabled.                                               |
| 97409 | <b>flipr</b>            | Reverse the meanings of the <b>R</b> and <b>r</b> commands. The default shall be <b>noflipr</b> .             |
| 97410 | <b>folder=directory</b> |                                                                                                               |
| 97411 |                         | The default directory for saving mail files. User-specified filenames beginning with                          |
| 97412 |                         | a <plus-sign> ('+') shall be expanded by preceding the filename with this                                     |



|       |                            |                                                                                                          |
|-------|----------------------------|----------------------------------------------------------------------------------------------------------|
| 97413 |                            | directory name to obtain the real pathname. If <i>directory</i> does not start with a                    |
| 97414 |                            | <slash> ( ' / ' ), the contents of <i>HOME</i> shall be prefixed to it. The default shall be             |
| 97415 |                            | <b>nofolder</b> . If <b>folder</b> is unset or set to null, user-specified filenames beginning with      |
| 97416 |                            | '+' shall refer to files in the current directory that begin with the literal '+'                        |
| 97417 |                            | character. See also <b>outfolder</b> below. The <b>folder</b> value need not affect the processing       |
| 97418 |                            | of the files named in <i>MBOX</i> and <i>DEAD</i> .                                                      |
| 97419 | <b>header</b>              | Enable writing of the header summary when entering <i>mailx</i> in Receive Mode. The                     |
| 97420 |                            | default shall be <b>header</b> .                                                                         |
| 97421 | <b>hold</b>                | Preserve all messages that are read in the system mailbox instead of putting them                        |
| 97422 |                            | in the <b>mbox</b> save file. The default shall be <b>nohold</b> .                                       |
| 97423 | <b>ignore</b>              | Ignore interrupts while entering messages. The default shall be <b>noignore</b> .                        |
| 97424 | <b>ignoreeof</b>           | Ignore normal end-of-file during message input. Input can be terminated only by                          |
| 97425 |                            | entering a <period> ( ' . ' ) on a line by itself or by the ~. command escape. The                       |
| 97426 |                            | default shall be <b>noignoreeof</b> . See also <b>dot</b> above.                                         |
| 97427 | <b>indentprefix=string</b> |                                                                                                          |
| 97428 |                            | A string that shall be added as a prefix to each line that is inserted into the message                  |
| 97429 |                            | by the ~m command escape. This variable shall default to one <tab>.                                      |
| 97430 | <b>keep</b>                | When a system mailbox, secondary mailbox, or <b>mbox</b> is empty, truncate it to zero                   |
| 97431 |                            | length instead of removing it. The default shall be <b>nokeep</b> .                                      |
| 97432 | <b>keepsave</b>            | Keep the messages that have been saved from the system mailbox into other files                          |
| 97433 |                            | in the file designated by the variable <i>MBOX</i> , instead of deleting them. The default               |
| 97434 |                            | shall be <b>nokeepsave</b> .                                                                             |
| 97435 | <b>metoo</b>               | Suppress the deletion of the login name of the user from the recipient list when                         |
| 97436 |                            | replying to a message or sending to a group. The default shall be <b>nometoo</b> .                       |
| 97437 | XSI <b>onehop</b>          | When responding to a message that was originally sent to several recipients, the                         |
| 97438 |                            | other recipient addresses are normally forced to be relative to the originating                          |
| 97439 |                            | author's machine for the response. This flag disables alteration of the recipients'                      |
| 97440 |                            | addresses, improving efficiency in a network where all machines can send directly                        |
| 97441 |                            | to all other machines (that is, one hop away). The default shall be <b>noonehop</b> .                    |
| 97442 | <b>outfolder</b>           | Cause the files used to record outgoing messages to be located in the directory                          |
| 97443 |                            | specified by the <b>folder</b> variable unless the pathname is absolute. The default shall               |
| 97444 |                            | be <b>nooutfolder</b> . See the <b>record</b> variable.                                                  |
| 97445 | <b>page</b>                | Insert a <form-feed> after each message sent through the pipe created by the <b>pipe</b>                 |
| 97446 |                            | command. The default shall be <b>nopage</b> .                                                            |
| 97447 | <b>prompt=string</b>       |                                                                                                          |
| 97448 |                            | Set the command-mode prompt to <i>string</i> . If <i>string</i> is null or if <b>noprompt</b> is set, no |
| 97449 |                            | prompting shall occur. The default shall be to prompt with the string " ? ".                             |
| 97450 | <b>quiet</b>               | Refrain from writing the opening message and version when entering <i>mailx</i> . The                    |
| 97451 |                            | default shall be <b>noquiet</b> .                                                                        |
| 97452 | <b>record=file</b>         | Record all outgoing mail in the file with the pathname <i>file</i> . The default shall be                |
| 97453 |                            | <b>norecord</b> . See also <b>outfolder</b> above.                                                       |
| 97454 | <b>save</b>                | Enable saving of messages in the dead-letter file on interrupt or delivery error. See                    |
| 97455 |                            | the variable <i>DEAD</i> for the location of the dead-letter file. The default shall be <b>save</b> .    |

**screen=number**

Set the number of lines in a screenful of headers for the **headers** and **z** commands. If **screen** is not specified, a value based on the terminal type identified by the *TERM* environment variable, the window size, the baud rate, or some combination of these shall be used.

**sendwait**

Wait for the background mailer to finish before returning. The default shall be **nosendwait**.

**showto**

When the sender of the message was the user who is invoking *mailx*, write the information from the **To:** line instead of the **From:** line in the header summary. The default shall be **noshowto**.

**sign=string**

Set the variable inserted into the text of a message when the **~a** command escape is given. The default shall be **nosign**. The character sequences '**\t**' and '**\n**' shall be recognized in the variable as <tab> and <newline> characters, respectively. (See also **~i** in [Command Escapes in mailx](#) (on page 2951).)

**Sign=string**

Set the variable inserted into the text of a message when the **~A** command escape is given. The default shall be **noSign**. The character sequences '**\t**' and '**\n**' shall be recognized in the variable as <tab> and <newline> characters, respectively.

**toplines=number**

Set the number of lines of the message to write with the **top** command. The default shall be 5.

**Commands in mailx**

The following *mailx* commands shall be provided. In the following list, header refers to lines from the message header, as shown in the OUTPUT FILES section. Header-line refers to lines within the header that begin with one or more non-white-space characters, immediately followed by a <colon> and white space and continuing until the next line beginning with a non-white-space character or an empty line. Header-field refers to the portion of a header line prior to the first <colon> in that line.

For each of the commands listed below, the command can be entered as the abbreviation (those characters in the Synopsis command word preceding the '**[ ]**'), the full command (all characters shown for the command word, omitting the '**[ ]**' and '**'**'), or any truncation of the full command down to the abbreviation. For example, the **exit** command (shown as **ex[it]** in the Synopsis) can be entered as **ex**, **exi**, or **exit**.

The arguments to commands can be quoted, using the following methods:

- An argument can be enclosed between paired double-quotes ("**"**") or single-quotes ("**'**"); any white space, shell word expansion, or <backslash> characters within the quotes shall be treated literally as part of the argument. A double-quote shall be treated literally within single-quotes and *vice versa*. These special properties of the <quotation-mark> characters shall occur only when they are paired at the beginning and end of the argument.
- A <backslash> outside of the enclosing quotes shall be discarded and the following character treated literally as part of the argument.
- An unquoted <backslash> at the end of a command line shall be discarded and the next line shall continue the command.

97498 Filenames, where expected, shall be subjected to the following transformations, in sequence:

- 97499 • If the filename begins with an unquoted <plus-sign>, and the **folder** variable is defined  
97500 (see the **folder** variable), the <plus-sign> shall be replaced by the value of the **folder**  
97501 variable followed by a <slash>. If the **folder** variable is unset or is set to null, the filename  
97502 shall be unchanged.
- 97503 • Shell word expansions shall be applied to the filename (see [Section 2.6](#), on page 2343). If  
97504 more than a single pathname results from this expansion and the command is expecting  
97505 one file, the effects are unspecified.

## 97506 Declare Aliases

97507 *Synopsis:*     a[lias] [alias [address...]]  
97508               g[roup] [alias [address...]]

97509 Add the given addresses to the alias specified by *alias*. The names shall be substituted when  
97510 *alias* is used as a recipient address specified by the user in an outgoing message (that is, other  
97511 recipients addressed indirectly through the **reply** command shall not be substituted in this  
97512 manner). Mail address alias substitution shall apply only when the alias string is used as a full  
97513 address; for example, when **hlj** is an alias, *hlj@posix.com* does not trigger the alias substitution. If  
97514 no arguments are given, write a listing of the current aliases to standard output. If only an *alias*  
97515 argument is given, write a listing of the specified alias to standard output. These listings need  
97516 not reflect the same order of addresses that were entered.

## 97517 Declare Alternatives

97518 *Synopsis:*     alt[ernates] name...

97519 (See also the **metoo** variable.) Declare a list of alternative names for the user's login. When  
97520 responding to a message, these names shall be removed from the list of recipients for the  
97521 response. The comparison of names shall be in a case-insensitive manner. With no arguments,  
97522 **alternates** shall write the current list of alternative names.

## 97523 Change Current Directory

97524 *Synopsis:*     cd [directory]  
97525               ch[dir] [directory]

97526 Change directory. If *directory* is not specified, the contents of *HOME* shall be used.

## 97527 Copy Messages

97528 *Synopsis:*     c[opy] [file]  
97529               c[opy] [msglist] file  
97530               C[opy] [msglist]

97531 Copy messages to the file named by the pathname *file* without marking the messages as saved.  
97532 Otherwise, it shall be equivalent to the **save** command.

97533 In the capitalized form, save the specified messages in a file whose name is derived from the  
97534 author of the message to be saved, without marking the messages as saved. Otherwise, it shall  
97535 be equivalent to the **Save** command.

## Delete Messages

*Synopsis:*     d[delete] [msglist]

Mark messages for deletion from the mailbox. The deletions shall not occur until *mailx* quits (see the **quit** command) or changes mailboxes (see the **folder** command). If **autoprint** is set and there are messages remaining after the **delete** command, the current message shall be written as described for the **print** command (see the **print** command); otherwise, the *mailx* prompt shall be written.

## Discard Header Fields

*Synopsis:*     di[scard] [header-field...]  
              ig[nore] [header-field...]

Suppress the specified header fields when writing messages. Specified *header-fields* shall be added to the list of suppressed header fields. Examples of header fields to ignore are **status** and **cc**. The fields shall be included when the message is saved. The **Print** and **Type** commands shall override this command. The comparison of header fields shall be in a case-insensitive manner. If no arguments are specified, write a list of the currently suppressed header fields to standard output; the listing need not reflect the same order of header fields that were entered.

If both **retain** and **discard** commands are given, **discard** commands shall be ignored.

## Delete Messages and Display

*Synopsis:*     dp [msglist]  
              dt [msglist]

Delete the specified messages as described for the **delete** command, except that the **autoprint** variable shall have no effect, and the current message shall be written only if it was set to a message after the last message deleted by the command. Otherwise, an informational message to the effect that there are no further messages in the mailbox shall be written, followed by the *mailx* prompt.

## Echo a String

*Synopsis:*     ec[ho] string ...

Echo the given strings, equivalent to the shell *echo* utility.

## Edit Messages

*Synopsis:*     e[dit] [msglist]

Edit the given messages. The messages shall be placed in a temporary file and the utility named by the *EDITOR* variable is invoked to edit each file in sequence. The default *EDITOR* is unspecified.

The **edit** command does not modify the contents of those messages in the mailbox.

**Exit**

*Synopsis:*     ex[it]  
              x[it]

Exit from *mailx* without changing the mailbox. No messages shall be saved in the **mbox** (see also **quit**).

**Change Folder**

*Synopsis:*     fi[le] [*file*]  
              fold[er] [*file*]

Quit (see the **quit** command) from the current file of messages and read in the file named by the pathname *file*. If no argument is given, the name and status of the current mailbox shall be written.

Several unquoted special characters shall be recognized when used as *file* names, with the following substitutions:

- %       The system mailbox for the invoking user.
  - %*user*   The system mailbox for *user*.
  - #       The previous file.
  - &       The current **mbox**.
  - +*file*   The named file in the **folder** directory. (See the **folder** variable.)
- The default file shall be the current mailbox.

**Display List of Folders**

*Synopsis:*     folders

Write the names of the files in the directory set by the **folder** variable. The command specified by the *LISTER* environment variable shall be used (see the ENVIRONMENT VARIABLES section).

**Follow Up Specified Messages**

*Synopsis:*     fo[llowup] [*message*]  
              F[ollowup] [*msglist*]

In the lowercase form, respond to a message, recording the response in a file whose name is derived from the author of the message. See also the **save** and **copy** commands and **outfolder**.

In the capitalized form, respond to the first message in the *msglist*, sending the message to the author of each message in the *msglist*. The subject line shall be taken from the first message and the response shall be recorded in a file whose name is derived from the author of the first message. See also the **Save** and **Copy** commands and **outfolder**.

Both forms shall override the **record** variable, if set.

**Display Header Summary for Specified Messages**

*Synopsis:*    f[rom] [msglist]

Write the header summary for the specified messages.

**Display Header Summary**

*Synopsis:*    h[eaders] [message]

Write the page of headers that includes the message specified. If the *message* argument is not specified, the current message shall not change. However, if the *message* argument is specified, the current message shall become the message that appears at the top of the page of headers that includes the message specified. The **screen** variable sets the number of headers per page. See also the **z** command.

**Help**

*Synopsis:*    hel[p]  
              ?

Write a summary of commands.

**Hold Messages**

*Synopsis:*    ho[ld] [msglist]  
              pre[serve] [msglist]

Mark the messages in *msglist* to be retained in the mailbox when *mailx* terminates. This shall override any commands that might previously have marked the messages to be deleted. During the current invocation of *mailx*, only the **delete**, **dp**, or **dt** commands shall remove the *preserve* marking of a message.

**Execute Commands Conditionally**

*Synopsis:*    i[f] s|r  
              mail-commands  
              el[se]  
              mail-commands  
              en[dif]

Execute commands conditionally, where **if s** executes the following *mail-commands*, up to an **else** or **endif**, if the program is in Send Mode, and **if r** shall cause the *mail-commands* to be executed only in Receive Mode.

**List Available Commands**

*Synopsis:*    l[ist]

Write a list of all commands available. No explanation shall be given.

**Mail a Message**

*Synopsis:*    m[ail] address...

Mail a message to the specified addresses or aliases.

**Direct Messages to mbox**

*Synopsis:*    mb[ox] [msglist]

Arrange for the given messages to end up in the **mbox** save file when *mailx* terminates normally. See **MBOX**. See also the **exit** and **quit** commands.

**Process Next Specified Message**

*Synopsis:*    n[ext] [message]

If the current message has not been written (for example, by the **print** command) since *mailx* started or since any other message was the current message, behave as if the **print** command was entered. Otherwise, if there is an undeleted message after the current message, make it the current message and behave as if the **print** command was entered. Otherwise, an informational message to the effect that there are no further messages in the mailbox shall be written, followed by the *mailx* prompt. Should the current message location be the result of an immediately preceding **hold**, **mbox**, **preserve**, or **touch** command, **next** will act as if the current message has already been written.

**Pipe Message**

*Synopsis:*    pi[pe] [[msglist] command]  
              | [[msglist] command]

Pipe the messages through the given *command* by invoking the command interpreter specified by *SHELL* with two arguments: **-c** and *command*. (See also *sh -c*.) The application shall ensure that the command is given as a single argument. Quoting, described previously, can be used to accomplish this. If no arguments are given, the current message shall be piped through the command specified by the value of the **cmd** variable. If the **page** variable is set, a <form-feed> shall be inserted after each message.

**Display Message with Headers**

*Synopsis:*    P[rint] [msglist]  
              T[ype] [msglist]

Write the specified messages, including all header lines, to standard output. Override suppression of lines by the **discard**, **ignore**, and **retain** commands. If **crt** is set, the messages longer than the number of lines specified by the **crt** variable shall be paged through the command specified by the *PAGER* environment variable.

**Display Message**

*Synopsis:*    p[rint] [msglist]  
              t[ype] [msglist]

Write the specified messages to standard output. If **crt** is set, the messages longer than the number of lines specified by the **crt** variable shall be paged through the command specified by the *PAGER* environment variable.



## Quit

*Synopsis:*     q[uit]  
              end-of-file

Terminate *mailx*, storing messages that were read in **mbox** (if the current mailbox is the system mailbox and unless **hold** is set), deleting messages that have been explicitly saved (unless **keepsave** is set), discarding messages that have been deleted, and saving all remaining messages in the mailbox.

## Reply to a Message List

*Synopsis:*     R[eply] [msglist]  
              R[espond] [msglist]

Mail a reply message to the sender of each message in the *msglist*. The subject line shall be formed by concatenating **Re:**<space> (unless it already begins with that string) and the subject from the first message. If **record** is set to a filename, the response shall be saved at the end of that file.

See also the **flipr** variable.

## Reply to a Message

*Synopsis:*     r[eply] [message]  
              r[espond] [message]

Mail a reply message to all recipients included in the header of the message. The subject line shall be formed by concatenating **Re:**<space> (unless it already begins with that string) and the subject from the message. If **record** is set to a filename, the response shall be saved at the end of that file.

See also the **flipr** variable.

## Retain Header Fields

*Synopsis:*     ret[ain] [header-field...]

Retain the specified header fields when writing messages. This command shall override all **discard** and **ignore** commands. The comparison of header fields shall be in a case-insensitive manner. If no arguments are specified, write a list of the currently retained header fields to standard output; the listing need not reflect the same order of header fields that were entered.

## Save Messages

*Synopsis:*     s[ave] [file]  
              s[ave] [msglist] file  
              S[ave] [msglist]

Save the specified messages in the file named by the pathname *file*, or the **mbox** if the *file* argument is omitted. The file shall be created if it does not exist; otherwise, the messages shall be appended to the file. The message shall be put in the state *saved*, and shall behave as specified in the description of the *saved* state when the current mailbox is exited by the **quit** or **file** command.

In the capitalized form, save the specified messages in a file whose name is derived from the author of the first message. The name of the file shall be taken to be the author's name with all network addressing stripped off. See also the **Copy**, **followup**, and **Followup** commands and



97717 **outfolder** variable.

## 97718 **Set Variables**

97719 *Synopsis:*    **se**[t] [**name**=[*string*] ...] [**name**=*number* ...] [**no**name ...]

97720 Define one or more variables called *name*. The variable can be given a null, string, or numeric  
97721 value. Quoting and <backslash>-escapes can occur anywhere in *string*, as described previously,  
97722 as if the *string* portion of the argument were the entire argument. The forms *name* and *name*=  
97723 shall be equivalent to *name*="" for variables that take string values. The **set** command without  
97724 arguments shall write a list of all defined variables and their values. The **no** *name* form shall be  
97725 equivalent to **unset** *name*.

## 97726 **Invoke a Shell**

97727 *Synopsis:*    **sh**[ell]

97728 Invoke an interactive command interpreter (see also *SHELL*).

## 97729 **Display Message Size**

97730 *Synopsis:*    **si**[ze] [*msglist*]

97731 Write the size in bytes of each of the specified messages.

## 97732 **Read mailx Commands From a File**

97733 *Synopsis:*    **so**[urce] *file*

97734 Read and execute commands from the file named by the pathname *file* and return to command  
97735 mode.

## 97736 **Display Beginning of Messages**

97737 *Synopsis:*    **to**[p] [*msglist*]

97738 Write the top few lines of each of the specified messages. If the **toplines** variable is set, it is taken  
97739 as the number of lines to write. The default shall be 5.

## 97740 **Touch Messages**

97741 *Synopsis:*    **tou**[ch] [*msglist*]

97742 Touch the specified messages. If any message in *msglist* is not specifically deleted nor saved in a  
97743 file, it shall be placed in the **mbox** upon normal termination. See **exit** and **quit**.

## 97744 **Delete Aliases**

97745 *Synopsis:*    **una**[lias] [*alias*]...

97746 Delete the specified alias names. If a specified alias does not exist, the results are unspecified.

## Undelete Messages

*Synopsis:*     u[ndelete] [msglist]

Change the state of the specified messages from deleted to read. If **autoprint** is set, the last message of those restored shall be written. If *msglist* is not specified, the message shall be selected as follows:

- If there are any deleted messages that follow the current message, the first of these shall be chosen.
- Otherwise, the last deleted message that also precedes the current message shall be chosen.

## Unset Variables

*Synopsis:*     uns[et] name...

Cause the specified variables to be erased.

## Edit Message with Full-Screen Editor

*Synopsis:*     v[isual] [msglist]

Edit the given messages with a screen editor. Each message shall be placed in a temporary file, and the utility named by the *VISUAL* variable shall be invoked to edit each file in sequence. The default editor shall be *vi*.

The **visual** command does not modify the contents of those messages in the mailbox.

## Write Messages to a File

*Synopsis:*     w[rite] [msglist] file

Write the given messages to the file specified by the pathname *file*, minus the message header. Otherwise, it shall be equivalent to the **save** command.

## Scroll Header Display

*Synopsis:*     z[+|-]

Scroll the header display forward (if '+' is specified or if no option is specified) or backward (if '-' is specified) one screenful. The number of headers written shall be set by the **screen** variable.

## Invoke Shell Command

*Synopsis:*     !command

Invoke the command interpreter specified by *SHELL* with two arguments: **-c** and *command*. (See also *sh -c*.) If the **bang** variable is set, each unescaped occurrence of '!' in *command* shall be replaced with the command executed by the previous ! command or ~! command escape.

**Null Command**

*Synopsis:*    # *comment*

This null command (comment) shall be ignored by *mailx*.

**Display Current Message Number**

*Synopsis:*       =

Write the current message number.

**Command Escapes in mailx**

The following commands can be entered only from input mode, by beginning a line with the escape character (by default, <tilde> ('~')). See the **escape** variable description for changing this special character. The format for the commands shall be:

<escape-character><command-char><separator>[<arguments>]

where the <separator> can be zero or more <blank> characters.

In the following descriptions, the application shall ensure that the argument *command* (but not *mailx-command*) is a shell command string. Any string acceptable to the command interpreter specified by the *SHELL* variable when it is invoked as *SHELL -c command\_string* shall be valid. The command can be presented as multiple arguments (that is, quoting is not required).

Command escapes that are listed with *msglist* or *mailx-command* arguments are invalid in Send Mode and produce unspecified results.

**~! *command*** Invoke the command interpreter specified by *SHELL* with two arguments: **-c** and *command*; and then return to input mode. If the **bang** variable is set, each unescaped occurrence of '!' in *command* shall be replaced with the command executed by the previous ! command or ~! command escape.

**~.** Simulate end-of-file (terminate message input).

**~: *mailx-command*, ~\_ *mailx-command***  
Perform the command-level request.

**~?** Write a summary of command escapes.

**~A** This shall be equivalent to **~i Sign**.

**~a** This shall be equivalent to **~i sign**.

**~b *name...*** Add the *names* to the blind carbon copy (**Bcc**) list.

**~c *name...*** Add the *names* to the carbon copy (**Cc**) list.

**~d** Read in the dead-letter file. See *DEAD* for a description of this file.

**~e** Invoke the editor, as specified by the *EDITOR* environment variable, on the partial message.

**~f [*msglist*]** Forward the specified messages. The specified messages shall be inserted into the current message without alteration. This command escape also shall insert message headers into the message with field selection affected by the **discard**, **ignore**, and **retain** commands.

|       |                                                                             |                                                                                                                  |
|-------|-----------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------|
| 97815 | ~F [ <i>msglist</i> ]                                                       | This shall be the equivalent of the ~f command escape, except that all headers shall                             |
| 97816 |                                                                             | be included in the message, regardless of previous <b>discard</b> , <b>ignore</b> , and <b>retain</b>            |
| 97817 |                                                                             | commands.                                                                                                        |
| 97818 | ~h                                                                          | If standard input is a terminal, prompt for a <b>Subject</b> line and the <b>To</b> , <b>Cc</b> , and <b>Bcc</b> |
| 97819 |                                                                             | lists. Other implementation-defined headers may also be presented for editing. If                                |
| 97820 |                                                                             | the field is written with an initial value, it can be edited as if it had just been typed.                       |
| 97821 | ~i <i>string</i>                                                            | Insert the value of the named variable, followed by a <newline>, into the text of                                |
| 97822 |                                                                             | the message. If the string is unset or null, the message shall not be changed.                                   |
| 97823 | ~m [ <i>msglist</i> ]                                                       | Insert the specified messages into the message, prefixing non-empty lines with the                               |
| 97824 |                                                                             | string in the <b>indentprefix</b> variable. This command escape also shall insert message                        |
| 97825 |                                                                             | headers into the message, with field selection affected by the <b>discard</b> , <b>ignore</b> , and              |
| 97826 |                                                                             | <b>retain</b> commands.                                                                                          |
| 97827 | ~M [ <i>msglist</i> ]                                                       | This shall be the equivalent of the ~m command escape, except that all headers                                   |
| 97828 |                                                                             | shall be included in the message, regardless of previous <b>discard</b> , <b>ignore</b> , and <b>retain</b>      |
| 97829 |                                                                             | commands.                                                                                                        |
| 97830 | ~p                                                                          | Write the message being entered. If the message is longer than <b>crt</b> lines (see                             |
| 97831 |                                                                             | <a href="#">Internal Variables in mailx</a> , on page 2939), the output shall be paginated as                    |
| 97832 |                                                                             | described for the <b>PAGER</b> variable.                                                                         |
| 97833 | ~q                                                                          | Quit (see the <b>quit</b> command) from input mode by simulating an interrupt. If the                            |
| 97834 |                                                                             | body of the message is not empty, the partial message shall be saved in the dead-                                |
| 97835 |                                                                             | letter file. See <i>DEAD</i> for a description of this file.                                                     |
| 97836 | ~r <i>file</i> , ~< <i>file</i> , ~r ! <i>command</i> , ~< ! <i>command</i> |                                                                                                                  |
| 97837 |                                                                             | Read in the file specified by the pathname <i>file</i> . If the argument begins with an                          |
| 97838 |                                                                             | <exclamation-mark> ('!'), the rest of the string shall be taken as an arbitrary                                  |
| 97839 |                                                                             | system command; the command interpreter specified by <i>SHELL</i> shall be invoked                               |
| 97840 |                                                                             | with two arguments: -c and <i>command</i> . The standard output of <i>command</i> shall be                       |
| 97841 |                                                                             | inserted into the message.                                                                                       |
| 97842 | ~s <i>string</i>                                                            | Set the subject line to <i>string</i> .                                                                          |
| 97843 | ~t <i>name...</i>                                                           | Add the given <i>names</i> to the <b>To</b> list.                                                                |
| 97844 | ~v                                                                          | Invoke the full-screen editor, as specified by the <i>VISUAL</i> environment variable, on                        |
| 97845 |                                                                             | the partial message.                                                                                             |
| 97846 | ~w <i>file</i>                                                              | Write the partial message, without the header, onto the file named by the                                        |
| 97847 |                                                                             | pathname <i>file</i> . The file shall be created or the message shall be appended to it if                       |
| 97848 |                                                                             | the file exists.                                                                                                 |
| 97849 | ~x                                                                          | Exit as with ~q, except the message shall not be saved in the dead-letter file.                                  |
| 97850 | ~l <i>command</i>                                                           | Pipe the body of the message through the given <i>command</i> by invoking the                                    |
| 97851 |                                                                             | command interpreter specified by <i>SHELL</i> with two arguments: -c and <i>command</i> . If                     |
| 97852 |                                                                             | the <i>command</i> returns a successful exit status, the standard output of the command                          |
| 97853 |                                                                             | shall replace the message. Otherwise, the message shall remain unchanged. If the                                 |
| 97854 |                                                                             | <i>command</i> fails, an error message giving the exit status shall be written.                                  |

97855 **EXIT STATUS**97856 UP When the **-e** option is specified, the following exit values are returned:

97857 0 Mail was found.

97858 &gt;0 Mail was not found or an error occurred.

97859 Otherwise, the following exit values are returned:

97860 0 Successful completion; note that this status implies that all messages were *sent*, but it gives  
97861 no assurances that any of them were actually *delivered*.

97862 &gt;0 An error occurred.

97863 **CONSEQUENCES OF ERRORS**97864 UP When in **input mode (Receive Mode)** or Send Mode:97865 • If an error is encountered processing an input line beginning with a <tilde> ('~')  
97866 UP character, (see [Command Escapes in mailx](#), on page 2951), a diagnostic message shall be  
97867 written to standard error, and the message being composed may be modified, but this  
97868 condition shall not prevent the message from being sent.

97869 • Other errors shall prevent the sending of the message.

97870 UP When in command mode:

97871 • Default.

97872 **APPLICATION USAGE**97873 Delivery of messages to remote systems requires the existence of communication paths to such  
97874 systems. These need not exist.97875 Input lines are limited to {LINE\_MAX} bytes, but mailers between systems may impose more  
97876 severe line-length restrictions. This volume of POSIX.1-2008 does not place any restrictions on  
97877 the length of messages handled by *mailx*, and for delivery of local messages the only limitations  
97878 should be the normal problems of available disk space for the target mail file. When sending  
97879 messages to external machines, applications are advised to limit messages to less than 100 000  
97880 bytes because some mail gateways impose message-length restrictions.97881 The format of the system mailbox is intentionally unspecified. Not all systems implement  
97882 system mailboxes as flat files, particularly with the advent of multimedia mail messages. Some  
97883 system mailboxes may be multiple files, others records in a database. The internal format of the  
97884 messages themselves is specified with the historical format from Version 7, but only after the  
97885 messages have been saved in some file other than the system mailbox. This was done so that  
97886 many historical applications expecting text-file mailboxes are not broken.97887 Some new formats for messages can be expected in the future, probably including binary data,  
97888 bit maps, and various multimedia objects. As described here, *mailx* is not prohibited from  
97889 handling such messages, but it must store them as text files in secondary mailboxes (unless some  
97890 extension, such as a variable or command line option, is used to change the stored format). Its  
97891 method of doing so is implementation-defined and might include translating the data into text  
97892 file-compatible or readable form or omitting certain portions of the message from the stored  
97893 output.97894 The **discard** and **ignore** commands are not inverses of the **retain** command. The **retain**  
97895 command discards all header-fields except those explicitly retained. The **discard** command  
97896 keeps all header-fields except those explicitly discarded. If headers exist on the retained header

list, **discard** and **ignore** commands are ignored.

## EXAMPLES

None.

## RATIONALE

The standard developers felt strongly that a method for applications to send messages to specific users was necessary. The obvious example is a batch utility, running non-interactively, that wishes to communicate errors or results to a user. However, the actual format, delivery mechanism, and method of reading the message are clearly beyond the scope of this volume of POSIX.1-2008.

The intent of this command is to provide a simple, portable interface for sending messages non-interactively. It merely defines a “front-end” to the historical mail system. It is suggested that implementations explicitly denote the sender and recipient in the body of the delivered message. Further specification of formats for either the message envelope or the message itself were deliberately not made, as the industry is in the midst of changing from the current standards to a more internationalized standard and it is probably incorrect, at this time, to require either one.

Implementations are encouraged to conform to the various delivery mechanisms described in the CCITT X.400 standards or to the equivalent Internet standards, described in Internet Request for Comment (RFC) documents RFC 819, RFC 920, RFC 921, RFC 1123, and RFC 5322 (which succeeded RFC 822).

Many historical systems modified each body line that started with **From** by prefixing the ‘F’ with ‘>’. It is unnecessary, but allowed, to do that when the string does not follow a blank line because it cannot be confused with the next header.

The **edit** and **visual** commands merely edit the specified messages in a temporary file. They do not modify the contents of those messages in the mailbox; such a capability could be added as an extension, such as by using different command names.

The restriction on a subject line being {LINE\_MAX}–10 bytes is based on the historical format that consumes 10 bytes for **Subject:** and the trailing <newline>. Many historical mailers that a message may encounter on other systems are not able to handle lines that long, however.

Like the utilities *logger* and *lp*, *mailx* admittedly is difficult to test. This was not deemed sufficient justification to exclude this utility from this volume of POSIX.1-2008. It is also arguable that it is, in fact, testable, but that the tests themselves are not portable.

When *mailx* is being used by an application that wishes to receive the results as if none of the User Portability Utilities option features were supported, the *DEAD* environment variable must be set to **/dev/null**. Otherwise, it may be subject to the file creations described in *mailx* ASYNCHRONOUS EVENTS. Similarly, if the *MAILRC* environment variable is not set to **/dev/null**, historical versions of *mailx* and *Mail* read initialization commands from a file before processing begins. Since the initialization that a user specifies could alter the contents of messages an application is trying to send, such applications must set *MAILRC* to **/dev/null**.

The description of *LC\_TIME* uses “may affect” because many historical implementations do not or cannot manipulate the date and time strings in the incoming mail headers. Some headers found in incoming mail do not have enough information to determine the timezone in which the mail originated, and, therefore, *mailx* cannot convert the date and time strings into the internal form that then is parsed by routines like *strftime()* that can take *LC\_TIME* settings into account. Changing all these times to a user-specified format is allowed, but not required.

The paginator selected when *PAGER* is null or unset is partially unspecified to allow the System V historical practice of using *pg* as the default. Bypassing the pagination function, such as by



declaring that *cat* is the paginator, would not meet with the intended meaning of this description. However, any “portable user” would have to set *PAGER* explicitly to get his or her preferred paginator on all systems. The paginator choice was made partially unspecified, unlike the *VISUAL* editor choice (mandated to be *vi*) because most historical pagers follow a common theme of user input, whereas editors differ dramatically.

Options to specify addresses as **cc** (carbon copy) or **bcc** (blind carbon copy) were considered to be format details and were omitted.

A zero exit status implies that all messages were *sent*, but it gives no assurances that any of them were actually *delivered*. The reliability of the delivery mechanism is unspecified and is an appropriate marketing distinction between systems.

In order to conform to the Utility Syntax Guidelines, a solution was required to the optional *file* option-argument to **-f**. By making *file* an operand, the guidelines are satisfied and users remain portable. However, it does force implementations to support usage such as:

```
mailx -fin mymail.box
```

The **no name** method of unsetting variables is not present in all historical systems, but it is in System V and provides a logical set of commands corresponding to the format of the display of options from the *mailx set* command without arguments.

The **ask** and **asksub** variables are the names selected by BSD and System V, respectively, for the same feature. They are synonyms in this volume of POSIX.1-2008.

The *mailx echo* command was not documented in the BSD version and has been omitted here because it is not obviously useful for interactive users.

The default prompt on the System V *mailx* is a <question-mark>, on BSD *Mail* an <ampersand>. Since this volume of POSIX.1-2008 chose the *mailx* name, it kept the System V default, assuming that BSD users would not have difficulty with this minor incompatibility (that they can override).

The meanings of **r** and **R** are reversed between System V *mailx* and SunOS *Mail*. Once again, since this volume of POSIX.1-2008 chose the *mailx* name, it kept the System V default, but allows the SunOS user to achieve the desired results using **flpr**, an internal variable in System V *mailx*, although it has not been documented in the SVID.

The **indentprefix** variable, the **retain** and **unalias** commands, and the **~F** and **~M** command escapes were adopted from 4.3 BSD *Mail*.

The **version** command was not included because no sufficiently general specification of the version information could be devised that would still be useful to a portable user. This command name should be used by suppliers who wish to provide version information about the *mailx* command.

The “implementation-specific (unspecified) system start-up file” historically has been named */etc/mailx.rc*, but this specific name and location are not required.

The intent of the wording for the **next** command is that if any command has already displayed the current message it should display a following message, but, otherwise, it should display the current message. Consider the command sequence:

```
next 3
delete 3
next
```

where the **autoprint** option was not set. The normative text specifies that the second **next**

command should display a message following the third message, because even though the current message has not been displayed since it was set by the **delete** command, it has been displayed since the current message was anything other than message number 3. This does not always match historical practice in some implementations, where the command file address followed by **next** (or the default command) would skip the message for which the user had searched.

## FUTURE DIRECTIONS

None.

## SEE ALSO

Chapter 2 (on page 2335), *ed*, *ls*, *more*, *vi*

XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

## CHANGE HISTORY

First released in Issue 2.

### Issue 5

The description of the *EDITOR* environment variable is changed to indicate that *ed* is the default editor if this variable is not set. In previous issues, this default was not stated explicitly at this point but was implied further down in the text.

The FUTURE DIRECTIONS section is added.

### Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **-F** option is added.
- The **allnet**, **debug**, and **sendwait** internal variables are added.
- The **C**, **ec**, **fo**, **F**, and **S** *mailx* commands are added.

In the DESCRIPTION and ENVIRONMENT VARIABLES sections, text stating “*HOME* directory” is replaced by “directory referred to by the *HOME* environment variable”.

The *mailx* utility is aligned with the IEEE P1003.2b draft standard, which includes various clarifications to resolve IEEE PASC Interpretations submitted for the ISO POSIX-2:1993 standard. In particular, the changes here address IEEE PASC Interpretations 1003.2 #10, #11, #103, #106, #108, #114, #115, #122, and #129.

The normative text is reworded to avoid use of the term “must” for application requirements.

The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/32 is applied, applying a change to the EXTENDED DESCRIPTION, raised by IEEE PASC Interpretation 1003.2 #122, which was overlooked in the first version of this standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/17 is applied, updating the EXTENDED DESCRIPTION (Internal Variables in *mailx*). The system start-up file is changed from “implementation-defined” to “unspecified” for consistency with other text in the EXTENDED DESCRIPTION.

### Issue 7

Austin Group Interpretation 1003.1-2001 #089 is applied, clarifying the effect of the *LC\_TIME* environment variable.

Austin Group Interpretation 1003.1-2001 #090 is applied, updating the description of the **next**



- 98030 command.
- 98031 SD5-XCU-ERN-69 is applied.
- 98032 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.
- 98033 Shading to indicate support for the User Portability Utilities option is added.

98034 **NAME**98035 `make` — maintain, update, and regenerate groups of programs (**DEVELOPMENT**)98036 **SYNOPSIS**

```
98037 SD  make [-einpqrst] [-f makefile]... [-k|-S] [macro=value...]
98038      [target_name...]
```

98039 **DESCRIPTION**

98040 The *make* utility shall update files that are derived from other files. A typical case is one where  
 98041 object files are derived from the corresponding source files. The *make* utility examines time  
 98042 relationships and shall update those derived files (called targets) that have modified times  
 98043 earlier than the modified times of the files (called prerequisites) from which they are derived. A  
 98044 description file (makefile) contains a description of the relationships between files, and the  
 98045 commands that need to be executed to update the targets to reflect changes in their  
 98046 prerequisites. Each specification, or rule, shall consist of a target, optional prerequisites, and  
 98047 optional commands to be executed when a prerequisite is newer than the target. There are two  
 98048 types of rule:

- 98049 1. *Inference rules*, which have one target name with at least one <period> ('.') and no  
 98050 <slash> ('/')
- 98051 2. *Target rules*, which can have more than one target name

98052 In addition, *make* shall have a collection of built-in macros and inference rules that infer  
 98053 prerequisite relationships to simplify maintenance of programs.

98054 To receive exactly the behavior described in this section, the user shall ensure that a portable  
 98055 makefile shall:

- 98056 • Include the special target **.POSIX**
- 98057 • Omit any special target reserved for implementations (a leading period followed by  
 98058 uppercase letters) that has not been specified by this section

98059 The behavior of *make* is unspecified if either or both of these conditions are not met.

98060 **OPTIONS**

98061 The *make* utility shall conform to XBD [Section 12.2](#) (on page 216), except for Guideline 9.

98062 The following options shall be supported:

- |       |                    |                                                                                                                                                                                                                                                                                                                                                                                            |
|-------|--------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 98063 | <b>-e</b>          | Cause environment variables, including those with null values, to override macro assignments within makefiles.                                                                                                                                                                                                                                                                             |
| 98064 |                    |                                                                                                                                                                                                                                                                                                                                                                                            |
| 98065 | <b>-f makefile</b> | Specify a different makefile. The argument <i>makefile</i> is a pathname of a description file, which is also referred to as the <i>makefile</i> . A pathname of '-' shall denote the standard input. There can be multiple instances of this option, and they shall be processed in the order specified. The effect of specifying the same option-argument more than once is unspecified. |
| 98066 |                    |                                                                                                                                                                                                                                                                                                                                                                                            |
| 98067 |                    |                                                                                                                                                                                                                                                                                                                                                                                            |
| 98068 |                    |                                                                                                                                                                                                                                                                                                                                                                                            |
| 98069 |                    |                                                                                                                                                                                                                                                                                                                                                                                            |
| 98070 | <b>-i</b>          | Ignore error codes returned by invoked commands. This mode is the same as if the special target <b>.IGNORE</b> were specified without prerequisites.                                                                                                                                                                                                                                       |
| 98071 |                    |                                                                                                                                                                                                                                                                                                                                                                                            |
| 98072 | <b>-k</b>          | Continue to update other targets that do not depend on the current target if a non-ignored error occurs while executing the commands to bring a target up-to-date.                                                                                                                                                                                                                         |
| 98073 |                    |                                                                                                                                                                                                                                                                                                                                                                                            |
| 98074 | <b>-n</b>          | Write commands that would be executed on standard output, but do not execute them. However, lines with a <plus-sign> ('+') prefix shall be executed. In this mode, lines with an at-sign ('@') character prefix shall be written to standard                                                                                                                                               |
| 98075 |                    |                                                                                                                                                                                                                                                                                                                                                                                            |
| 98076 |                    |                                                                                                                                                                                                                                                                                                                                                                                            |

- 98077 output.
- 98078 **-p** Write to standard output the complete set of macro definitions and target  
98079 descriptions. The output format is unspecified.
- 98080 **-q** Return a zero exit value if the target file is up-to-date; otherwise, return an exit  
98081 value of 1. Targets shall not be updated if this option is specified. However, a  
98082 makefile command line (associated with the targets) with a <plus-sign> ('+')  
98083 prefix shall be executed.
- 98084 **-r** Clear the suffix list and do not use the built-in rules.
- 98085 **-S** Terminate *make* if an error occurs while executing the commands to bring a target  
98086 up-to-date. This shall be the default and the opposite of **-k**.
- 98087 **-s** Do not write makefile command lines or touch messages (see **-t**) to standard  
98088 output before executing. This mode shall be the same as if the special target  
98089 **.SILENT** were specified without prerequisites.
- 98090 **-t** Update the modification time of each target as though a *touch target* had been  
98091 executed. Targets that have prerequisites but no commands (see [Target Rules](#), on  
98092 page 2963), or that are already up-to-date, shall not be touched in this manner.  
98093 Write messages to standard output for each target file indicating the name of the  
98094 file and that it was touched. Normally, the *makefile* command lines associated with  
98095 each target are not executed. However, a command line with a <plus-sign> ('+')  
98096 prefix shall be executed.

98097 Any options specified in the *MAKEFLAGS* environment variable shall be evaluated before any  
98098 options specified on the *make* utility command line. If the **-k** and **-S** options are both specified  
98099 on the *make* utility command line or by the *MAKEFLAGS* environment variable, the last option  
98100 specified shall take precedence. If the **-f** or **-p** options appear in the *MAKEFLAGS* environment  
98101 variable, the result is undefined.

## 98102 OPERANDS

98103 The following operands shall be supported:

98104 *target\_name* Target names, as defined in the EXTENDED DESCRIPTION section. If no target is  
98105 specified, while *make* is processing the makefiles, the first target that *make*  
98106 encounters that is not a special target or an inference rule shall be used.

98107 *macro=value* Macro definitions, as defined in [Macros](#) (on page 2965).

98108 If the *target\_name* and *macro=value* operands are intermixed on the *make* utility command line,  
98109 the results are unspecified.

## 98110 STDIN

98111 The standard input shall be used only if the *makefile* option-argument is '-'. See the INPUT  
98112 FILES section.

## 98113 INPUT FILES

98114 The input file, otherwise known as the makefile, is a text file containing rules, macro definitions, |  
98115 include lines, and comments. See the EXTENDED DESCRIPTION section.

## 98116 ENVIRONMENT VARIABLES

98117 The following environment variables shall affect the execution of *make*:

98118 *LANG* Provide a default value for the internationalization variables that are unset or null.  
98119 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
98120 variables used to determine the values of locale categories.)

|       |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------|-----|----------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 98121 |     | <b>LC_ALL</b>              | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           |
| 98122 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98123 |     | <b>LC_CTYPE</b>            | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files).                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 98124 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98125 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98126 |     | <b>LC_MESSAGES</b>         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98127 |     |                            | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
| 98128 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98129 |     | <b>MAKEFLAGS</b>           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98130 |     |                            | This variable shall be interpreted as a character string representing a series of option characters to be used as the default options. The implementation shall accept both of the following formats (but need not accept them when intermixed):                                                                                                                                                                                                                                                                                                                                                                   |
| 98131 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98132 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98133 |     |                            | • The characters are option letters without the leading <hyphen> characters or <blank> separation used on a <i>make</i> utility command line.                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 98134 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98135 |     |                            | • The characters are formatted in a manner similar to a portion of the <i>make</i> utility command line: options are preceded by <hyphen-minus> characters and <blank>-separated as described in XBD <a href="#">Section 12.2</a> (on page 216). The <i>macro=value</i> macro definition operands can also be included. The difference between the contents of <b>MAKEFLAGS</b> and the <i>make</i> utility command line is that the contents of the variable shall not be subjected to the word expansions (see <a href="#">Section 2.6</a> , on page 2343) associated with parsing the command line values.      |
| 98136 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98137 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98138 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98139 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98140 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98141 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98142 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98143 | XSI | <b>NLSPATH</b>             | Determine the location of message catalogs for the processing of <b>LC_MESSAGES</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 98144 | XSI | <b>PROJECTDIR</b>          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98145 |     |                            | Provide a directory to be used to search for SCCS files not found in the current directory. In all of the following cases, the search for SCCS files is made in the directory <b>SCCS</b> in the identified directory. If the value of <b>PROJECTDIR</b> begins with a <slash>, it shall be considered an absolute pathname; otherwise, the value of <b>PROJECTDIR</b> is treated as a user name and that user's initial working directory shall be examined for a subdirectory <b>src</b> or <b>source</b> . If such a directory is found, it shall be used. Otherwise, the value is used as a relative pathname. |
| 98146 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98147 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98148 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98149 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98150 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98151 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98152 |     |                            | If <b>PROJECTDIR</b> is not set or has a null value, the search for SCCS files shall be made in the directory <b>SCCS</b> in the current directory.                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 98153 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98154 |     |                            | The setting of <b>PROJECTDIR</b> affects all files listed in the remainder of this utility description for files with a component named <b>SCCS</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 98155 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98156 |     |                            | The value of the <b>SHELL</b> environment variable shall not be used as a macro and shall not be modified by defining the <b>SHELL</b> macro in a makefile or on the command line. All other environment variables, including those with null values, shall be used as macros, as defined in <a href="#">Macros</a> (on page 2965).                                                                                                                                                                                                                                                                                |
| 98157 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98158 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98159 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98160 |     | <b>ASYNCHRONOUS EVENTS</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98161 |     |                            | If not already ignored, <i>make</i> shall trap SIGHUP, SIGTERM, SIGINT, and SIGQUIT and remove the current target unless the target is a directory or the target is a prerequisite of the special target <b>.PRECIOUS</b> or unless one of the <b>-n</b> , <b>-p</b> , or <b>-q</b> options was specified. Any targets removed in this manner shall be reported in diagnostic messages of unspecified format, written to standard error. After this cleanup process, if any, <i>make</i> shall take the standard action for all other signals.                                                                     |
| 98162 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98163 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98164 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98165 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 98166 |     |                            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |

**STDOUT**

The *make* utility shall write all commands to be executed to standard output unless the `-s` option was specified, the command is prefixed with an at-sign, or the special target **.SILENT** has either the current target as a prerequisite or has no prerequisites. If *make* is invoked without any work needing to be done, it shall write a message to standard output indicating that no action was taken. If the `-t` option is present and a file is touched, *make* shall write to standard output a message of unspecified format indicating that the file was touched, including the filename of the file.

**STDERR**

The standard error shall be used only for diagnostic messages.

**OUTPUT FILES**

Files can be created when the `-t` option is present. Additional files can also be created by the utilities invoked by *make*.

**EXTENDED DESCRIPTION**

The *make* utility attempts to perform the actions required to ensure that the specified targets are up-to-date. A target shall be considered up-to-date if it exists and is newer than all of its dependencies, or if it has already been made up-to-date by the current invocation of *make* (regardless of the target's existence or age). A target may also be considered up-to-date if it exists, is the same age as one or more of its prerequisites, and is newer than the remaining prerequisites (if any). The *make* utility shall treat all prerequisites as targets themselves and recursively ensure that they are up-to-date, processing them in the order in which they appear in the rule. The *make* utility shall use the modification times of files to determine whether the corresponding targets are out-of-date.

To ensure that a target is up-to-date, *make* shall ensure that all of the prerequisites of a target are up-to-date, then check to see if the target itself is up-to-date. If the target is not up-to-date, the target shall be made up-to-date by executing the rule's commands (if any). If the target does not exist after the target has been successfully made up-to-date, the target shall be treated as being newer than any target for which it is a prerequisite.

If a target exists and there is neither a target rule nor an inference rule for the target, the target shall be considered up-to-date. It shall be an error if *make* attempts to ensure that a target is up-to-date but the target does not exist and there is neither a target rule nor an inference rule for the target.

**Makefile Syntax**

A makefile can contain rules, macro definitions (see [Macros](#), on page 2965), include lines, and comments. There are two kinds of rules: *inference rules* and *target rules*. The *make* utility shall contain a set of built-in inference rules. If the `-r` option is present, the built-in rules shall not be used and the suffix list shall be cleared. Additional rules of both types can be specified in a makefile. If a rule is defined more than once, the value of the rule shall be that of the last one specified. Macros can also be defined more than once, and the value of the macro is specified in [Macros](#) (on page 2965). There are three kinds of comments: blank lines, empty lines, and a `<number-sign> (' #')` and all following characters up to the first unescaped `<newline>` character. Blank lines, empty lines, and lines with `<number-sign> (' #')` as the first character on the line are also known as comment lines.

By default, the following files shall be tried in sequence: **./makefile** and **./Makefile**. If neither **./makefile** or **./Makefile** are found, other implementation-defined files may also be tried. On XSI-conformant systems, the additional files **./s.makefile**, **SCCS/s.makefile**, **./s.Makefile**, and **SCCS/s.Makefile** shall also be tried.

The `-f` option shall direct *make* to ignore any of these default files and use the specified argument as a makefile instead. If the `'-'` argument is specified, standard input shall be used.

The term *makefile* is used to refer to any rules provided by the user, whether in `./makefile` or its variants, or specified by the `-f` option.

The rules in makefiles shall consist of the following types of lines: target rules, including special targets (see [Target Rules](#), on page 2963), inference rules (see [Inference Rules](#), on page 2966), macro definitions (see [Macros](#), on page 2965), and comments.

Target and Inference Rules may contain *command lines*. Command lines can have a prefix that shall be removed before execution (see [Makefile Execution](#), on page 2963).

When an escaped `<newline>` (one preceded by a `<backslash>`) is found anywhere in the makefile except in a command line, an include line, or a line immediately preceding an include line, it shall be replaced, along with any leading white space on the following line, with a single `<space>`. When an escaped `<newline>` is found in a command line in a makefile, the command line shall contain the `<backslash>`, the `<newline>`, and the next line, except that the first character of the next line shall not be included if it is a `<tab>`. When an escaped `<newline>` is found in an include line or in a line immediately preceding an include line, the behavior is unspecified.

### Include Lines

If the word **include** appears at the beginning of a line and is followed by one or more `<blank>` characters, the string formed by the remainder of the line shall be processed as follows to produce a pathname:

- The trailing `<newline>`, any `<blank>` characters immediately preceding a comment, and any comment shall be discarded. If the resulting string contains any double-quote characters (`' ' ' '`) the behavior is unspecified.
- The resulting string shall be processed for macro expansion (see [Macros](#) (on page 2965)).
- Any `<blank>` characters that appear after the first non-`<blank>` shall be used as separators to divide the macro-expanded string into fields. It is unspecified whether any other white-space characters are also used as separators. It is unspecified whether pathname expansion (see [Section 2.13](#), on page 2372) is also performed.
- If the processing of separators and optional pathname expansion results in either zero or two or more non-empty fields, the behavior is unspecified. If it results in one non-empty field, that field is taken as the pathname.

If the pathname does not begin with a `' / '` it shall be treated as relative to the current working directory of the process, not relative to the directory containing the makefile. If the file does not exist in this location, it is unspecified whether additional directories are searched.

The contents of the file specified by the pathname shall be read and processed as if they appeared in the makefile in place of the include line. If the file ends with an escaped `<newline>` the behavior is unspecified.

The file may itself contain further include lines. Implementations shall support nesting of include files up to a depth of at least 16.



## Makefile Execution

Makefile command lines shall be processed one at a time.

Makefile command lines can have one or more of the following prefixes: a <hyphen-minus> ('-'), an at-sign ('@'), or a <plus-sign> ('+'). These shall modify the way in which *make* processes the command.

- If the command prefix contains a <hyphen-minus>, or the `-i` option is present, or the special target **.IGNORE** has either the current target as a prerequisite or has no prerequisites, any error found while executing the command shall be ignored.
- @ If the command prefix contains an at-sign and the *make* utility command line `-n` option is not specified, or the `-s` option is present, or the special target **.SILENT** has either the current target as a prerequisite or has no prerequisites, the command shall not be written to standard output before it is executed.
- + If the command prefix contains a <plus-sign>, this indicates a makefile command line that shall be executed even if `-n`, `-q`, or `-t` is specified.

An *execution line* is built from the command line by removing any prefix characters. Except as described under the at-sign prefix, the execution line shall be written to the standard output, optionally preceded by a <tab>. The execution line shall then be executed by a shell as if it were passed as the argument to the *system()* interface, except that if errors are not being ignored then the shell `-e` option shall also be in effect. If errors are being ignored for the command (as a result of the `-i` option, a '-' command prefix, or a **.IGNORE** special target), the shell `-e` option shall not be in effect. The environment for the command being executed shall contain all of the variables in the environment of *make*.

By default, when *make* receives a non-zero status from the execution of a command, it shall terminate with an error message to standard error.

## Target Rules

Target rules are formatted as follows:

```
target [target...]: [prerequisite...][;command]
[<tab>command
<tab>command
...]
```

*line that does not begin with <tab>*

Target entries are specified by a <blank>-separated, non-null list of targets, then a <colon>, then a <blank>-separated, possibly empty list of prerequisites. Text following a <semicolon>, if any, and all following lines that begin with a <tab>, are makefile command lines to be executed to update the target. The first non-empty line that does not begin with a <tab> or '#' shall begin a new entry. Any comment line may begin a new entry.

Applications shall select target names from the set of characters consisting solely of periods, underscores, digits, and alphabets from the portable character set (see XBD Section 6.1, on page 125). Implementations may allow other characters in target names as extensions. The interpretation of targets containing the characters '%' and '"' is implementation-defined.

A target that has prerequisites, but does not have any commands, can be used to add to the prerequisite list for that target. Only one target rule for any given target can contain commands.

Lines that begin with one of the following are called *special targets* and control the operation of *make*:

- 98298 **.DEFAULT** If the makefile uses this special target, the application shall ensure that it is  
 98299 specified with commands, but without prerequisites. The commands shall be used  
 98300 by *make* if there are no other rules available to build a target.
- 98301 **.IGNORE** Prerequisites of this special target are targets themselves; this shall cause errors  
 98302 from commands associated with them to be ignored in the same manner as  
 98303 specified by the `-i` option. Subsequent occurrences of **.IGNORE** shall add to the  
 98304 list of targets ignoring command errors. If no prerequisites are specified, *make* shall  
 98305 behave as if the `-i` option had been specified and errors from all commands  
 98306 associated with all targets shall be ignored.
- 98307 **.POSIX** The application shall ensure that this special target is specified without  
 98308 prerequisites or commands. If it appears as the first non-comment line in the  
 98309 makefile, *make* shall process the makefile as specified by this section; otherwise, the  
 98310 behavior of *make* is unspecified.
- 98311 **.PRECIOUS** Prerequisites of this special target shall not be removed if *make* receives one of the  
 98312 asynchronous events explicitly described in the ASYNCHRONOUS EVENTS  
 98313 section. Subsequent occurrences of **.PRECIOUS** shall add to the list of precious  
 98314 files. If no prerequisites are specified, all targets in the makefile shall be treated as  
 98315 if specified with **.PRECIOUS**.
- 98316 XSI **.SCCS\_GET** The application shall ensure that this special target is specified without  
 98317 prerequisites. If this special target is included in a makefile, the commands  
 98318 specified with this target shall replace the default commands associated with this  
 98319 special target (see [Default Rules](#), on page 2969). The commands specified with this  
 98320 target are used to get all SCCS files that are not found in the current directory.
- 98321 When source files are named in a dependency list, *make* shall treat them just like  
 98322 any other target. Because the source file is presumed to be present in the directory,  
 98323 there is no need to add an entry for it to the makefile. When a target has no  
 98324 dependencies, but is present in the directory, *make* shall assume that that file is up-  
 98325 to-date. If, however, an SCCS file named **SCCS/s.source\_file** is found for a target  
 98326 *source\_file*, *make* compares the timestamp of the target file with that of the  
 98327 **SCCS/s.source\_file** to ensure the target is up-to-date. If the target is missing, or if  
 98328 the SCCS file is newer, *make* shall automatically issue the commands specified for  
 98329 the **.SCCS\_GET** special target to retrieve the most recent version. However, if the  
 98330 target is writable by anyone, *make* shall not retrieve a new version.
- 98331 **.SILENT** Prerequisites of this special target are targets themselves; this shall cause  
 98332 commands associated with them not to be written to the standard output before  
 98333 they are executed. Subsequent occurrences of **.SILENT** shall add to the list of  
 98334 targets with silent commands. If no prerequisites are specified, *make* shall behave  
 98335 as if the `-s` option had been specified and no commands or touch messages  
 98336 associated with any target shall be written to standard output.
- 98337 **.SUFFIXES** Prerequisites of **.SUFFIXES** shall be appended to the list of known suffixes and are  
 98338 used in conjunction with the inference rules (see [Inference Rules](#), on page 2966). If  
 98339 **.SUFFIXES** does not have any prerequisites, the list of known suffixes shall be  
 98340 cleared.
- 98341 The special targets **.IGNORE**, **.POSIX**, **.PRECIOUS**, **.SILENT**, and **.SUFFIXES** shall be specified  
 98342 without commands.
- 98343 Targets with names consisting of a leading <period> followed by the uppercase letters "POSIX"  
 98344 and then any other characters are reserved for future standardization. Targets with names



consisting of a leading <period> followed by one or more uppercase letters are reserved for implementation extensions.

## Macros

Macro definitions are in the form:

```
string1 = [string2]
```

The macro named *string1* is defined as having the value of *string2*, where *string2* is defined as all characters, if any, after the <equals-sign>, up to a comment character ('#') or an unescaped <newline>. Any <blank> characters immediately before or after the <equals-sign> shall be ignored.

Applications shall select macro names from the set of characters consisting solely of periods, underscores, digits, and alphabets from the portable character set (see XBD Section 6.1, on page 125). A macro name shall not contain an <equals-sign>. Implementations may allow other characters in macro names as extensions.

Macros can appear anywhere in the makefile. Macro expansions using the forms  $\$(string1)$  or  $\${string1}$  shall be replaced by *string2*, as follows:

- Macros in target lines shall be evaluated when the target line is read.
- Macros in makefile command lines shall be evaluated when the command is executed.
- Macros in the string before the <equals-sign> in a macro definition shall be evaluated when the macro assignment is made.
- Macros after the <equals-sign> in a macro definition shall not be evaluated until the defined macro is used in a rule or command, or before the <equals-sign> in a macro definition.

The parentheses or braces are optional if *string1* is a single character. The macro  $\$\$$  shall be replaced by the single character '\$'. If *string1* in a macro expansion contains a macro expansion, the results are unspecified.

Macro expansions using the forms  $\$(string1[:subst1=[subst2]])$  or  $\${string1[:subst1=[subst2]]}$  can be used to replace all occurrences of *subst1* with *subst2* when the macro substitution is performed. The *subst1* to be replaced shall be recognized when it is a suffix at the end of a word in *string1* (where a *word*, in this context, is defined to be a string delimited by the beginning of the line, a <blank>, or a <newline>). If *string1* in a macro expansion contains a macro expansion, the results are unspecified. If a <percent-sign> character appears as part of *subst1* or *subst2* after any macros have been recursively expanded, the results are unspecified. +

Macro expansions in *string1* of macro definition lines shall be evaluated when read. Macro expansions in *string2* of macro definition lines shall be performed when the macro identified by *string1* is expanded in a rule or command.

Macro definitions shall be taken from the following sources, in the following logical order, before the makefile(s) are read.

1. Macros specified on the *make* utility command line, in the order specified on the command line. It is unspecified whether the internal macros defined in **Internal Macros** (on page 2968) are accepted from this source.
2. Macros defined by the *MAKEFLAGS* environment variable, in the order specified in the environment variable. It is unspecified whether the internal macros defined in **Internal Macros** (on page 2968) are accepted from this source.

3. The contents of the environment, excluding the *MAKEFLAGS* and *SHELL* variables and including the variables with null values.

4. Macros defined in the inference rules built into *make*.

Macro definitions from these sources shall not override macro definitions from a lower-numbered source. Macro definitions from a single source (for example, the *make* utility command line, the *MAKEFLAGS* environment variable, or the other environment variables) shall override previous macro definitions from the same source.

Macros defined in the makefile(s) shall override macro definitions that occur before them in the makefile(s) and macro definitions from source 4. If the *-e* option is not specified, macros defined in the makefile(s) shall override macro definitions from source 3. Macros defined in the makefile(s) shall not override macro definitions from source 1 or source 2.

Before the makefile(s) are read, all of the *make* utility command line options (except *-f* and *-p*) and *make* utility command line macro definitions (except any for the *MAKEFLAGS* macro), not already included in the *MAKEFLAGS* macro, shall be added to the *MAKEFLAGS* macro, quoted in an implementation-defined manner such that when *MAKEFLAGS* is read by another instance of the *make* command, the original macro's value is recovered. Other implementation-defined options and macros may also be added to the *MAKEFLAGS* macro. If this modifies the value of the *MAKEFLAGS* macro, or, if the *MAKEFLAGS* macro is modified at any subsequent time, the *MAKEFLAGS* environment variable shall be modified to match the new value of the *MAKEFLAGS* macro. The result of setting *MAKEFLAGS* in the Makefile is unspecified.

Before the makefile(s) are read, all of the *make* utility command line macro definitions (except the *MAKEFLAGS* macro or the *SHELL* macro) shall be added to the environment of *make*. Other implementation-defined variables may also be added to the environment of *make*. Macros defined by the *MAKEFLAGS* environment variable and macros defined in the makefile(s) shall not be added to the environment of *make* if they are not already in its environment. With the exception of *SHELL* (see below), it is unspecified whether macros defined in these ways update the value of an environment variable that already exists in the environment of *make*.

The **SHELL** macro shall be treated specially. It shall be provided by *make* and set to the pathname of the shell command language interpreter (see *sh*). The *SHELL* environment variable shall not affect the value of the **SHELL** macro. If **SHELL** is defined in the makefile or is specified on the command line, it shall replace the original value of the **SHELL** macro, but shall not affect the *SHELL* environment variable. Other effects of defining **SHELL** in the makefile or on the command line are implementation-defined.

## Inference Rules

Inference rules are formatted as follows:

```
target:
<tab>command
[<tab>command]
...

line that does not begin with <tab> or #
```

The application shall ensure that the *target* portion is a valid target name (see [Target Rules](#), on page 2963) of the form *.s2* or *.s1.s2* (where *.s1* and *.s2* are suffixes that have been given as prerequisites of the **SUFFIXES** special target and *s1* and *s2* do not contain any *<slash>* or *<period>* characters.) If there is only one *<period>* in the target, it is a single-suffix inference rule. Targets with two periods are double-suffix inference rules. Inference rules can have only one target before the *<colon>*.

The application shall ensure that the makefile does not specify prerequisites for inference rules; no characters other than white space shall follow the <colon> in the first line, except when creating the *empty rule*, described below. Prerequisites are inferred, as described below.

Inference rules can be redefined. A target that matches an existing inference rule shall overwrite the old inference rule. An empty rule can be created with a command consisting of simply a <semicolon> (that is, the rule still exists and is found during inference rule search, but since it is empty, execution has no effect). The empty rule can also be formatted as follows:

```
rule: ;
```

where zero or more <blank> characters separate the <colon> and <semicolon>.

The *make* utility uses the suffixes of targets and their prerequisites to infer how a target can be made up-to-date. A list of inference rules defines the commands to be executed. By default, *make* contains a built-in set of inference rules. Additional rules can be specified in the makefile.

The special target **.SUFFIXES** contains as its prerequisites a list of suffixes that shall be used by the inference rules. The order in which the suffixes are specified defines the order in which the inference rules for the suffixes are used. New suffixes shall be appended to the current list by specifying a **.SUFFIXES** special target in the makefile. A **.SUFFIXES** target with no prerequisites shall clear the list of suffixes. An empty **.SUFFIXES** target followed by a new **.SUFFIXES** list is required to change the order of the suffixes.

Normally, the user would provide an inference rule for each suffix. The inference rule to update a target with a suffix **.s1** from a prerequisite with a suffix **.s2** is specified as a target **.s2.s1**. The internal macros provide the means to specify general inference rules (see [Internal Macros](#), on page 2968).

When no target rule is found to update a target, the inference rules shall be checked. The suffix of the target (**.s1**) to be built is compared to the list of suffixes specified by the **.SUFFIXES** special targets. If the **.s1** suffix is found in **.SUFFIXES**, the inference rules shall be searched in the order defined for the first **.s2.s1** rule whose prerequisite file (**\$.s2**) exists. If the target is out-of-date with respect to this prerequisite, the commands for that inference rule shall be executed.

If the target to be built does not contain a suffix and there is no rule for the target, the single suffix inference rules shall be checked. The single-suffix inference rules define how to build a target if a file is found with a name that matches the target name with one of the single suffixes appended. A rule with one suffix **.s2** is the definition of how to build *target* from **target.s2**. The other suffix (**.s1**) is treated as null.

A <tilde> ('~') in the above rules refers to an SCCS file in the current directory. Thus, the rule **.c~.o** would transform an SCCS C-language source file into an object file (**.o**). Because the **s.** of the SCCS files is a prefix, it is incompatible with *make's* suffix point of view. Hence, the '~' is a way of changing any file reference into an SCCS file reference.

## Libraries

If a target or prerequisite contains parentheses, it shall be treated as a member of an archive library. For the *lib(member.o)* expression *lib* refers to the name of the archive library and *member.o* to the member name. The application shall ensure that the member is an object file with the **.o** suffix. The modification time of the expression is the modification time for the member as kept in the archive library; see [ar](#). The **.a** suffix shall refer to an archive library. The **.s2.a** rule shall be used to update a member in the library from a file with a suffix **.s2**.

## Internal Macros

The *make* utility shall maintain five internal macros that can be used in target and inference rules. In order to clearly define the meaning of these macros, some clarification of the terms *target rule*, *inference rule*, *target*, and *prerequisite* is necessary.

Target rules are specified by the user in a makefile for a particular target. Inference rules are user-specified or *make*-specified rules for a particular class of target name. Explicit prerequisites are those prerequisites specified in a makefile on target lines. Implicit prerequisites are those prerequisites that are generated when inference rules are used. Inference rules are applied to implicit prerequisites or to explicit prerequisites that do not have target rules defined for them in the makefile. Target rules are applied to targets specified in the makefile.

Before any target in the makefile is updated, each of its prerequisites (both explicit and implicit) shall be updated. This shall be accomplished by recursively processing each prerequisite. Upon recursion, each prerequisite shall become a target itself. Its prerequisites in turn shall be processed recursively until a target is found that has no prerequisites, or further recursion would require applying two inference rules one immediately after the other, at which point the recursion shall stop. As an extension, implementations may continue recursion when two or more successive inference rules need to be applied; however, if there are multiple different chains of such rules that could be used to create the target, it is unspecified which chain is used. The recursion shall then back up, updating each target as it goes.

In the definitions that follow, the word *target* refers to one of:

- A target specified in the makefile
- An explicit prerequisite specified in the makefile that becomes the target when *make* processes it during recursion
- An implicit prerequisite that becomes a target when *make* processes it during recursion

In the definitions that follow, the word *prerequisite* refers to one of the following:

- An explicit prerequisite specified in the makefile for a particular target
- An implicit prerequisite generated as a result of locating an appropriate inference rule and corresponding file that matches the suffix of the target

The five internal macros are:

**\$@** The **\$@** shall evaluate to the full target name of the current target, or the archive filename part of a library archive target. It shall be evaluated for both target and inference rules.

For example, in the **.c.a** inference rule, **\$@** represents the out-of-date **.a** file to be built. Similarly, in a makefile target rule to build **lib.a** from **file.c**, **\$@** represents the out-of-date **lib.a**.

**\$\$** The **\$\$** macro shall be evaluated only when the current target is an archive library member of the form *libname(member.o)*. In these cases, **\$@** shall evaluate to *libname* and **\$\$** shall evaluate to *member.o*. The **\$\$** macro shall be evaluated for both target and inference rules.

For example, in a makefile target rule to build **lib.a(file.o)**, **\$\$** represents **file.o**, as opposed to **\$@**, which represents **lib.a**.

**\$?** The **\$?** macro shall evaluate to the list of prerequisites that are newer than the current target. It shall be evaluated for both target and inference rules.

98520 For example, in a makefile target rule to build *prog* from **file1.o**, **file2.o**, and **file3.o**, and  
 98521 where *prog* is not out-of-date with respect to **file1.o**, but is out-of-date with respect to  
 98522 **file2.o** and **file3.o**, **\$?** represents **file2.o** and **file3.o**.

98523 **\$<** In an inference rule, the **\$<** macro shall evaluate to the filename whose existence  
 98524 allowed the inference rule to be chosen for the target. In the **.DEFAULT** rule, the **\$<**  
 98525 macro shall evaluate to the current target name. The meaning of the **\$<** macro shall be  
 98526 otherwise unspecified.

98527 For example, in the **.c.a** inference rule, **\$<** represents the prerequisite **.c** file.

98528 **\$\*** The **\$\*** macro shall evaluate to the current target name with its suffix deleted. It shall be  
 98529 evaluated at least for inference rules.

98530 For example, in the **.c.a** inference rule, **\$\*.o** represents the out-of-date **.o** file that  
 98531 corresponds to the prerequisite **.c** file.

98532 Each of the internal macros has an alternative form. When an uppercase '**D**' or '**F**' is appended  
 98533 to any of the macros, the meaning shall be changed to the *directory part* for '**D**' and *filename part*  
 98534 for '**F**'. The directory part is the path prefix of the file without a trailing **<slash>**; for the current  
 98535 directory, the directory part is '**. .**'. When the **\$?** macro contains more than one prerequisite  
 98536 filename, the **\$(?D)** and **\$(?F)** (or **\${?D}** and **\${?F}**) macros expand to a list of directory name parts  
 98537 and filename parts respectively.

98538 For the target *lib(member.o)* and the **s2.a** rule, the internal macros shall be defined as:

98539 **\$<** *member.s2*

98540 **\$\*** *member*

98541 **\$@** *lib*

98542 **\$?** *member.s2*

98543 **\$%** *member.o*

## 98544 Default Rules

98545 The default rules for *make* shall achieve results that are the same as if the following were used.  
 98546 Implementations that do not support the C-Language Development Utilities option may omit  
 98547 **CC**, **CFLAGS**, **YACC**, **YFLAGS**, **LEX**, **LFLAGS**, **LDFLAGS**, and the **.c**, **.y**, and **.l** inference rules.  
 98548 Implementations that do not support FORTRAN may omit **FC**, **FFLAGS**, and the **.f** inference  
 98549 rules. Implementations may provide additional macros and rules.

## 98550 SPECIAL TARGETS

98551 XSI **.SCCS\_GET:** *sccs \$(SCCSFLAGS) get \$(SCCSGETFLAGS) \$@*

98552 XSI **.SUFFIXES:** *.o .c .y .l .a .sh .f .c~ .y~ .l~ .sh~ .f~*

## 98553 MACROS

98554 **MAKE=***make*

98555 **AR=***ar*

98556 **ARFLAGS=***--rv*

98557 **YACC=***yacc*

98558 **YFLAGS=**

98559 **LEX=***lex*

98560 **LFLAGS=**

```

98561      LDFLAGS=
98562      CC=c99
98563      CFLAGS=-O 1
98564      FC=fort77
98565      FFLAGS=-O 1
98566 XSI    GET=get
98567      GFLAGS=
98568      SCCSFLAGS=
98569      SCCSGETFLAGS=-s

```

#### 98570 SINGLE SUFFIX RULES

```

98571      .c:
98572          $(CC) $(CFLAGS) $(LDFLAGS) -o $@ $<
98573      .f:
98574          $(FC) $(FFLAGS) $(LDFLAGS) -o $@ $<
98575      .sh:
98576          cp $< $@
98577          chmod a+x $@
98578 XSI    .c~:
98579          $(GET) $(GFLAGS) -p $< > $*.c
98580          $(CC) $(CFLAGS) $(LDFLAGS) -o $@ $*.c
98581      .f~:
98582          $(GET) $(GFLAGS) -p $< > $*.f
98583          $(FC) $(FFLAGS) $(LDFLAGS) -o $@ $*.f
98584      .sh~:
98585          $(GET) $(GFLAGS) -p $< > $*.sh
98586          cp $*.sh $@
98587          chmod a+x $@

```

#### 98588 DOUBLE SUFFIX RULES

```

98589      .c.o:
98590          $(CC) $(CFLAGS) -c $<
98591      .f.o:
98592          $(FC) $(FFLAGS) -c $<
98593      .y.o:
98594          $(YACC) $(YFLAGS) $<
98595          $(CC) $(CFLAGS) -c y.tab.c
98596          rm -f y.tab.c
98597          mv y.tab.o $@
98598      .l.o:
98599          $(LEX) $(LFLAGS) $<
98600          $(CC) $(CFLAGS) -c lex.yy.c
98601          rm -f lex.yy.c
98602          mv lex.yy.o $@
98603      .y.c:

```

```

98604         $(YACC) $(YFLAGS) $<
98605         mv y.tab.c $@

98606     .l.c:
98607         $(LEX) $(LFLAGS) $<
98608         mv lex.yy.c $@

98609 XSI     .c~.o:
98610         $(GET) $(GFLAGS) -p $< > $*.c
98611         $(CC) $(CFLAGS) -c $*.c

98612     .f~.o:
98613         $(GET) $(GFLAGS) -p $< > $*.f
98614         $(FC) $(FFLAGS) -c $*.f

98615     .y~.o:
98616         $(GET) $(GFLAGS) -p $< > $*.y
98617         $(YACC) $(YFLAGS) $*.y
98618         $(CC) $(CFLAGS) -c y.tab.c
98619         rm -f y.tab.c
98620         mv y.tab.o $@

98621     .l~.o:
98622         $(GET) $(GFLAGS) -p $< > $*.l
98623         $(LEX) $(LFLAGS) $*.l
98624         $(CC) $(CFLAGS) -c lex.yy.c
98625         rm -f lex.yy.c
98626         mv lex.yy.o $@

98627     .y~.c:
98628         $(GET) $(GFLAGS) -p $< > $*.y
98629         $(YACC) $(YFLAGS) $*.y
98630         mv y.tab.c $@

98631     .l~.c:
98632         $(GET) $(GFLAGS) -p $< > $*.l
98633         $(LEX) $(LFLAGS) $*.l
98634         mv lex.yy.c $@

98635     .c.a:
98636         $(CC) -c $(CFLAGS) $<
98637         $(AR) $(ARFLAGS) $@ $*.o
98638         rm -f $*.o

98639     .f.a:
98640         $(FC) -c $(FFLAGS) $<
98641         $(AR) $(ARFLAGS) $@ $*.o
98642         rm -f $*.o

```

## EXIT STATUS

When the **-q** option is specified, the *make* utility shall exit with one of the following values:

- 0 Successful completion.
- 1 The target was not up-to-date.



98647 >1 An error occurred.

98648 When the **-q** option is not specified, the *make* utility shall exit with one of the following values:

98649 0 Successful completion.

98650 >0 An error occurred.

## 98651 CONSEQUENCES OF ERRORS

98652 Default.

## 98653 APPLICATION USAGE

98654 If there is a source file (such as *./source.c*) and there are two SCCS files corresponding to it  
 98655 (*./s.source.c* and *./SCCS/s.source.c*), on XSI-conformant systems *make* uses the SCCS file in the  
 98656 current directory. However, users are advised to use the underlying SCCS utilities (*admin*, *delta*,  
 98657 *get*, and so on) or the *scs* utility for all source files in a given directory. If both forms are used for  
 98658 a given source file, future developers are very likely to be confused.

98659 It is incumbent upon portable makefiles to specify the **.POSIX** special target in order to  
 98660 guarantee that they are not affected by local extensions.

98661 The **-k** and **-S** options are both present so that the relationship between the command line, the  
 98662 *MAKEFLAGS* variable, and the makefile can be controlled precisely. If the **k** flag is passed in  
 98663 *MAKEFLAGS* and a command is of the form:

98664 `$(MAKE) -S foo`

98665 then the default behavior is restored for the child *make*.

98666 When the **-n** option is specified, it is always added to *MAKEFLAGS*. This allows a recursive  
 98667 *make -n target* to be used to see all of the action that would be taken to update *target*.

98668 Because of widespread historical practice, interpreting a <number-sign> ('#') inside a variable  
 98669 as the start of a comment has the unfortunate side-effect of making it impossible to place a  
 98670 <number-sign> in a variable, thus forbidding something like:

98671 `CFLAGS = "-D COMMENT_CHAR='#'"`

98672 Many historical *make* utilities stop chaining together inference rules when an intermediate target  
 98673 is nonexistent. For example, it might be possible for a *make* to determine that both *.y.c* and *.c.o*  
 98674 could be used to convert a *.y* to a *.o*. Instead, in this case, *make* requires the use of a *.y.o* rule.

98675 The best way to provide portable makefiles is to include all of the rules needed in the makefile  
 98676 itself. The rules provided use only features provided by other parts of this volume of  
 98677 POSIX.1-2008. The default rules include rules for optional commands in this volume of  
 98678 POSIX.1-2008. Only rules pertaining to commands that are provided are needed in an  
 98679 implementation's default set.

98680 Macros used within other macros are evaluated when the new macro is used rather than when  
 98681 the new macro is defined. Therefore:

98682 `MACRO = value1`  
 98683 `NEW = $(MACRO)`  
 98684 `MACRO = value2`

98685 `target:`  
 98686 `echo $(NEW)`

98687 would produce *value2* and not *value1* since **NEW** was not expanded until it was needed in the  
 98688 *echo* command line.



Some historical applications have been known to intermix *target\_name* and *macro=name* operands on the command line, expecting that all of the macros are processed before any of the targets are dealt with. Conforming applications do not do this, although some backwards-compatibility support may be included in some implementations.

The following characters in filenames may give trouble: `'=`, `':`, `'\``, single-quote, and `'@`. In include filenames, pattern matching characters and `'"` should also be avoided, as they may be treated as special by some implementations.

For inference rules, the description of `$<` and `$?` seem similar. However, an example shows the minor difference. In a makefile containing:

```
foo.o: foo.h
```

if **foo.h** is newer than **foo.o**, yet **foo.c** is older than **foo.o**, the built-in rule to make **foo.o** from **foo.c** is used, with `$<` equal to **foo.c** and `$?` equal to **foo.h**. If **foo.c** is also newer than **foo.o**, `$<` is equal to **foo.c** and `$?` is equal to **foo.h foo.c**.

As a consequence of the general rules for target updating, a useful special case is that if a target has no prerequisites and no commands, and the target of the rule is a nonexistent file, then *make* acts as if this target has been updated whenever its rule is run.

**Note:** This implies that all targets depending on this one will always have their commands run.

Shell command sequences like `make; cp original copy; make` may have problems on filesystems where the timestamp resolution is the minimum (1 second) required by the standard and where *make* considers identical timestamps to be up-to-date. Conversely, rules like `copy: original; cp -p original copy` will result in redundant work on *make* implementations that consider identical timestamps to be out-of-date.

This standard does not specify precedence between macro definition and include directives. Thus, the behavior of:

```
include =foo.mk
```

is unspecified. To define a variable named `include`, either the white space before the `<equal-sign>` should be removed, or another macro should be used, as in:

```
INCLUDE_NAME = include
$(INCLUDE_NAME) =foo.mk
```

On the other hand, if the intent is to include a file which starts with an `<equal-sign>`, either the filename should be changed to `./=foo.mk`, or the makefile should be written as:

```
INCLUDE_FILE = =foo.mk
include $(INCLUDE_FILE)
```

## EXAMPLES

1. The following command:

```
make
```

makes the first target found in the makefile.

2. The following command:

```
make junk
```

makes the target **junk**.

3. The following makefile says that **pgm** depends on two files, **a.o** and **b.o**, and that they in turn depend on their corresponding source files (**a.c** and **b.c**), and a common file **incl.h**:

```
.POSIX:
pgm: a.o b.o
    c99 a.o b.o -o pgm
a.o: incl.h a.c
    c99 -c a.c
b.o: incl.h b.c
    c99 -c b.c
```

4. An example for making optimized **.o** files from **.c** files is:

```
.c.o:
    c99 -c -O 1 $*.c

or:

.c.o:
    c99 -c -O 1 $<
```

5. The most common use of the archive interface follows. Here, it is assumed that the source files are all C-language source:

```
lib: lib(file1.o) lib(file2.o) lib(file3.o)
    @echo lib is now up-to-date
```

The **.c.a** rule is used to make **file1.o**, **file2.o**, and **file3.o** and insert them into **lib**.

The treatment of escaped <newline> characters throughout the makefile is historical practice. For example, the inference rule:

```
.c.o\
:

works, and the macro:

f= bar baz\
    biz
a:
    echo ==$f==

echoes "==bar baz biz==".
```

If **\$\$?** were:

```
/usr/include/stdio.h /usr/include/unistd.h foo.h

then $(?D) would be:

/usr/include /usr/include .

and $(?F) would be:

stdio.h unistd.h foo.h
```

6. The contents of the built-in rules can be viewed by running:

```
make -p -f /dev/null 2>/dev/null
```

## RATIONALE

The *make* utility described in this volume of POSIX.1-2008 is intended to provide the means for changing portable source code into executables that can be run on an POSIX.1-2008-conforming system. It reflects the most common features present in System V and BSD *makes*.

Historically, the *make* utility has been an especially fertile ground for vendor and research organization-specific syntax modifications and extensions. Examples include:

- Syntax supporting parallel execution (such as from various multi-processor vendors, GNU, and others)
- Additional “operators” separating targets and their prerequisites (System V, BSD, and others)
- Specifying that command lines containing the strings “`${MAKE}`” and “`$(MAKE)`” are executed when the `-n` option is specified (GNU and System V)
- Modifications of the meaning of internal macros when referencing libraries (BSD and others)
- Using a single instance of the shell for all of the command lines of the target (BSD and others)
- Allowing <space> characters as well as <tab> characters to delimit command lines (BSD)
- Adding C preprocessor-style “include” and “ifdef” constructs (System V, GNU, BSD, and others)
- Remote execution of command lines (Sprite and others)
- Specifying additional special targets (BSD, System V, and most others)
- Specifying an alternate shell to use to process commands.

+

Additionally, many vendors and research organizations have rethought the basic concepts of *make*, creating vastly extended, as well as completely new, syntaxes. Each of these versions of *make* fulfills the needs of a different community of users; it is unreasonable for this volume of POSIX.1-2008 to require behavior that would be incompatible (and probably inferior) to historical practice for such a community.

In similar circumstances, when the industry has enough sufficiently incompatible formats as to make them irreconcilable, this volume of POSIX.1-2008 has followed one or both of two courses of action. Commands have been renamed (*cksum*, *echo*, and *pax*) and/or command line options have been provided to select the desired behavior (*grep*, *od*, and *pax*).

Because the syntax specified for the *make* utility is, by and large, a subset of the syntaxes accepted by almost all versions of *make*, it was decided that it would be counter-productive to change the name. And since the makefile itself is a basic unit of portability, it would not be completely effective to reserve a new option letter, such as *make -P*, to achieve the portable behavior. Therefore, the special target **.POSIX** was added to the makefile, allowing users to specify “standard” behavior. This special target does not preclude extensions in the *make* utility, nor does it preclude such extensions being used by the makefile specifying the target; it does, however, preclude any extensions from being applied that could alter the behavior of previously valid syntax; such extensions must be controlled via command line options or new special targets. It is incumbent upon portable makefiles to specify the **.POSIX** special target in order to guarantee that they are not affected by local extensions.

The portable version of *make* described in this reference page is not intended to be the state-of-the-art software generation tool and, as such, some newer and more leading-edge features have

not been included. An attempt has been made to describe the portable makefile in a manner that does not preclude such extensions as long as they do not disturb the portable behavior described here.

When the `-n` option is specified, it is always added to `MAKEFLAGS`. This allows a recursive `make -n target` to be used to see all of the action that would be taken to update `target`.

The definition of `MAKEFLAGS` allows both the System V letter string and the BSD command line formats. The two formats are sufficiently different to allow implementations to support both without ambiguity.

Early proposals stated that an “unquoted” `<number-sign>` was treated as the start of a comment. The `make` utility does not pay any attention to quotes. A `<number-sign>` starts a comment regardless of its surroundings.

The text about “other implementation-defined pathnames may also be tried” in addition to `./makefile` and `./Makefile` is to allow such extensions as `SCCS/s.Makefile` and other variations. It was made an implementation-defined requirement (as opposed to unspecified behavior) to highlight surprising implementations that might select something unexpected like `/etc/Makefile`. XSI-conformant systems also try `./s.makefile`, `SCCS/s.makefile`, `./s.Makefile`, and `SCCS/s.Makefile`.

Early proposals contained the macro `NPROC` as a means of specifying that `make` should use `n` processes to do the work required. While this feature is a valuable extension for many systems, it is not common usage and could require other non-trivial extensions to makefile syntax. This extension is not required by this volume of POSIX.1-2008, but could be provided as a compatible extension. The macro `PARALLEL` is used by some historical systems with essentially the same meaning (but without using a name that is a common system limit value). It is suggested that implementors recognize the existing use of `NPROC` and/or `PARALLEL` as extensions to `make`.

The default rules are based on System V. The default `CC=` value is `c99` instead of `cc` because this volume of POSIX.1-2008 does not standardize the utility named `cc`. Thus, every conforming application would be required to define `CC=c99` to expect to run. There is no advantage conferred by the hope that the makefile might hit the “preferred” compiler because this cannot be guaranteed to work. Also, since the portable makescript can only use the `c99` options, no advantage is conferred in terms of what the script can do. It is a quality-of-implementation issue as to whether `c99` is as valuable as `cc`.

The `-d` option to `make` is frequently used to produce debugging information, but is too implementation-defined to add to this volume of POSIX.1-2008.

The `-p` option is not passed in `MAKEFLAGS` on most historical implementations and to change this would cause many implementations to break without sufficiently increased portability.

Commands that begin with a `<plus-sign>` (`'+'`) are executed even if the `-n` option is present. Based on the GNU version of `make`, the behavior of `-n` when the `<plus-sign>` prefix is encountered has been extended to apply to `-q` and `-t` as well. However, the System V convention of forcing command execution with `-n` when the command line of a target contains either of the strings `"$(MAKE)"` or `"${MAKE}"` has not been adopted. This functionality appeared in early proposals, but the danger of this approach was pointed out with the following example of a portion of a makefile:

```
subdir:
    cd subdir; rm all_the_files; $(MAKE)
```

The loss of the System V behavior in this case is well-balanced by the safety afforded to other makefiles that were not aware of this situation. In any event, the command line `<plus-sign>`

prefix can provide the desired functionality.

The double <colon> in the target rule format is supported in BSD systems to allow more than one target line containing the same target name to have commands associated with it. Since this is not functionality described in the SVID or XPG3 it has been allowed as an extension, but not mandated.

The default rules are provided with text specifying that the built-in rules shall be the same as if the listed set were used. The intent is that implementations should be able to use the rules without change, but will be allowed to alter them in ways that do not affect the primary behavior.

One point of discussion was whether to drop the default rules list from this volume of POSIX.1-2008. They provide convenience, but do not enhance portability of applications. The prime benefit is in portability of users who wish to type *make command* and have the command build from a **command.c** file.

The historical *MAKESHELL* feature, and related features provided by other *make* implementations, were omitted. In some implementations it is used to let a user override the shell to be used to run *make* commands. This was confusing; for a portable *make*, the shell should be chosen by the makefile writer. Further, a makefile writer cannot require an alternate shell to be used and still consider the makefile portable. While it would be possible to standardize a mechanism for specifying an alternate shell, existing implementations do not agree on such a mechanism, and makefile writers can already invoke an alternate shell by specifying the shell name in the rule for a target; for example:

```
python -c "foo"
```

The *make* utilities in most historical implementations process the prerequisites of a target in left-to-right order, and the makefile format requires this. It supports the standard idiom used in many makefiles that produce *yacc* programs; for example:

```
foo: y.tab.o lex.o main.o
    $(CC) $(CFLAGS) -o $@ t.tab.o lex.o main.o
```

In this example, if *make* chose any arbitrary order, the **lex.o** might not be made with the correct **y.tab.h**. Although there may be better ways to express this relationship, it is widely used historically. Implementations that desire to update prerequisites in parallel should require an explicit extension to *make* or the makefile format to accomplish it, as described previously.

The algorithm for determining a new entry for target rules is partially unspecified. Some historical *makes* allow comment lines (including blank and empty lines) within the collection of commands marked by leading <tab> characters. A conforming makefile must ensure that each command starts with a <tab>, but implementations are free to ignore comments without triggering the start of a new entry.

The ASYNCHRONOUS EVENTS section includes having SIGTERM and SIGHUP, along with the more traditional SIGINT and SIGQUIT, remove the current target unless directed not to do so. SIGTERM and SIGHUP were added to parallel other utilities that have historically cleaned up their work as a result of these signals. When *make* receives any signal other than SIGQUIT, it is required to resend itself the signal it received so that it exits with a status that reflects the signal. The results from SIGQUIT are partially unspecified because, on systems that create **core** files upon receipt of SIGQUIT, the **core** from *make* would conflict with a **core** file from the command that was running when the SIGQUIT arrived. The main concern was to prevent damaged files from appearing up-to-date when *make* is rerun.

The **.PRECIOUS** special target was extended to affect all targets globally (by specifying no

prerequisites). The **.IGNORE** and **.SILENT** special targets were extended to allow prerequisites; it was judged to be more useful in some cases to be able to turn off errors or echoing for a list of targets than for the entire makefile. These extensions to *make* in System V were made to match historical practice from the BSD *make*.

Macros are not exported to the environment of commands to be run. This was never the case in any historical *make* and would have serious consequences. The environment is the same as the environment to *make* except that **MAKEFLAGS** and macros defined on the *make* command line are added, and except that macros defined by the **MAKEFLAGS** environment variable and macros defined in the makefile(s) may update the value of an existing environment variable (other than **SHELL**).

Some implementations do not use *system()* for all command lines, as required by the portable makefile format; as a performance enhancement, they select lines without shell metacharacters for direct execution by *execve()*. There is no requirement that *system()* be used specifically, but merely that the same results be achieved. The metacharacters typically used to bypass the direct *execve()* execution have been any of:

```
= | ^ ( ) ; & < > * ? [ ] : $ ` ' " \ \n
```

The default in some advanced versions of *make* is to group all the command lines for a target and execute them using a single shell invocation; the System V method is to pass each line individually to a separate shell. The single-shell method has the advantages in performance and the lack of a requirement for many continued lines. However, converting to this newer method has caused portability problems with many historical makefiles, so the behavior with the POSIX makefile is specified to be the same as that of System V. It is suggested that the special target **.ONESHELL** be used as an implementation extension to achieve the single-shell grouping for a target or group of targets.

Novice users of *make* have had difficulty with the historical need to start commands with a <tab>. Since it is often difficult to discern differences between <tab> and <space> characters on terminals or printed listings, confusing bugs can arise. In early proposals, an attempt was made to correct this problem by allowing leading <blank> characters instead of <tab> characters. However, implementors reported many makefiles that failed in subtle ways following this change, and it is difficult to implement a *make* that unambiguously can differentiate between macro and command lines. There is extensive historical practice of allowing leading <space> characters before macro definitions. Forcing macro lines into column 1 would be a significant backwards-compatibility problem for some makefiles. Therefore, historical practice was restored.

There is substantial variation in the handling of include lines by different implementations. However, there is enough commonality for the standard to be able to specify a minimum set of requirements that allow the feature to be used portably. Known variations have been explicitly called out as unspecified behavior in the description.

The System V dynamic dependency feature was not included. It would support:

```
cat: $$@.c
```

that would expand to;

```
cat: cat.c
```

This feature exists only in the new version of System V *make* and, while useful, is not in wide usage. This means that macros are expanded twice for prerequisites: once at makefile parse time and once at target update time.

Consideration was given to adding metarules to the POSIX *make*. This would make **%.o: %.c** the



same as **.c.o.**. This is quite useful and available from some vendors, but it would cause too many changes to this *make* to support. It would have introduced rule chaining and new substitution rules. However, the rules for target names have been set to reserve the '%' and '"' characters. These are traditionally used to implement metarules and quoting of target names, respectively. Implementors are strongly encouraged to use these characters only for these purposes.

A request was made to extend the suffix delimiter character from a <period> to any character. The metarules feature in newer *makes* solves this problem in a more general way. This volume of POSIX.1-2008 is staying with the more conservative historical definition.

The standard output format for the **-p** option is not described because it is primarily a debugging option and because the format is not generally useful to programs. In historical implementations the output is not suitable for use in generating makefiles. The **-p** format has been variable across historical implementations. Therefore, the definition of **-p** was only to provide a consistently named option for obtaining *make* script debugging information.

Some historical implementations have not cleared the suffix list with **-r**.

Implementations should be aware that some historical applications have intermixed *target\_name* and *macro=value* operands on the command line, expecting that all of the macros are processed before any of the targets are dealt with. Conforming applications do not do this, but some backwards-compatibility support may be warranted.

Empty inference rules are specified with a <semicolon> command rather than omitting all commands, as described in an early proposal. The latter case has no traditional meaning and is reserved for implementation extensions, such as in GNU *make*.

Earlier versions of this standard defined comment lines only as lines with '#' as the first character. Many places then talked about comments, blank lines, and empty lines; but some places inadvertently only mentioned comments when blank lines and empty lines had also been accepted in all known implementations. The standard now defines comment lines to be blank lines, empty lines, and lines starting with a '#' character and explicitly lists cases where blank lines and empty lines are not acceptable.

On most historic systems, the *make* utility considered a target with a prerequisite that had an identical timestamp as up-to-date. The HP-UX implementation of *make* treated it as out-of-date. The standard now allows either behavior, but implementations are encouraged to follow the example set by HP-UX. This is especially important on file systems where the timestamp resolution is the minimum (1 second) required by the standard. All implementations of *make* should make full use of the finest timestamp resolution available on the file systems holding targets and prerequisites to ensure that targets are up-to-date even for prerequisite files with timestamps that were updated within the same second. However, if the timestamp resolutions of the file systems containing a target and a prerequisite are different, the timestamp with the more precise resolution should be rounded down to the resolution of the less precise timestamp for the comparison.

## FUTURE DIRECTIONS

Some implementations of *make* include an *export* directive to add specified *make* variables to the environment. This may be considered for standardization in a future version.

A future version of this standard may require that macro expansions using the forms  $\$(string1:[op]\%[os]=[np][\%][ns])$  or  $\${string1:[op]\%[os]=[np][\%][ns]}$  are treated as pattern macro expansions.

98993 **SEE ALSO**98994 [Chapter 2](#) (on page 2335), [ar](#), [c99](#), [get](#), [lex](#), [sccs](#), [sh](#), [yacc](#)98995 [XBD Section 6.1](#) (on page 125), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)98996 [XSH \*exec\*, \*system\*\(\)](#)98997 **CHANGE HISTORY**

98998 First released in Issue 2.

98999 **Issue 5**

99000 The FUTURE DIRECTIONS section is added.

99001 **Issue 6**

99002 This utility is marked as part of the Software Development Utilities option.

99003 The Open Group Corrigendum U029/1 is applied, correcting a typographical error in the  
99004 SPECIAL TARGETS section.99005 In the ENVIRONMENT VARIABLES section, the *PROJECTDIR* description is updated from  
99006 “otherwise, the home directory of a user of that name is examined” to “otherwise, the value of  
99007 *PROJECTDIR* is treated as a user name and that user’s initial working directory is examined”.99008 It is specified whether the command line is related to the makefile or to the *make* command, and  
99009 the macro processing rules are updated to align with the IEEE P1003.2b draft standard.

99010 The normative text is reworded to avoid use of the term “must” for application requirements.

99011 PASC Interpretation 1003.2 #193 is applied.

99012 **Issue 7**99013 SD5-XCU-ERN-6 is applied, clarifying that Guideline 9 of the Utility Syntax Guidelines does not  
99014 apply.

99015 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

99016 Include lines in makefiles are introduced.

99017 Austin Group Interpretation 1003.1-2001 #131 is applied, changing the **Makefile Execution**  
99018 section.

99019 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0121 [257] is applied.



99020 **NAME**

99021       man — display system documentation

99022 **SYNOPSIS**99023       man [-k] *name* . . .99024 **DESCRIPTION**

99025       The *man* utility shall write information about each of the *name* operands. If *name* is the name of a  
 99026       standard utility, *man* at a minimum shall write a message describing the syntax used by the  
 99027       standard utility, its options, and operands. If more information is available, the *man* utility shall  
 99028       provide it in an implementation-defined manner.

99029       An implementation may provide information for values of *name* other than the standard utilities.  
 99030       Standard utilities that are listed as optional and that are not supported by the implementation  
 99031       either shall cause a brief message indicating that fact to be displayed or shall cause a full display  
 99032       of information as described previously.

99033 **OPTIONS**99034       The *man* utility shall conform to XBD [Section 12.2](#) (on page 216).

99035       The following option shall be supported:

99036       **-k**       Interpret *name* operands as keywords to be used in searching a utilities summary  
 99037       database that contains a brief purpose entry for each standard utility and write lines  
 99038       from the summary database that match any of the keywords. The keyword search shall  
 99039       produce results that are the equivalent of the output of the following command:

```
99040       grep -Ei '
99041        name
99042        name
99043        . . .
99044        ' summary-database
```

99045       This assumes that the *summary-database* is a text file with a single entry per line; this  
 99046       organization is not required and the example using *grep -Ei* is merely illustrative of the  
 99047       type of search intended. The purpose entry to be included in the database shall consist  
 99048       of a terse description of the purpose of the utility.

99049 **OPERANDS**

99050       The following operand shall be supported:

99051       *name*       A keyword or the name of a standard utility. When **-k** is not specified and *name*  
 99052       does not represent one of the standard utilities, the results are unspecified.

99053 **STDIN**

99054       Not used.

99055 **INPUT FILES**

99056       None.

99057 **ENVIRONMENT VARIABLES**99058       The following environment variables shall affect the execution of *man*:

99059       **LANG**       Provide a default value for the internationalization variables that are unset or null.  
 99060       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 99061       variables used to determine the values of locale categories.)

99062       **LC\_ALL**      If set to a non-empty string value, override the values of all the other  
 99063       internationalization variables.

99064 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 99065 characters (for example, single-byte as opposed to multi-byte characters in  
 99066 arguments and in the summary database). The value of *LC\_CTYPE* need not affect  
 99067 the format of the information written about the *name* operands.

99068 **LC\_MESSAGES**  
 99069 Determine the locale that should be used to affect the format and contents of  
 99070 diagnostic messages written to standard error and informative messages written to  
 99071 standard output.

99072 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

99073 **PAGER** Determine an output filtering command for writing the output to a terminal. Any  
 99074 string acceptable as a *command\_string* operand to the *sh -c* command shall be valid.  
 99075 When standard output is a terminal device, the reference page output shall be  
 99076 piped through the command. If the *PAGER* variable is null or not set, the command  
 99077 shall be either *more* or another paginator utility documented in the system  
 99078 documentation.

## 99079 **ASYNCHRONOUS EVENTS**

99080 Default.

## 99081 **STDOUT**

99082 The *man* utility shall write text describing the syntax of the utility *name*, its options and its  
 99083 operands, or, when *-k* is specified, lines from the summary database. The format of this text is  
 99084 implementation-defined.

## 99085 **STDERR**

99086 The standard error shall be used for diagnostic messages, and may also be used for  
 99087 informational messages of unspecified format.

## 99088 **OUTPUT FILES**

99089 None.

## 99090 **EXTENDED DESCRIPTION**

99091 None.

## 99092 **EXIT STATUS**

99093 The following exit values shall be returned:

99094 0 Successful completion.

99095 >0 An error occurred.

## 99096 **CONSEQUENCES OF ERRORS**

99097 Default.

## 99098 **APPLICATION USAGE**

99099 None.

## 99100 **EXAMPLES**

99101 None.

## 99102 **RATIONALE**

99103 It is recognized that the *man* utility is only of minimal usefulness as specified. The opinion of the  
 99104 standard developers was strongly divided as to how much or how little information *man* should  
 99105 be required to provide. They considered, however, that the provision of some portable way of  
 99106 accessing documentation would aid user portability. The arguments against a fuller specification  
 99107 were:

- Large quantities of documentation should not be required on a system that does not have excess disk space.
- The current manual system does not present information in a manner that greatly aids user portability.
- A “better help system” is currently an area in which vendors feel that they can add value to their POSIX implementations.

The `-f` option was considered, but due to implementation differences, it was not included in this volume of POSIX.1-2008.

The description was changed to be more specific about what has to be displayed for a utility. The standard developers considered it insufficient to allow a display of only the synopsis without giving a short description of what each option and operand does.

The “purpose” entry to be included in the database can be similar to the section title (less the numeric prefix) from this volume of POSIX.1-2008 for each utility. These titles are similar to those used in historical systems for this purpose.

See *mailx* for rationale concerning the default paginator.

The caveat in the *LC\_CTYPE* description was added because it is not a requirement that an implementation provide reference pages for all of its supported locales on each system; changing *LC\_CTYPE* does not necessarily translate the reference page into another language. This is equivalent to the current state of *LC\_MESSAGES* in POSIX.1-2008—locale-specific messages are not yet a requirement.

The historical *MANPATH* variable is not included in POSIX because no attempt is made to specify naming conventions for reference page files, nor even to mandate that they are files at all. On some implementations they could be a true database, a hypertext file, or even fixed strings within the *man* executable. The standard developers considered the portability of reference pages to be outside their scope of work. However, users should be aware that *MANPATH* is implemented on a number of historical systems and that it can be used to tailor the search pattern for reference pages from the various categories (utilities, functions, file formats, and so on) when the system administrator reveals the location and conventions for reference pages on the system.

The keyword search can rely on at least the text of the section titles from these utility descriptions, and the implementation may add more keywords. The term “section titles” refers to the strings such as:

```
man - Display system documentation
ps - Report process status
```

## FUTURE DIRECTIONS

None.

## SEE ALSO

[more](#)

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

## CHANGE HISTORY

First released in Issue 4.

99149 **Issue 5**

99150 The FUTURE DIRECTIONS section is added.

99151 **Issue 7**99152 Austin Group Interpretation 1003.1-2001 #108 is applied, clarifying that informational messages  
99153 may appear on standard error.

99154 **NAME**

99155       mesg — permit or deny messages

99156 **SYNOPSIS**99157       mesg [*y*|*n*]99158 **DESCRIPTION**

99159       The *mesg* utility shall control whether other users are allowed to send messages via *write*, *talk*, or  
 99160       other utilities to a terminal device. The terminal device affected shall be determined by searching  
 99161       for the first terminal in the sequence of devices associated with standard input, standard output,  
 99162       and standard error, respectively. With no arguments, *mesg* shall report the current state without  
 99163       changing it. Processes with appropriate privileges may be able to send messages to the terminal  
 99164       independent of the current state.

99165 **OPTIONS**

99166       None.

99167 **OPERANDS**

99168       The following operands shall be supported in the POSIX locale:

99169       *y*               Grant permission to other users to send messages to the terminal device.99170       *n*               Deny permission to other users to send messages to the terminal device.99171 **STDIN**

99172       Not used.

99173 **INPUT FILES**

99174       None.

99175 **ENVIRONMENT VARIABLES**99176       The following environment variables shall affect the execution of *mesg*:

99177       *LANG*           Provide a default value for the internationalization variables that are unset or null.  
 99178       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 99179       variables used to determine the values of locale categories.)

99180       *LC\_ALL*          If set to a non-empty string value, override the values of all the other  
 99181       internationalization variables.

99182       *LC\_CTYPE*       Determine the locale for the interpretation of sequences of bytes of text data as  
 99183       characters (for example, single-byte as opposed to multi-byte characters in  
 99184       arguments).

99185       *LC\_MESSAGES*

99186                       Determine the locale that should be used to affect the format and contents of  
 99187       diagnostic messages written (by *mesg*) to standard error.

99188       XSI       *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.99189 **ASYNCHRONOUS EVENTS**

99190       Default.

99191 **STDOUT**99192       If no operand is specified, *mesg* shall display the current terminal state in an unspecified format.99193 **STDERR**

99194       The standard error shall be used only for diagnostic messages.

**OUTPUT FILES**

None.

**EXTENDED DESCRIPTION**

None.

**EXIT STATUS**

The following exit values shall be returned:

0 Receiving messages is allowed.

1 Receiving messages is not allowed.

&gt;1 An error occurred.

**CONSEQUENCES OF ERRORS**

Default.

**APPLICATION USAGE**

The mechanism by which the message status of the terminal is changed is unspecified. Therefore, unspecified actions may cause the status of the terminal to change after *mesg* has successfully completed. These actions may include, but are not limited to: another invocation of the *mesg* utility, login procedures; invocation of the *stty* utility, invocation of the *chmod* utility or *chmod()* function, and so on.

**EXAMPLES**

None.

**RATIONALE**

The terminal changed by *mesg* is that associated with the standard input, output, or error, rather than the controlling terminal for the session. This is because users logged in more than once should be able to change any of their login terminals without having to stop the job running in those sessions. This is not a security problem involving the terminals of other users because appropriate privileges would be required to affect the terminal of another user.

The method of checking each of the first three file descriptors in sequence until a terminal is found was adopted from System V.

The file */dev/tty* is not specified for the terminal device because it was thought to be too restrictive. Typical environment changes for the *n* operand are that write permissions are removed for *others* and *group* from the appropriate device. It was decided to leave the actual description of what is done as unspecified because of potential differences between implementations.

The format for standard output is unspecified because of differences between historical implementations. This output is generally not useful to shell scripts (they can use the exit status), so exact parsing of the output is unnecessary.

**FUTURE DIRECTIONS**

None.

**SEE ALSO***talk*, *write*

XBD Chapter 8 (on page 173)

99235 **CHANGE HISTORY**

99236 First released in Issue 2.

99237 **Issue 6**

99238 This utility is marked as part of the User Portability Utilities option.

99239 **Issue 7**99240 The *mesg* utility is moved from the User Portability Utilities option to the Base. User Portability  
99241 Utilities is now an option for interactive utilities.

**NAME**

mkdir — make directories

**SYNOPSIS**

mkdir [-p] [-m *mode*] *dir*...

**DESCRIPTION**

The *mkdir* utility shall create the directories specified by the operands, in the order specified.

For each *dir* operand, the *mkdir* utility shall perform actions equivalent to the *mkdir()* function defined in the System Interfaces volume of POSIX.1-2008, called with the following arguments:

1. The *dir* operand is used as the *path* argument.
2. The value of the bitwise-inclusive OR of S\_IRWXU, S\_IRWXG, and S\_IRWXO is used as the *mode* argument. (If the **-m** option is specified, the value of the *mkdir()* *mode* argument is unspecified, but the directory shall at no time have permissions less restrictive than the **-m mode** option-argument.)

**OPTIONS**

The *mkdir* utility shall conform to XBD [Section 12.2](#) (on page 216).

The following options shall be supported:

**-m mode** Set the file permission bits of the newly-created directory to the specified *mode* value. The *mode* option-argument shall be the same as the *mode* operand defined for the *chmod* utility. In the *symbolic\_mode* strings, the *op* characters '+' and '-' shall be interpreted relative to an assumed initial mode of *a=rwx*; '+' shall add permissions to the default mode, '-' shall delete permissions from the default mode.

**-p** Create any missing intermediate pathname components.

For each *dir* operand that does not name an existing directory, before performing the actions described in the DESCRIPTION above, the *mkdir* utility shall create any pathname components of the path prefix of *dir* that do not name an existing directory by performing actions equivalent to first calling the *mkdir()* function with the following arguments:

1. A pathname naming the missing pathname component, ending with a trailing <slash> character, as the *path* argument
2. The value zero as the *mode* argument

and then calling the *chmod()* function with the following arguments:

1. The same *path* argument as in the *mkdir()* call
2. The value (S\_IWUSR|S\_IXUSR|~*filemask*)&0777 as the *mode* argument, where *filemask* is the file mode creation mask of the process (see XSH [umask\(\)](#))

Each *dir* operand that names an existing directory shall be ignored without error.

**OPERANDS**

The following operand shall be supported:

*dir* A pathname of a directory to be created.



99282 **STDIN**

99283 Not used.

99284 **INPUT FILES**

99285 None.

99286 **ENVIRONMENT VARIABLES**99287 The following environment variables shall affect the execution of *mkdir*:

99288 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 99289 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 99290 variables used to determine the values of locale categories.)

99291 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 99292 internationalization variables.

99293 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 99294 characters (for example, single-byte as opposed to multi-byte characters in  
 99295 arguments).

99296 *LC\_MESSAGES*

99297 Determine the locale that should be used to affect the format and contents of  
 99298 diagnostic messages written to standard error.

99299 *XSI* *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

99300 **ASYNCHRONOUS EVENTS**

99301 Default.

99302 **STDOUT**

99303 Not used.

99304 **STDERR**

99305 The standard error shall be used only for diagnostic messages.

99306 **OUTPUT FILES**

99307 None.

99308 **EXTENDED DESCRIPTION**

99309 None.

99310 **EXIT STATUS**

99311 The following exit values shall be returned:

99312 0 All the specified directories were created successfully, or the *-p* option was specified and all  
 99313 the specified directories either already existed or were created successfully.

99314 &gt;0 An error occurred.

99315 **CONSEQUENCES OF ERRORS**

99316 Default.

**APPLICATION USAGE**

The default file mode for directories is *a=rwx* (777 on most systems) with selected permissions removed in accordance with the file mode creation mask. For intermediate pathname components created by *mkdir*, the mode is the default modified by *u+wx* so that the subdirectories can always be created regardless of the file mode creation mask; if different ultimate permissions are desired for the intermediate directories, they can be changed afterwards with *chmod*.

Note that some of the requested directories may have been created even if an error occurs.

**EXAMPLES**

None.

**RATIONALE**

The System V *-m* option was included to control the file mode.

The System V *-p* option was included to create any needed intermediate directories and to complement the functionality provided by *rmdir* for removing directories in the path prefix as they become empty. Because no error is produced if any path component already exists, the *-p* option is also useful to ensure that a particular directory exists.

The functionality of *mkdir* is described substantially through a reference to the *mkdir()* function in the System Interfaces volume of POSIX.1-2008. For example, by default, the mode of the directory is affected by the file mode creation mask in accordance with the specified behavior of the *mkdir()* function. In this way, there is less duplication of effort required for describing details of the directory creation.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*chmod*, *rm*, *rmdir*, *umask*

XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

XSH *mkdir()*, *umask()*

**CHANGE HISTORY**

First released in Issue 2.

**Issue 5**

The FUTURE DIRECTIONS section is added.

**Issue 7**

SD5-XCU-ERN-56 is applied, aligning the *-m* option with the IEEE P1003.2b draft standard to clarify an ambiguity.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0122 [161] is applied.

**NAME**

mkfifo — make FIFO special files

**SYNOPSIS**

mkfifo [-m *mode*] *file*...

**DESCRIPTION**

The *mkfifo* utility shall create the FIFO special files specified by the operands, in the order specified.

For each *file* operand, the *mkfifo* utility shall perform actions equivalent to the *mkfifo*() function defined in the System Interfaces volume of POSIX.1-2008, called with the following arguments:

1. The *file* operand is used as the *path* argument.
2. The value of the bitwise-inclusive OR of S\_IRUSR, S\_IWUSR, S\_IRGRP, S\_IWGRP, S\_IROTH, and S\_IWOTH is used as the *mode* argument. (If the -m option is specified, the value of the *mkfifo*() *mode* argument is unspecified, but the FIFO shall at no time have permissions less restrictive than the -m *mode* option-argument.)

**OPTIONS**

The *mkfifo* utility shall conform to XBD [Section 12.2](#) (on page 216).

The following option shall be supported:

-m *mode* Set the file permission bits of the newly-created FIFO to the specified *mode* value. The *mode* option-argument shall be the same as the *mode* operand defined for the *chmod* utility. In the *symbolic\_mode* strings, the *op* characters '+' and '-' shall be interpreted relative to an assumed initial mode of *a=rw*.

**OPERANDS**

The following operand shall be supported:

*file* A pathname of the FIFO special file to be created.

**STDIN**

Not used.

**INPUT FILES**

None.

**ENVIRONMENT VARIABLES**

The following environment variables shall affect the execution of *mkfifo*:

LANG Provide a default value for the internationalization variables that are unset or null. (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization variables used to determine the values of locale categories.)

LC\_ALL If set to a non-empty string value, override the values of all the other internationalization variables.

LC\_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).

LC\_MESSAGES

Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.

99394 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

99395 **ASYNCHRONOUS EVENTS**

99396 Default.

99397 **STDOUT**

99398 Not used.

99399 **STDERR**

99400 The standard error shall be used only for diagnostic messages.

99401 **OUTPUT FILES**

99402 None.

99403 **EXTENDED DESCRIPTION**

99404 None.

99405 **EXIT STATUS**

99406 The following exit values shall be returned:

99407 0 All the specified FIFO special files were created successfully.

99408 >0 An error occurred.

99409 **CONSEQUENCES OF ERRORS**

99410 Default.

99411 **APPLICATION USAGE**

99412 None.

99413 **EXAMPLES**

99414 None.

99415 **RATIONALE**

99416 This utility was added to permit shell applications to create FIFO special files.

99417 The **-m** option was added to control the file mode, for consistency with the similar functionality

99418 provided by the *mkdir* utility.

99419 Early proposals included a **-p** option similar to the *mkdir -p* option that created intermediate

99420 directories leading up to the FIFO specified by the final component. This was removed because

99421 it is not commonly needed and is not common practice with similar utilities.

99422 The functionality of *mkfifo* is described substantially through a reference to the *mkfifo()* function

99423 in the System Interfaces volume of POSIX.1-2008. For example, by default, the mode of the FIFO

99424 file is affected by the file mode creation mask in accordance with the specified behavior of the

99425 *mkfifo()* function. In this way, there is less duplication of effort required for describing details of

99426 the file creation.

99427 **FUTURE DIRECTIONS**

99428 None.

99429 **SEE ALSO**

99430 *chmod*, *umask*

99431 XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

99432 XSH *mkfifo()*

99433 **CHANGE HISTORY**

99434 First released in Issue 3.

99435 **Issue 6**99436 The **-m** option is aligned with the IEEE P1003.2b draft standard to clarify an ambiguity.

99437 **NAME**99438 `more` — display files on a page-by-page basis99439 **SYNOPSIS**99440 UP `more [-ceisu] [-n number] [-p command] [-t tagstring] [file...]`99441 **DESCRIPTION**

99442 The *more* utility shall read files and either write them to the terminal on a page-by-page basis or  
 99443 filter them to standard output. If standard output is not a terminal device, all input files shall be  
 99444 copied to standard output in their entirety, without modification, except as specified for the `-s`  
 99445 option. If standard output is a terminal device, the files shall be written a number of lines (one  
 99446 screenful) at a time under the control of user commands. See the EXTENDED DESCRIPTION  
 99447 section.

99448 Certain block-mode terminals do not have all the capabilities necessary to support the complete  
 99449 *more* definition; they are incapable of accepting commands that are not terminated with a  
 99450 <newline>. Implementations that support such terminals shall provide an operating mode to  
 99451 *more* in which all commands can be terminated with a <newline> on those terminals. This mode:

- 99452 • Shall be documented in the system documentation
- 99453 • Shall, at invocation, inform the user of the terminal deficiency that requires the <newline>  
 99454 usage and provide instructions on how this warning can be suppressed in future  
 99455 invocations
- 99456 • Shall not be required for implementations supporting only fully capable terminals
- 99457 • Shall not affect commands already requiring <newline> characters
- 99458 • Shall not affect users on the capable terminals from using *more* as described in this volume  
 99459 of POSIX.1-2008

99460 **OPTIONS**

99461 The *more* utility shall conform to XBD [Section 12.2](#) (on page 216), except that '+' may be  
 99462 recognized as an option delimiter as well as '-'.

99463 The following options shall be supported:

- 99464 **-c** If a screen is to be written that has no lines in common with the current screen, or  
 99465 *more* is writing its first screen, *more* shall not scroll the screen, but instead shall  
 99466 redraw each line of the screen in turn, from the top of the screen to the bottom. In  
 99467 addition, if *more* is writing its first screen, the screen shall be cleared. This option  
 99468 may be silently ignored on devices with insufficient terminal capabilities.
- 99469 **-e** Exit immediately after writing the last line of the last file in the argument list; see  
 99470 the EXTENDED DESCRIPTION section.
- 99471 **-i** Perform pattern matching in searches without regard to case; see XBD [Section 9.2](#)  
 99472 (on page 182).
- 99473 **-n *number*** Specify the number of lines per screenful. The *number* argument is a positive  
 99474 decimal integer. The `-n` option shall override any values obtained from any other  
 99475 source.
- 99476 **-p *command*** Each time a screen from a new file is displayed or redisplayed (including as a  
 99477 result of *more* commands; for example, `:p`), execute the *more* command(s) in the  
 99478 command arguments in the order specified, as if entered by the user after the first  
 99479 screen has been displayed. No intermediate results shall be displayed (that is, if the  
 99480 command is a movement to a screen different from the normal first screen, only the

- 99481 screen resulting from the command shall be displayed.) If any of the commands  
 99482 fail for any reason, an informational message to this effect shall be written, and no  
 99483 further commands specified using the **-p** option shall be executed for this file.
- 99484 **-s** Behave as if consecutive empty lines were a single empty line.
- 99485 **-t tagstring** Write the screenful of the file containing the tag named by the *tagstring* argument.  
 99486 See the *ctags* utility. The tags feature represented by **-t tagstring** and the **:t**  
 99487 command is optional. It shall be provided on any system that also provides a  
 99488 conforming implementation of *ctags*; otherwise, the use of **-t** produces undefined  
 99489 results.
- 99490 The filename resulting from the **-t** option shall be logically added as a prefix to the  
 99491 list of command line files, as if specified by the user. If the tag named by the  
 99492 *tagstring* argument is not found, it shall be an error, and *more* shall take no further  
 99493 action.
- 99494 If the tag specifies a line number, the first line of the display shall contain the  
 99495 beginning of that line. If the tag specifies a pattern, the first line of the display shall  
 99496 contain the beginning of the matching text from the first line of the file that  
 99497 contains that pattern. If the line does not exist in the file or matching text is not  
 99498 found, an informational message to this effect shall be displayed, and *more* shall  
 99499 display the default screen as if **-t** had not been specified.
- 99500 If both the **-t tagstring** and **-p command** options are given, the **-t tagstring** shall be  
 99501 processed first; that is, the file and starting line for the display shall be as specified  
 99502 by **-t**, and then the **-p more** command shall be executed. If the line (matching text)  
 99503 specified by the **-t** command does not exist (is not found), no **-p more** command  
 99504 shall be executed for this file at any time.
- 99505 **-u** Treat a <backspace> as a printable control character, displayed as an  
 99506 implementation-defined character sequence (see the EXTENDED DESCRIPTION  
 99507 section), suppressing backspacing and the special handling that produces  
 99508 underlined or standout mode text on some terminal types. Also, do not ignore a  
 99509 <carriage-return> at the end of a line.

## 99510 OPERANDS

- 99511 The following operand shall be supported:

- 99512 *file* A pathname of an input file. If no *file* operands are specified, the standard input  
 99513 shall be used. If a *file* is '-', the standard input shall be read at that point in the  
 99514 sequence.

## 99515 STDIN

- 99516 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '-'.

## 99517 INPUT FILES

- 99518 The input files being examined shall be text files. If standard output is a terminal, standard error  
 99519 shall be used to read commands from the user. If standard output is a terminal, standard error is  
 99520 not readable, and command input is needed, *more* may attempt to obtain user commands from  
 99521 the controlling terminal (for example, */dev/tty*); otherwise, *more* shall terminate with an error  
 99522 indicating that it was unable to read user commands. If standard output is not a terminal, no  
 99523 error shall result if standard error cannot be opened for reading.

99524 **ENVIRONMENT VARIABLES**99525 The following environment variables shall affect the execution of *more*:

99526 *COLUMNS* Override the system-selected horizontal display line size. See XBD [Chapter 8](#) (on  
99527 page 173) for valid values and results when it is unset or null.

99528 *EDITOR* Used by the *v* command to select an editor. See the EXTENDED DESCRIPTION  
99529 section.

99530 *LANG* Provide a default value for the internationalization variables that are unset or null.  
99531 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
99532 variables used to determine the values of locale categories.)

99533 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
99534 internationalization variables.

99535 *LC\_COLLATE*  
99536 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
99537 character collating elements within regular expressions.

99538 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
99539 characters (for example, single-byte as opposed to multi-byte characters in  
99540 arguments and input files) and the behavior of character classes within regular  
99541 expressions.

99542 *LC\_MESSAGES*  
99543 Determine the locale that should be used to affect the format and contents of  
99544 diagnostic messages written to standard error and informative messages written to  
99545 standard output.

99546 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

99547 *LINES* Override the system-selected vertical screen size, used as the number of lines in a  
99548 screenful. See XBD [Chapter 8](#) (on page 173) for valid values and results when it is  
99549 unset or null. The *-n* option shall take precedence over the *LINES* variable for  
99550 determining the number of lines in a screenful.

99551 *MORE* Determine a string containing options described in the OPTIONS section preceded  
99552 with <hyphen-minus> characters and <blank>-separated as on the command line.  
99553 Any command line options shall be processed after those in the *MORE* variable, as  
99554 if the command line were:

99555 *more \$MORE options operands*

99556 The *MORE* variable shall take precedence over the *TERM* and *LINES* variables for  
99557 determining the number of lines in a screenful.

99558 *TERM* Determine the name of the terminal type. If this variable is unset or null, an  
99559 unspecified default terminal type is used.

99560 **ASYNCHRONOUS EVENTS**

99561 Default.

99562 **STDOUT**

99563 The standard output shall be used to write the contents of the input files.



**STDERR**

The standard error shall be used for diagnostic messages and user commands (see the INPUT FILES section), and, if standard output is a terminal device, to write a prompting string. The prompting string shall appear on the screen line below the last line of the file displayed in the current screenful. The prompt shall contain the name of the file currently being examined and shall contain an end-of-file indication and the name of the next file, if any, when prompting at the end-of-file. If an error or informational message is displayed, it is unspecified whether it is contained in the prompt. If it is not contained in the prompt, it shall be displayed and then the user shall be prompted for a continuation character, at which point another message or the user prompt may be displayed. The prompt is otherwise unspecified. It is unspecified whether informational messages are written for other user commands.

**OUTPUT FILES**

None.

**EXTENDED DESCRIPTION**

The following section describes the behavior of *more* when the standard output is a terminal device. If the standard output is not a terminal device, no options other than *-s* shall have any effect, and all input files shall be copied to standard output otherwise unmodified, at which time *more* shall exit without further action.

The number of lines available per screen shall be determined by the *-n* option, if present, or by examining values in the environment (see the ENVIRONMENT VARIABLES section). If neither method yields a number, an unspecified number of lines shall be used.

The maximum number of lines written shall be one less than this number, because the screen line after the last line written shall be used to write a user prompt and user input. If the number of lines in the screen is less than two, the results are undefined. It is unspecified whether user input is permitted to be longer than the remainder of the single line where the prompt has been written.

The number of columns available per line shall be determined by examining values in the environment (see the ENVIRONMENT VARIABLES section), with a default value as described in XBD Chapter 8 (on page 173).

Lines that are longer than the display shall be folded; the length at which folding occurs is unspecified, but should be appropriate for the output device. Folding may occur between glyphs of single characters that take up multiple display columns.

When standard output is a terminal and *-u* is not specified, *more* shall treat <backspace> and <carriage-return> characters specially:

- A character, followed first by a sequence of *n* <backspace> characters (where *n* is the same as the number of column positions that the character occupies), then by *n* <underscore> characters ( ' \_ ' ), shall cause that character to be written as underlined text, if the terminal type supports that. The *n* <underscore> characters, followed first by *n* <backspace> characters, then any character with *n* column positions, shall also cause that character to be written as underlined text, if the terminal type supports that.
- A sequence of *n* <backspace> characters (where *n* is the same as the number of column positions that the previous character occupies) that appears between two identical printable characters shall cause the first of those two characters to be written as emboldened text (that is, visually brighter, standout mode, or inverse-video mode), if the terminal type supports that, and the second to be discarded. Immediately subsequent occurrences of <backspace>/character pairs for that same character shall also be discarded. (For example, the sequence "a\ba\ba\ba" is interpreted as a single

emboldened 'a'.)

- The *more* utility shall logically discard all other <backspace> characters from the line as well as the character which precedes them, if any.
- A <carriage-return> at the end of a line shall be ignored, rather than being written as a non-printable character, as described in the next paragraph.

It is implementation-defined how other non-printable characters are written. Implementations should use the same format that they use for the *ex* **print** command; see the OPTIONS section within the *ed* utility. It is unspecified whether a multi-column character shall be separated if it crosses a display line boundary; it shall not be discarded. The behavior is unspecified if the number of columns on the display is less than the number of columns any single character in the line being displayed would occupy.

When each new file is displayed (or redisplayed), *more* shall write the first screen of the file. Once the initial screen has been written, *more* shall prompt for a user command. If the execution of the user command results in a screen that has lines in common with the current screen, and the device has sufficient terminal capabilities, *more* shall scroll the screen; otherwise, it is unspecified whether the screen is scrolled or redrawn.

For all files but the last (including standard input if no file was specified, and for the last file as well, if the **-e** option was not specified), when *more* has written the last line in the file, *more* shall prompt for a user command. This prompt shall contain the name of the next file as well as an indication that *more* has reached end-of-file. If the user command is **f**, <control>-F, <space>, **j**, <newline>, **d**, <control>-D, or **s**, *more* shall display the next file. Otherwise, if displaying the last file, *more* shall exit. Otherwise, *more* shall execute the user command specified.

Several of the commands described in this section display a previous screen from the input stream. In the case that text is being taken from a non-rewindable stream, such as a pipe, it is implementation-defined how much backwards motion is supported. If a command cannot be executed because of a limitation on backwards motion, an error message to this effect shall be displayed, the current screen shall not change, and the user shall be prompted for another command.

If a command cannot be performed because there are insufficient lines to display, *more* shall alert the terminal. If a command cannot be performed because there are insufficient lines to display or a **/** command fails: if the input is the standard input, the last screen in the file may be displayed; otherwise, the current file and screen shall not change, and the user shall be prompted for another command.

The interactive commands in the following sections shall be supported. Some commands can be preceded by a decimal integer, called *count* in the following descriptions. If not specified with the command, *count* shall default to 1. In the following descriptions, *pattern* is a basic regular expression, as described in XBD [Section 9.3](#) (on page 183). The term “examine” is historical usage meaning “open the file for viewing”; for example, *more* **foo** would be expressed as examining file **foo**.

In the following descriptions, unless otherwise specified, *line* is a line in the *more* display, not a line from the file being examined.

In the following descriptions, the *current position* refers to two things:

1. The position of the current line on the screen
2. The line number (in the file) of the current line on the screen

Usually, the line on the screen corresponding to the current position is the third line on the

screen. If this is not possible (there are fewer than three lines to display or this is the first page of the file, or it is the last page of the file), then the current position is either the first or last line on the screen as described later.

## Help

*Synopsis:*     **h**

Write a summary of these commands and other implementation-defined commands. The behavior shall be as if the *more* utility were executed with the **-e** option on a file that contained the summary information. The user shall be prompted as described earlier in this section when end-of-file is reached. If the user command is one of those specified to continue to the next file, *more* shall return to the file and screen state from which the **h** command was executed.

## Scroll Forward One Screenful

*Synopsis:*     **[count]f**  
                 **[count]<control>-F**

Scroll forward *count* lines, with a default of one screenful. If *count* is more than the screen size, only the final screenful shall be written.

## Scroll Backward One Screenful

*Synopsis:*     **[count]b**  
                 **[count]<control>-B**

Scroll backward *count* lines, with a default of one screenful (see the **-n** option). If *count* is more than the screen size, only the final screenful shall be written.

## Scroll Forward One Line

*Synopsis:*     **[count]<space>**  
                 **[count]j**  
                 **[count]<newline>**

Scroll forward *count* lines. The default *count* for the **<space>** shall be one screenful; for **j** and **<newline>**, one line. The entire *count* lines shall be written, even if *count* is more than the screen size.

## Scroll Backward One Line

*Synopsis:*     **[count]k**

Scroll backward *count* lines. The entire *count* lines shall be written, even if *count* is more than the screen size.

## Scroll Forward One Half Screenful

*Synopsis:*     **[count]d**  
                 **[count]<control>-D**

Scroll forward *count* lines, with a default of one half of the screen size. If *count* is specified, it shall become the new default for subsequent **d**, **<control>-D**, and **u** commands.

**Skip Forward One Line**

*Synopsis:*     [*count*]**s**

Display the screenful beginning with the line *count* lines after the last line on the current screen. If *count* would cause the current position to be such that less than one screenful would be written, the last screenful in the file shall be written.

**Scroll Backward One Half Screenful**

*Synopsis:*     [*count*]**u**  
                  [*count*]**<control>-U**

Scroll backward *count* lines, with a default of one half of the screen size. If *count* is specified, it shall become the new default for subsequent **d**, **<control>-D**, **u**, and **<control>-U** commands. The entire *count* lines shall be written, even if *count* is more than the screen size.

**Go to Beginning of File**

*Synopsis:*     [*count*]**g**

Display the screenful beginning with line *count*.

**Go to End-of-File**

*Synopsis:*     [*count*]**G**

If *count* is specified, display the screenful beginning with the line *count*. Otherwise, display the last screenful of the file.

**Refresh the Screen**

*Synopsis:*     **r**  
                  **<control>-L**

Refresh the screen.

**Discard and Refresh**

*Synopsis:*     **R**

Refresh the screen, discarding any buffered input. If the current file is non-seekable, buffered input shall not be discarded and the **R** command shall be equivalent to the **r** command.

**Mark Position**

*Synopsis:*     **m***letter*

Mark the current position with the letter named by *letter*, where *letter* represents the name of one of the lowercase letters of the portable character set. When a new file is examined, all marks may be lost.

**Return to Mark**

*Synopsis:*     ' *letter*

Return to the position that was previously marked with the letter named by *letter*, making that line the current position.

**Return to Previous Position**

*Synopsis:*     ' '

Return to the position from which the last large movement command was executed (where a "large movement" is defined as any movement of more than a screenful of lines). If no such movements have been made, return to the beginning of the file.

**Search Forward for Pattern**

*Synopsis:*     [*count*]/[!]*pattern*<newline>

Display the screenful beginning with the *count*th line containing the pattern. The search shall start after the first line currently displayed. The null regular expression ('/' followed by a <newline>) shall repeat the search using the previous regular expression, with a default *count*. If the character '!' is included, the matching lines shall be those that do not contain the *pattern*. If no match is found for the *pattern*, a message to that effect shall be displayed.

**Search Backward for Pattern**

*Synopsis:*     [*count*?][!]*pattern*<newline>

Display the screenful beginning with the *count*th previous line containing the pattern. The search shall start on the last line before the first line currently displayed. The null regular expression ('?' followed by a <newline>) shall repeat the search using the previous regular expression, with a default *count*. If the character '!' is included, matching lines shall be those that do not contain the *pattern*. If no match is found for the *pattern*, a message to that effect shall be displayed.

**Repeat Search**

*Synopsis:*     [*count*]n

Repeat the previous search for *count*th line containing the last *pattern* (or not containing the last *pattern*, if the previous search was "/" or "?").

**Repeat Search in Reverse**

*Synopsis:*     [*count*]N

Repeat the search in the opposite direction of the previous search for the *count*th line containing the last *pattern* (or not containing the last *pattern*, if the previous search was "/" or "?").

**Examine New File**

*Synopsis:*        :e [*filename*]  
<newline>

Examine a new file. If the *filename* argument is not specified, the current file (see the :n and :p commands below) shall be re-examined. The *filename* shall be subjected to the process of shell word expansions (see [Section 2.6](#), on page 2343); if more than a single pathname results, the effects are unspecified. If *filename* is a <number-sign> ('#'), the previously examined file shall be re-examined. If *filename* is not accessible for any reason (including that it is a non-seekable file), an error message to this effect shall be displayed and the current file and screen shall not change.

**Examine Next File**

*Synopsis:*        [count]:n

Examine the next file. If a number *count* is specified, the *count*th next file shall be examined. If *filename* refers to a non-seekable file, the results are unspecified.

**Examine Previous File**

*Synopsis:*        [count]:p

Examine the previous file. If a number *count* is specified, the *count*th previous file shall be examined. If *filename* refers to a non-seekable file, the results are unspecified.

**Go to Tag**

*Synopsis:*        :t *tagstring*  
<newline>

If the file containing the tag named by the *tagstring* argument is not the current file, examine the file, as if the :e command was executed with that file as the argument. Otherwise, or in addition, display the screenful beginning with the tag, as described for the -t option (see the OPTIONS section). If the *ctags* utility is not supported by the system, the use of :t produces undefined results.

**Invoke Editor**

*Synopsis:*        v

Invoke an editor to edit the current file being examined. If standard input is being examined, the results are unspecified. The name of the editor shall be taken from the environment variable *EDITOR*, or shall default to *vi*. If the last pathname component in *EDITOR* is either *vi* or *ex*, the editor shall be invoked with a -c *linenumber* command line argument, where *linenumber* is the line number of the file line containing the display line currently displayed as the first line of the screen. It is implementation-defined whether line-setting options are passed to editors other than *vi* and *ex*.

When the editor exits, *more* shall resume with the same file and screen as when the editor was invoked.

**Display Position**

*Synopsis:*       =

                  <control>-G

Write a message for which the information references the first byte of the line after the last line of the file on the screen. This message shall include the name of the file currently being examined, its number relative to the total number of files there are to examine, the line number in the file, the byte number and the total bytes in the file, and what percentage of the file precedes the current position. If *more* is reading from standard input, or the file is shorter than a single screen, the line number, the byte number, the total bytes, and the percentage need not be written.

**Quit**

*Synopsis:*       q

                  :q

                  ZZ

Exit *more*.

**EXIT STATUS**

The following exit values shall be returned:

- 0   Successful completion.
- >0   An error occurred.

**CONSEQUENCES OF ERRORS**

If an error is encountered accessing a file when using the **:n** command, *more* shall attempt to examine the next file in the argument list, but the final exit status shall be affected. If an error is encountered accessing a file via the **:p** command, *more* shall attempt to examine the previous file in the argument list, but the final exit status shall be affected. If an error is encountered accessing a file via the **:e** command, *more* shall remain in the current file and the final exit status shall not be affected.

**APPLICATION USAGE**

When the standard output is not a terminal, only the **-s** filter-modification option is effective. This is based on historical practice. For example, a typical implementation of *man* pipes its output through *more -s* to squeeze excess white space for terminal users. When *man* is piped to *lp*, however, it is undesirable for this squeezing to happen.

**EXAMPLES**

The **-p** allows arbitrary commands to be executed at the start of each file. Examples are:

*more -p G file1 file2*  
Examine each file starting with its last screenful.

*more -p 100 file1 file2*  
Examine each file starting with line 100 in the current position (usually the third line, so line 98 would be the first line written).

*more -p /100 file1 file2*  
Examine each file starting with the first line containing the string "100" in the current position



## RATIONALE

The *more* utility, available in BSD and BSD-derived systems, was chosen as the prototype for the POSIX file display program since it is more widely available than either the public-domain program *less* or than *pg*, a pager provided in System V. The 4.4 BSD *more* is the model for the features selected; it is almost fully upwards-compatible from the 4.3 BSD version in wide use and has become more amenable for *vi* users. Several features originally derived from various file editors, found in both *less* and *pg*, have been added to this volume of POSIX.1-2008 as they have proved extremely popular with users.

There are inconsistencies between *more* and *vi* that result from historical practice. For example, the single-character commands **h**, **f**, **b**, and **<space>** are screen movers in *more*, but cursor movers in *vi*. These inconsistencies were maintained because the cursor movements are not applicable to *more* and the powerful functionality achieved without the use of the control key justifies the differences.

The tags interface has been included in a program that is not a text editor because it promotes another degree of consistent operation with *vi*. It is conceivable that the paging environment of *more* would be superior for browsing source code files in some circumstances.

The operating mode referred to for block-mode terminals effectively adds a **<newline>** to each Synopsis line that currently has none. So, for example, **d<newline>** would page one screenful. The mode could be triggered by a command line option, environment variable, or some other method. The details are not imposed by this volume of POSIX.1-2008 because there are so few systems known to support such terminals. Nevertheless, it was considered that all systems should be able to support *more* given the exception cited for this small community of terminals because, in comparison to *vi*, the cursor movements are few and the command set relatively amenable to the optional **<newline>** characters.

Some versions of *more* provide a shell escaping mechanism similar to the *ex* **!** command. The standard developers did not consider that this was necessary in a paginator, particularly given the wide acceptance of multiple window terminals and job control features. (They chose to retain such features in the editors and *mailx* because the shell interaction also gives an opportunity to modify the editing buffer, which is not applicable to *more*.)

The **-p** (position) option replaces the **+** command because of the Utility Syntax Guidelines. The **+command** option is no longer specified by POSIX.1-2008 but may be present in some implementations. In early proposals, it took a *pattern* argument, but historical *less* provided the *more* general facility of a command. It would have been desirable to use the same **-c** as *ex* and *vi*, but the letter was already in use.

The text stating “from a non-rewindable stream ... implementations may limit the amount of backwards motion supported” would allow an implementation that permitted no backwards motion beyond text already on the screen. It was not possible to require a minimum amount of backwards motion that would be effective for all conceivable device types. The implementation should allow the user to back up as far as possible, within device and reasonable memory allocation constraints.

Historically, non-printable characters were displayed using the ARPA standard mappings, which are as follows:

1. Printable characters are left alone.
2. Control characters less than **\177** are represented as followed by the character offset from the **'@'** character in the ASCII map; for example, **\007** is represented as **'G'**.



3. \177 is represented as followed by ' ? '.

The display of characters having their eighth bit set was less standard. Existing implementations use hex (0x00), octal (\000), and a meta-bit display. (The latter displayed characters with their eighth bit set as the two characters "M-", followed by the seven-bit display as described previously.) The latter probably has the best claim to historical practice because it was used with the -v option of 4 BSD and 4 BSD-derived versions of the *cat* utility since 1980.

No specific display format is required by POSIX.1-2008. Implementations are encouraged to conform to historic practice in the absence of any strong reason to diverge.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

Chapter 2 (on page 2335), *ctags*, *ed*, *ex*, *vi*

XBD Chapter 8 (on page 173), Section 9.2 (on page 182), Section 9.3 (on page 183), Section 12.2 (on page 216)

#### CHANGE HISTORY

First released in Issue 4.

##### Issue 5

The FUTURE DIRECTIONS section is added.

##### Issue 6

This utility is marked as part of the User Portability Utilities option.

The obsolescent SYNOPSIS is removed.

The utility has been extensively reworked for alignment with the IEEE P1003.2b draft standard:

- Changes have been made as a result of IEEE PASC Interpretations 1003.2 #37 and #109.
- The *more* utility should be able to handle underlined and emboldened displays of characters that are wider than a single column position.

##### Issue 7

Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying that '+' may be recognized as an option delimiter in the OPTIONS section.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0123 [265] is applied.

99905 **NAME**

99906 mv — move files

99907 **SYNOPSIS**99908 mv [-if] *source\_file target\_file*99909 mv [-if] *source\_file... target\_dir*99910 **DESCRIPTION**

99911 In the first synopsis form, the *mv* utility shall move the file named by the *source\_file* operand to the destination specified by the *target\_file*. This first synopsis form is assumed when the final operand does not name an existing directory and is not a symbolic link referring to an existing directory. In this case, if *source\_file* names a non-directory file and *target\_file* ends with a trailing <slash> character, *mv* shall treat this as an error and no *source\_file* operands will be processed.

99916 In the second synopsis form, *mv* shall move each file named by a *source\_file* operand to a destination file in the existing directory named by the *target\_dir* operand, or referenced if *target\_dir* is a symbolic link referring to an existing directory. The destination path for each *source\_file* shall be the concatenation of the target directory, a single <slash> character if the target did not end in a <slash>, and the last pathname component of the *source\_file*. This second form is assumed when the final operand names an existing directory.

99922 If any operand specifies an existing file of a type not specified by the System Interfaces volume of POSIX.1-2008, the behavior is implementation-defined.

99924 For each *source\_file* the following steps shall be taken:

- 99925 1. If the destination path exists, the *-f* option is not specified, and either of the following conditions is true:

- 99926 a. The permissions of the destination path do not permit writing and the standard input is a terminal.
- 99927 b. The *-i* option is specified.

99930 the *mv* utility shall write a prompt to standard error and read a line from standard input. If the response is not affirmative, *mv* shall do nothing more with the current *source\_file* and go on to any remaining *source\_files*.

- 99933 2. If the *source\_file* operand and destination path resolve to either the same existing directory entry or different directory entries for the same existing file, then the destination path shall not be removed, and one of the following shall occur:

- 99936 a. No change is made to *source\_file*, no error occurs, and no diagnostic is issued.
- 99937 b. No change is made to *source\_file*, a diagnostic is issued to standard error identifying the two names, and the exit status is affected.
- 99939 c. If the *source\_file* operand and destination path name distinct directory entries, then the *source\_file* operand is removed, no error occurs, and no diagnostic is issued.

99941 The *mv* utility shall do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.

- 99943 3. The *mv* utility shall perform actions equivalent to the *rename()* function defined in the System Interfaces volume of POSIX.1-2008, called with the following arguments:

- 99945 a. The *source\_file* operand is used as the *old* argument.

- b. The destination path is used as the *new* argument.

If this succeeds, *mv* shall do nothing more with the current *source\_file* and go on to any remaining *source\_files*. If this fails for any reasons other than those described for the *errno* [EXDEV] in the System Interfaces volume of POSIX.1-2008, *mv* shall write a diagnostic message to standard error, do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.

4. If the destination path exists, and it is a file of type directory and *source\_file* is not a file of type directory, or it is a file not of type directory and *source\_file* is a file of type directory, *mv* shall write a diagnostic message to standard error, do nothing more with the current *source\_file*, and go on to any remaining *source\_files*. If the destination path exists and was created by a previous step, it is unspecified whether this will be treated as an error or the destination path will be overwritten.
5. If the destination path exists, *mv* shall attempt to remove it. If this fails for any reason, *mv* shall write a diagnostic message to standard error, do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.
6. The file hierarchy rooted in *source\_file* shall be duplicated as a file hierarchy rooted in the destination path. If *source\_file* or any of the files below it in the hierarchy are symbolic links, the links themselves shall be duplicated, including their contents, rather than any files to which they refer. The following characteristics of each file in the file hierarchy shall be duplicated:

- The time of last data modification and time of last access
- The user ID and group ID
- The file mode

If the user ID, group ID, or file mode of a regular file cannot be duplicated, the file mode bits S\_ISUID and S\_ISGID shall not be duplicated.

When files are duplicated to another file system, the implementation may require that the process invoking *mv* has read access to each file being duplicated.

If files being duplicated to another file system have hard links to other files, it is unspecified whether the files copied to the new file system have the hard links preserved or separate copies are created for the linked files.

If the duplication of the file hierarchy fails for any reason, *mv* shall write a diagnostic message to standard error, do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.

If the duplication of the file characteristics fails for any reason, *mv* shall write a diagnostic message to standard error, but this failure shall not cause *mv* to modify its exit status.

7. The file hierarchy rooted in *source\_file* shall be removed. If this fails for any reason, *mv* shall write a diagnostic message to the standard error, do nothing more with the current *source\_file*, and go on to any remaining *source\_files*.

## OPTIONS

The *mv* utility shall conform to XBD [Section 12.2](#) (on page 216).

The following options shall be supported:

- f** Do not prompt for confirmation if the destination path exists. Any previous occurrence of the **-i** option is ignored.

99989            **-i**            Prompt for confirmation if the destination path exists. Any previous occurrence of  
 99990                            the **-f** option is ignored.

99991            Specifying more than one of the **-f** or **-i** options shall not be considered an error. The last option  
 99992            specified shall determine the behavior of *mv*.

#### 99993 **OPERANDS**

99994            The following operands shall be supported:

99995            *source\_file*    A pathname of a file or directory to be moved.

99996            *target\_file*    A new pathname for the file or directory being moved.

99997            *target\_dir*     A pathname of an existing directory into which to move the input files.

#### 99998 **STDIN**

99999            The standard input shall be used to read an input line in response to each prompt specified in  
 100000            the **STDERR** section. Otherwise, the standard input shall not be used.

#### 100001 **INPUT FILES**

100002            The input files specified by each *source\_file* operand can be of any file type.

#### 100003 **ENVIRONMENT VARIABLES**

100004            The following environment variables shall affect the execution of *mv*:

100005            *LANG*            Provide a default value for the internationalization variables that are unset or null.  
 100006                            (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 100007                            variables used to determine the values of locale categories.)

100008            *LC\_ALL*          If set to a non-empty string value, override the values of all the other  
 100009                            internationalization variables.

100010            *LC\_COLLATE*

100011                            Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 100012                            character collating elements used in the extended regular expression defined for  
 100013                            the **yesexpr** locale keyword in the *LC\_MESSAGES* category.

100014            *LC\_CTYPE*       Determine the locale for the interpretation of sequences of bytes of text data as  
 100015                            characters (for example, single-byte as opposed to multi-byte characters in  
 100016                            arguments and input files), the behavior of character classes used in the extended  
 100017                            regular expression defined for the **yesexpr** locale keyword in the *LC\_MESSAGES*  
 100018                            category.

100019            *LC\_MESSAGES*

100020                            Determine the locale used to process affirmative responses, and the locale used to  
 100021                            affect the format and contents of diagnostic messages and prompts written to  
 100022                            standard error.

100023 **XSI**            *NLSPATH*       Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

#### 100024 **ASYNCHRONOUS EVENTS**

100025            Default.

#### 100026 **STDOUT**

100027            Not used.

#### 100028 **STDERR**

100029            Prompts shall be written to the standard error under the conditions specified in the  
 100030            **DESCRIPTION** section. The prompts shall contain the destination pathname, but their format is  
 100031            otherwise unspecified. Otherwise, the standard error shall be used only for diagnostic

100032 messages.

#### 100033 OUTPUT FILES

100034 The output files may be of any file type.

#### 100035 EXTENDED DESCRIPTION

100036 None.

#### 100037 EXIT STATUS

100038 The following exit values shall be returned:

100039 0 All input files were moved successfully.

100040 >0 An error occurred.

#### 100041 CONSEQUENCES OF ERRORS

100042 If the copying or removal of *source\_file* is prematurely terminated by a signal or error, *mv* may  
 100043 leave a partial copy of *source\_file* at the source or destination. The *mv* utility shall not modify  
 100044 both *source\_file* and the destination path simultaneously; termination at any point shall leave  
 100045 either *source\_file* or the destination path complete.

#### 100046 APPLICATION USAGE

100047 Some implementations mark for update the last file status change timestamp of renamed files  
 100048 and some do not. Applications which make use of the last file status change timestamp may  
 100049 behave differently with respect to renamed files unless they are designed to allow for either  
 100050 behavior.

100051 The specification ensures that *mv a a* will not alter the contents of file **a**, and allows the  
 100052 implementation to issue an error that a file cannot be moved onto itself. Likewise, when **a** and **b**  
 100053 are hard links to the same file, *mv a b* will not alter **b**, but if a diagnostic is not issued, then it is  
 100054 unspecified whether **a** is left untouched (as it would be by the *rename()* function) or unlinked  
 100055 (reducing the link count of **b**).

#### 100056 EXAMPLES

100057 If the current directory contains only files **a** (of any type defined by the System Interfaces  
 100058 volume of POSIX.1-2008), **b** (also of any type), and a directory **c**:

100059 `mv a b c`

100060 `mv c d`

100061 results with the original files **a** and **b** residing in the directory **d** in the current directory.

#### 100062 RATIONALE

100063 Early proposals diverged from the SVID and BSD historical practice in that they required that  
 100064 when the destination path exists, the `-f` option is not specified, and input is not a terminal, *mv*  
 100065 fails. This was done for compatibility with *cp*. The current text returns to historical practice. It  
 100066 should be noted that this is consistent with the *rename()* function defined in the System  
 100067 Interfaces volume of POSIX.1-2008, which does not require write permission on the target.

100068 For absolute clarity, paragraph (1), describing the behavior of *mv* when prompting for  
 100069 confirmation, should be interpreted in the following manner:

100070 `if (exists AND (NOT f_option) AND`  
 100071 `((not_writable AND input_is_terminal) OR i_option))`

100072 The `-i` option exists on BSD systems, giving applications and users a way to avoid accidentally  
 100073 unlinking files when moving others. When the standard input is not a terminal, the 4.3 BSD *mv*  
 100074 deletes all existing destination paths without prompting, even when `-i` is specified; this is  
 100075 inconsistent with the behavior of the 4.3 BSD *cp* utility, which always generates an error when

the file is unwritable and the standard input is not a terminal. The standard developers decided that use of `-i` is a request for interaction, so when the destination path exists, the utility takes instructions from whatever responds to standard input.

The `rename()` function is able to move directories within the same file system. Some historical versions of `mv` have been able to move directories, but not to a different file system. The standard developers considered that this was an annoying inconsistency, so this volume of POSIX.1-2008 requires directories to be able to be moved even across file systems. There is no `-R` option to confirm that moving a directory is actually intended, since such an option was not required for moving directories in historical practice. Requiring the application to specify it sometimes, depending on the destination, seemed just as inconsistent. The semantics of the `rename()` function were preserved as much as possible. For example, `mv` is not permitted to “rename” files to or from directories, even though they might be empty and removable.

Historic implementations of `mv` did not exit with a non-zero exit status if they were unable to duplicate any file characteristics when moving a file across file systems, nor did they write a diagnostic message for the user. The former behavior has been preserved to prevent scripts from breaking; a diagnostic message is now required, however, so that users are alerted that the file characteristics have changed.

The exact format of the interactive prompts is unspecified. Only the general nature of the contents of prompts are specified because implementations may desire more descriptive prompts than those used on historical implementations. Therefore, an application not using the `-f` option or using the `-i` option relies on the system to provide the most suitable dialog directly with the user, based on the behavior specified.

When `mv` is dealing with a single file system and `source_file` is a symbolic link, the link itself is moved as a consequence of the dependence on the `rename()` functionality, per the DESCRIPTION. Across file systems, this has to be made explicit.

## FUTURE DIRECTIONS

None.

## SEE ALSO

`cp`, `ln`

XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

XSH `rename()`

## CHANGE HISTORY

First released in Issue 2.

### Issue 6

The `mv` utility is changed to describe processing of symbolic links as specified in the IEEE P1003.2b draft standard.

The APPLICATION USAGE section is added.

### Issue 7

Austin Group Interpretation 1003.1-2001 #016 is applied.

Austin Group Interpretation 1003.1-2001 #126 is applied, changing the description of the `LC_MESSAGES` environment variable.

Austin Group Interpretations 1003.1-2001 #164, #168, and #169 are applied.

SD5-XCU-ERN-13 is applied, making an editorial correction to the SYNOPSIS.

SD5-XCU-ERN-51 is applied to the DESCRIPTION, defining the behavior for when files are

- 100120 being duplicated to another file system while having hard links.
- 100121 Changes are made related to support for finegrained timestamps.
- 100122 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0124 [48] is applied.



100123 **NAME**

100124       newgrp — change to a new group

100125 **SYNOPSIS**

100126       newgrp [-l] [group]

100127 **DESCRIPTION**

100128       The *newgrp* utility shall create a new shell execution environment with a new real and effective  
 100129       group identification. Of the attributes listed in [Section 2.12](#) (on page 2371), the new shell  
 100130       execution environment shall retain the working directory, file creation mask, and exported  
 100131       variables from the previous environment (that is, open files, traps, unexported variables, alias  
 100132       definitions, shell functions, and *set* options may be lost). All other aspects of the process  
 100133       environment that are preserved by the *exec* family of functions defined in the System Interfaces  
 100134       volume of POSIX.1-2008 shall also be preserved by *newgrp*; whether other aspects are preserved  
 100135       is unspecified.

100136       A failure to assign the new group identifications (for example, for security or password-related  
 100137       reasons) shall not prevent the new shell execution environment from being created.

100138       The *newgrp* utility shall affect the supplemental groups for the process as follows:

- 100139       • On systems where the effective group ID is normally in the supplementary group list (or  
 100140       whenever the old effective group ID actually is in the supplementary group list):
  - 100141           — If the new effective group ID is also in the supplementary group list, *newgrp* shall  
 100142           change the effective group ID.
  - 100143           — If the new effective group ID is not in the supplementary group list, *newgrp* shall add  
 100144           the new effective group ID to the list, if there is room to add it.
- 100145       • On systems where the effective group ID is not normally in the supplementary group list  
 100146       (or whenever the old effective group ID is not in the supplementary group list):
  - 100147           — If the new effective group ID is in the supplementary group list, *newgrp* shall delete  
 100148           it.
  - 100149           — If the old effective group ID is not in the supplementary list, *newgrp* shall add it if  
 100150           there is room.

100151       **Note:**     The System Interfaces volume of POSIX.1-2008 does not specify whether the effective group ID  
 100152       of a process is included in its supplementary group list.

100153       With no operands, *newgrp* shall change the effective group back to the groups identified in the  
 100154       user's user entry, and shall set the list of supplementary groups to that set in the user's group  
 100155       database entries.

100156       If the first argument is *'-'*, the results are unspecified.

100157       If a password is required for the specified group, and the user is not listed as a member of that  
 100158       group in the group database, the user shall be prompted to enter the correct password for that  
 100159       group. If the user is listed as a member of that group, no password shall be requested. If no  
 100160       password is required for the specified group, it is implementation-defined whether users not  
 100161       listed as members of that group can change to that group. Whether or not a password is  
 100162       required, implementation-defined system accounting or security mechanisms may impose  
 100163       additional authorization restrictions that may cause *newgrp* to write a diagnostic message and  
 100164       suppress the changing of the group identification.



100165 **OPTIONS**

100166 The *newgrp* utility shall conform to XBD [Section 12.2](#) (on page 216), except for the unspecified  
 100167 usage of `'-'`.

100168 The following option shall be supported:

100169 `-l` (The letter ell.) Change the environment to what would be expected if the user  
 100170 actually logged in again.

100171 **OPERANDS**

100172 The following operand shall be supported:

100173 *group* A group name from the group database or a non-negative numeric group ID.  
 100174 Specifies the group ID to which the real and effective group IDs shall be set. If  
 100175 *group* is a non-negative numeric string and exists in the group database as a group  
 100176 name (see *getgrnam()*), the numeric group ID associated with that group name  
 100177 shall be used as the group ID.

100178 **STDIN**

100179 Not used.

100180 **INPUT FILES**

100181 The file `/dev/tty` shall be used to read a single line of text for password checking, when one is  
 100182 required.

100183 **ENVIRONMENT VARIABLES**

100184 The following environment variables shall affect the execution of *newgrp*:

100185 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 100186 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 100187 variables used to determine the values of locale categories.)

100188 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 100189 internationalization variables.

100190 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 100191 characters (for example, single-byte as opposed to multi-byte characters in  
 100192 arguments).

100193 *LC\_MESSAGES*

100194 Determine the locale that should be used to affect the format and contents of  
 100195 diagnostic messages written to standard error.

100196 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

100197 **ASYNCHRONOUS EVENTS**

100198 Default.

100199 **STDOUT**

100200 Not used.

100201 **STDERR**

100202 The standard error shall be used for diagnostic messages and a prompt string for a password, if  
 100203 one is required. Diagnostic messages may be written in cases where the exit status is not  
 100204 available. See the EXIT STATUS section.

**100205 OUTPUT FILES**

100206 None.

**100207 EXTENDED DESCRIPTION**

100208 None.

**100209 EXIT STATUS**

100210 If *newgrp* succeeds in creating a new shell execution environment, whether or not the group  
100211 identification was changed successfully, the exit status shall be the exit status of the shell.  
100212 Otherwise, the following exit value shall be returned:

100213 >0 An error occurred.

**100214 CONSEQUENCES OF ERRORS**

100215 The invoking shell may terminate.

**100216 APPLICATION USAGE**

100217 There is no convenient way to enter a password into the group database. Use of group  
100218 passwords is not encouraged, because by their very nature they encourage poor security  
100219 practices. Group passwords may disappear in the future.

100220 A common implementation of *newgrp* is that the current shell uses *exec* to overlay itself with  
100221 *newgrp*, which in turn overlays itself with a new shell after changing group. On some  
100222 implementations, however, this may not occur and *newgrp* may be invoked as a subprocess.

100223 The *newgrp* command is intended only for use from an interactive terminal. It does not offer a  
100224 useful interface for the support of applications.

100225 The exit status of *newgrp* is generally inapplicable. If *newgrp* is used in a script, in most cases it  
100226 successfully invokes a new shell and the rest of the original shell script is bypassed when the  
100227 new shell exits. Used interactively, *newgrp* displays diagnostic messages to indicate problems.  
100228 But usage such as:

100229 *newgrp* foo  
100230 *echo* \$?

100231 is not useful because the new shell might not have access to any status *newgrp* may have  
100232 generated (and most historical systems do not provide this status). A zero status echoed here  
100233 does not necessarily indicate that the user has changed to the new group successfully. Following  
100234 *newgrp* with the *id* command provides a portable means of determining whether the group  
100235 change was successful or not.

**100236 EXAMPLES**

100237 None.

**100238 RATIONALE**

100239 Most historical implementations use one of the *exec* functions to implement the behavior of  
100240 *newgrp*. Errors detected before the *exec* leave the environment unchanged, while errors detected  
100241 after the *exec* leave the user in a changed environment. While it would be useful to have *newgrp*  
100242 issue a diagnostic message to tell the user that the environment changed, it would be  
100243 inappropriate to require this change to some historical implementations.

100244 The password mechanism is allowed in the group database, but how this would be  
100245 implemented is not specified.

100246 The *newgrp* utility was retained in this volume of POSIX.1-2008, even given the existence of the  
100247 multiple group permissions feature in the System Interfaces volume of POSIX.1-2008, for several  
100248 reasons. First, in some implementations, the group ownership of a newly created file is  
100249 determined by the group of the directory in which the file is created, as allowed by the System

100250 Interfaces volume of POSIX.1-2008; on other implementations, the group ownership of a newly  
100251 created file is determined by the effective group ID. On implementations of the latter type,  
100252 *newgrp* allows files to be created with a specific group ownership. Finally, many  
100253 implementations use the real group ID in accounting, and on such systems, *newgrp* allows the  
100254 accounting identity of the user to be changed.

#### 100255 **FUTURE DIRECTIONS**

100256 None.

#### 100257 **SEE ALSO**

100258 [Chapter 2](#) (on page 2335), *sh*

100259 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

100260 XSH *exec*, *getgrnam()*

#### 100261 **CHANGE HISTORY**

100262 First released in Issue 2.

#### 100263 **Issue 6**

100264 This utility is marked as part of the User Portability Utilities option.

100265 The obsolescent SYNOPSIS is removed.

100266 The text describing supplemental groups is no longer conditional on {NGROUPS\_MAX} being  
100267 greater than 1. This is because {NGROUPS\_MAX} now has a minimum value of 8. This is a FIPS  
100268 requirement.

#### 100269 **Issue 7**

100270 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying the behavior if the first  
100271 argument is '- '.

100272 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

100273 The *newgrp* utility is moved from the User Portability Utilities option to the Base. User  
100274 Portability Utilities is now an option for interactive utilities.

100275 **NAME**

100276        *nice* — invoke a utility with an altered nice value

100277 **SYNOPSIS**

100278        *nice* [*-n increment*] *utility* [*argument...*]

100279 **DESCRIPTION**

100280        The *nice* utility shall invoke a utility, requesting that it be run with a different nice value (see  
100281        XBD [Section 3.244](#), on page 72). With no options, the executed utility shall be run with a nice  
100282        value that is some implementation-defined quantity greater than or equal to the nice value of the  
100283        current process. If the user lacks appropriate privileges to affect the nice value in the requested  
100284        manner, the *nice* utility shall not affect the nice value; in this case, a warning message may be  
100285        written to standard error, but this shall not prevent the invocation of *utility* or affect the exit  
100286        status.

100287 **OPTIONS**

100288        The *nice* utility shall conform to XBD [Section 12.2](#) (on page 216).

100289        The following option is supported:

100290        *-n increment* A positive or negative decimal integer which shall have the same effect on the  
100291        execution of the utility as if the utility had called the *nice()* function with the  
100292        numeric value of the *increment* option-argument.

100293 **OPERANDS**

100294        The following operands shall be supported:

100295        *utility*        The name of a utility that is to be invoked. If the *utility* operand names any of the  
100296        special built-in utilities in [Section 2.14](#) (on page 2374), the results are undefined.

100297        *argument*      Any string to be supplied as an argument when invoking the utility named by the  
100298        *utility* operand.

100299 **STDIN**

100300        Not used.

100301 **INPUT FILES**

100302        None.

100303 **ENVIRONMENT VARIABLES**

100304        The following environment variables shall affect the execution of *nice*:

100305        *LANG*        Provide a default value for the internationalization variables that are unset or null.  
100306        (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
100307        variables used to determine the values of locale categories.)

100308        *LC\_ALL*        If set to a non-empty string value, override the values of all the other  
100309        internationalization variables.

100310        *LC\_CTYPE*      Determine the locale for the interpretation of sequences of bytes of text data as  
100311        characters (for example, single-byte as opposed to multi-byte characters in  
100312        arguments).

100313        *LC\_MESSAGES*

100314        Determine the locale that should be used to affect the format and contents of  
100315        diagnostic messages written to standard error.

100316 XSI        *NLSPATH*        Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

|        |                                                                                                                                                     |                                                                             |
|--------|-----------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------|
| 100317 | <i>PATH</i>                                                                                                                                         | Determine the search path used to locate the utility to be invoked. See XBD |
| 100318 |                                                                                                                                                     | Chapter 8 (on page 173).                                                    |
| 100319 | <b>ASYNCHRONOUS EVENTS</b>                                                                                                                          |                                                                             |
| 100320 | Default.                                                                                                                                            |                                                                             |
| 100321 | <b>STDOUT</b>                                                                                                                                       |                                                                             |
| 100322 | Not used.                                                                                                                                           |                                                                             |
| 100323 | <b>STDERR</b>                                                                                                                                       |                                                                             |
| 100324 | The standard error shall be used only for diagnostic messages.                                                                                      |                                                                             |
| 100325 | <b>OUTPUT FILES</b>                                                                                                                                 |                                                                             |
| 100326 | None.                                                                                                                                               |                                                                             |
| 100327 | <b>EXTENDED DESCRIPTION</b>                                                                                                                         |                                                                             |
| 100328 | None.                                                                                                                                               |                                                                             |
| 100329 | <b>EXIT STATUS</b>                                                                                                                                  |                                                                             |
| 100330 | If <i>utility</i> is invoked, the exit status of <i>nice</i> shall be the exit status of <i>utility</i> ; otherwise, the <i>nice</i>                |                                                                             |
| 100331 | utility shall exit with one of the following values:                                                                                                |                                                                             |
| 100332 | 1-125                                                                                                                                               | An error occurred in the <i>nice</i> utility.                               |
| 100333 | 126                                                                                                                                                 | The utility specified by <i>utility</i> was found but could not be invoked. |
| 100334 | 127                                                                                                                                                 | The utility specified by <i>utility</i> could not be found.                 |
| 100335 | <b>CONSEQUENCES OF ERRORS</b>                                                                                                                       |                                                                             |
| 100336 | Default.                                                                                                                                            |                                                                             |
| 100337 | <b>APPLICATION USAGE</b>                                                                                                                            |                                                                             |
| 100338 | The only guaranteed portable uses of this utility are:                                                                                              |                                                                             |
| 100339 | <i>nice utility</i>                                                                                                                                 |                                                                             |
| 100340 | Run <i>utility</i> with the default higher or equal nice value.                                                                                     |                                                                             |
| 100341 | <i>nice -n &lt;positive integer&gt; utility</i>                                                                                                     |                                                                             |
| 100342 | Run <i>utility</i> with a higher nice value.                                                                                                        |                                                                             |
| 100343 | On some implementations they have no discernible effect on the invoked utility and on some                                                          |                                                                             |
| 100344 | others they are exactly equivalent.                                                                                                                 |                                                                             |
| 100345 | Historical systems have frequently supported the <i>&lt;positive integer&gt;</i> up to 20. Since there is no                                        |                                                                             |
| 100346 | error penalty associated with guessing a number that is too high, users without access to the                                                       |                                                                             |
| 100347 | system conformance document (to see what limits are actually in place) could use the historical 1                                                   |                                                                             |
| 100348 | to 20 range or attempt to use very large numbers if the job should be truly low priority.                                                           |                                                                             |
| 100349 | The nice value of a process can be displayed using the command:                                                                                     |                                                                             |
| 100350 | ps -o nice                                                                                                                                          |                                                                             |
| 100351 | The <i>command</i> , <i>env</i> , <i>nice</i> , <i>nohup</i> , <i>time</i> , and <i>xargs</i> utilities have been specified to use exit code 127 if |                                                                             |
| 100352 | an error occurs so that applications can distinguish “failure to find a utility” from “invoked                                                      |                                                                             |
| 100353 | utility exited with an error indication”. The value 127 was chosen because it is not commonly                                                       |                                                                             |
| 100354 | used for other meanings; most utilities use small values for “normal error conditions” and the                                                      |                                                                             |
| 100355 | values above 128 can be confused with termination due to receipt of a signal. The value 126 was                                                     |                                                                             |
| 100356 | chosen in a similar manner to indicate that the utility could be found, but not invoked. Some                                                       |                                                                             |
| 100357 | scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction                                                    |                                                                             |
| 100358 | between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to                                                    |                                                                             |
| 100359 | exec the utility fail with [ENOENT], and uses 126 when any attempt to exec the utility fails for                                                    |                                                                             |

100360 any other reason.

#### 100361 EXAMPLES

100362 None.

#### 100363 RATIONALE

100364 The 4.3 BSD version of *nice* does not check whether *increment* is a valid decimal integer. The  
100365 command *nice -x utility*, for example, would be treated the same as the command *nice --1*  
100366 *utility*. If the user does not have appropriate privileges, this results in a “permission denied”  
100367 error. This is considered a bug.

100368 When a user without appropriate privileges gives a negative *increment*, System V treats it like  
100369 the command *nice -0 utility*, while 4.3 BSD writes a “permission denied” message and does not  
100370 run the utility. The standard specifies the System V behavior together with an optional BSD-style  
100371 “permission denied” message.

100372 The C shell has a built-in version of *nice* that has a different interface from the one described in  
100373 this volume of POSIX.1-2008.

100374 The term “utility” is used, rather than “command”, to highlight the fact that shell compound  
100375 commands, pipelines, and so on, cannot be used. Special built-ins also cannot be used.  
100376 However, “utility” includes user application programs and shell scripts, not just utilities defined  
100377 in this volume of POSIX.1-2008.

100378 Historical implementations of *nice* provide a nice value range of 40 or 41 discrete steps, with the  
100379 default nice value being the midpoint of that range. By default, they raise the nice value of the  
100380 executed utility by 10.

100381 Some historical documentation states that the *increment* value must be within a fixed range. This  
100382 is misleading; the valid *increment* values on any invocation are determined by the current  
100383 process nice value, which is not always the default.

100384 The definition of nice value is not intended to suggest that all processes in a system have  
100385 priorities that are comparable. Scheduling policy extensions such as the realtime priorities in the  
100386 System Interfaces volume of POSIX.1-2008 make the notion of a single underlying priority for all  
100387 scheduling policies problematic. Some implementations may implement the *nice*-related features  
100388 to affect all processes on the system, others to affect just the general time-sharing activities  
100389 implied by this volume of POSIX.1-2008, and others may have no effect at all. Because of the use  
100390 of “implementation-defined” in *nice* and *renice*, a wide range of implementation strategies are  
100391 possible.

100392 Earlier versions of this standard allowed a *-increment* option. This form is no longer specified by  
100393 POSIX.1-2008 but may be present in some implementations.

#### 100394 FUTURE DIRECTIONS

100395 None.

#### 100396 SEE ALSO

100397 [Chapter 2](#) (on page 2335), *renice*

100398 [XBD Section 3.244](#) (on page 72), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

100399 [XSH \*nice\*\(\)](#)

#### 100400 CHANGE HISTORY

100401 First released in Issue 4.

100402 **Issue 6**

100403 This utility is marked as part of the User Portability Utilities option.

100404 The obsolescent SYNOPSIS is removed.

100405 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/18 is applied, deleting a paragraph of  
100406 RATIONALE that referred to text no longer in the standard.

100407 **Issue 7**

100408 Austin Group Interpretation 1003.1-2001 #027 is applied.

100409 SD5-XCU-ERN-32 and SD5-XCU-ERN-33 are applied, updating the DESCRIPTION,  
100410 APPLICATION USAGE, and RATIONALE sections.

100411 The *nice* utility is moved from the User Portability Utilities option to the Base. User Portability  
100412 Utilities is now an option for interactive utilities.

100413 **NAME**

100414 nl — line numbering filter

100415 **SYNOPSIS**

```
100416 XSI nl [-p] [-b type] [-d delim] [-f type] [-h type] [-i incr] [-l num]
100417          [-n format] [-s sep] [-v startnum] [-w width] [file]
```

100418 **DESCRIPTION**

100419 The *nl* utility shall read lines from the named *file* or the standard input if no *file* is named and  
 100420 shall reproduce the lines to standard output. Lines shall be numbered on the left. Additional  
 100421 functionality may be provided in accordance with the command options in effect.

100422 The *nl* utility views the text it reads in terms of logical pages. Line numbering shall be reset at  
 100423 the start of each logical page. A logical page consists of a header, a body, and a footer section.  
 100424 Empty sections are valid. Different line numbering options are independently available for  
 100425 header, body, and footer (for example, no numbering of header and footer lines while  
 100426 numbering blank lines only in the body).

100427 The starts of logical page sections shall be signaled by input lines containing nothing but the  
 100428 following delimiter characters:

| Line     | Start of |
|----------|----------|
| \: \: \: | Header   |
| \: \:    | Body     |
| \:       | Footer   |

100433 Unless otherwise specified, *nl* shall assume the text being read is in a single logical page body.

100434 **OPTIONS**

100435 The *nl* utility shall conform to XBD [Section 12.2](#) (on page 216). Only one file can be named.

100436 The following options shall be supported:

100437 **-b type** Specify which logical page body lines shall be numbered. Recognized *types* and  
 100438 their meaning are:

100439 **a** Number all lines.

100440 **t** Number only non-empty lines.

100441 **n** No line numbering.

100442 **pstring** Number only lines that contain the basic regular expression specified in  
 100443 *string*.

100444 The default *type* for logical page body shall be **t** (text lines numbered).

100445 **-d delim** Specify the delimiter characters that indicate the start of a logical page section.  
 100446 These can be changed from the default characters "\: " to two user-specified  
 100447 characters. If only one character is entered, the second character shall remain the  
 100448 default character ' : '.

100449 **-f type** Specify the same as **b type** except for footer. The default for logical page footer shall  
 100450 be **n** (no lines numbered).

100451 **-h type** Specify the same as **b type** except for header. The default *type* for logical page  
 100452 header shall be **n** (no lines numbered).



|        |                           |                                                                                                                                                                                                                                                                                         |
|--------|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 100453 | <b>-i</b> <i>incr</i>     | Specify the increment value used to number logical page lines. The default shall be 1.                                                                                                                                                                                                  |
| 100454 |                           |                                                                                                                                                                                                                                                                                         |
| 100455 | <b>-l</b> <i>num</i>      | Specify the number of blank lines to be considered as one. For example, <b>-l 2</b> results in only the second adjacent blank line being numbered (if the appropriate <b>-h a</b> , <b>-b a</b> , or <b>-f a</b> option is set). The default shall be 1.                                |
| 100456 |                           |                                                                                                                                                                                                                                                                                         |
| 100457 |                           |                                                                                                                                                                                                                                                                                         |
| 100458 | <b>-n</b> <i>format</i>   | Specify the line numbering format. Recognized values are: <b>ln</b> , left justified, leading zeros suppressed; <b>rn</b> , right justified, leading zeros suppressed; <b>rz</b> , right justified, leading zeros kept. The default <i>format</i> shall be <b>rn</b> (right justified). |
| 100459 |                           |                                                                                                                                                                                                                                                                                         |
| 100460 |                           |                                                                                                                                                                                                                                                                                         |
| 100461 | <b>-p</b>                 | Specify that numbering should not be restarted at logical page delimiters.                                                                                                                                                                                                              |
| 100462 | <b>-s</b> <i>sep</i>      | Specify the characters used in separating the line number and the corresponding text line. The default <i>sep</i> shall be a <tab>.                                                                                                                                                     |
| 100463 |                           |                                                                                                                                                                                                                                                                                         |
| 100464 | <b>-v</b> <i>startnum</i> | Specify the initial value used to number logical page lines. The default shall be 1.                                                                                                                                                                                                    |
| 100465 | <b>-w</b> <i>width</i>    | Specify the number of characters to be used for the line number. The default <i>width</i> shall be 6.                                                                                                                                                                                   |
| 100466 |                           |                                                                                                                                                                                                                                                                                         |

#### 100467 OPERANDS

100468 The following operand shall be supported:

100469 *file* A pathname of a text file to be line-numbered.

#### 100470 STDIN

100471 The standard input shall be used if no *file* operand is specified, and shall be used if the *file*  
 100472 operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
 100473 the standard input shall not be used. See the INPUT FILES section.

#### 100474 INPUT FILES

100475 The input file shall be a text file.

#### 100476 ENVIRONMENT VARIABLES

100477 The following environment variables shall affect the execution of *nl*:

100478 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 100479 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 100480 variables used to determine the values of locale categories.)

100481 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 100482 internationalization variables.

100483 *LC\_COLLATE*

100484 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 100485 character collating elements within regular expressions.

100486 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 100487 characters (for example, single-byte as opposed to multi-byte characters in  
 100488 arguments and input files), the behavior of character classes within regular  
 100489 expressions, and for deciding which characters are in character class **graph** (for the  
 100490 **-b t**, **-f t**, and **-h t** options).

100491 *LC\_MESSAGES*

100492 Determine the locale that should be used to affect the format and contents of  
 100493 diagnostic messages written to standard error.

100494 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

100495 **ASYNCHRONOUS EVENTS**

100496 Default.

100497 **STDOUT**

100498 The standard output shall be a text file in the following format:

100499 "%s%s%s", *<line number>*, *<separator>*, *<input line>*

100500 where *<line number>* is one of the following numeric formats:

100501 %6d When the **rn** format is used (the default; see **-n**).

100502 %06d When the **rz** format is used.

100503 %-6d When the **ln** format is used.

100504 *<empty>* When line numbers are suppressed for a portion of the page; the *<separator>* is also suppressed.

100505

100506 In the preceding list, the number 6 is the default width; the **-w** option can change this value.

100507 **STDERR**

100508 The standard error shall be used only for diagnostic messages.

100509 **OUTPUT FILES**

100510 None.

100511 **EXTENDED DESCRIPTION**

100512 None.

100513 **EXIT STATUS**

100514 The following exit values shall be returned:

100515 0 Successful completion.

100516 >0 An error occurred.

100517 **CONSEQUENCES OF ERRORS**

100518 Default.

100519 **APPLICATION USAGE**

100520 In using the **-d delim** option, care should be taken to escape characters that have special meaning to the command interpreter.

100521

100522 **EXAMPLES**

100523 The command:

100524 `nl -v 10 -i 10 -d \!+ file1`

100525 numbers *file1* starting at line number 10 with an increment of 10. The logical page delimiter is

100526 "**!+**". Note that the **'!'** has to be escaped when using *cs**h* as a command interpreter because of

100527 its history substitution syntax. For *ks**h* and *sh* the escape is not necessary, but does not do any

100528 harm.

100529 **RATIONALE**

100530 None.

100531 **FUTURE DIRECTIONS**

100532 None.

100533 **SEE ALSO**100534 *pr*100535 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)100536 **CHANGE HISTORY**

100537 First released in Issue 2.

100538 **Issue 5**100539 The option `[-f type]` is added to the SYNOPSIS. The option descriptions are presented in  
100540 alphabetic order. The description of `-bt` is changed to “Number only non-empty lines”.100541 **Issue 6**100542 The obsolescent behavior allowing the options to be intermingled with the optional *file* operand  
100543 is removed.100544 **Issue 7**

100545 Austin Group Interpretation 1003.1-2001 #092 is applied.

100546 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

100547 **NAME**100548 nm — write the name list of an object file (**DEVELOPMENT**)100549 **SYNOPSIS**100550 SD nm [-APv] [-g|-u] [-t *format*] *file*...100551 XSI nm [-APv] [-efox] [-g|-u] [-t *format*] *file*...100552 **DESCRIPTION**

100553 The *nm* utility shall display symbolic information appearing in the object file, executable file, or  
 100554 object-file library named by *file*. If no symbolic information is available for a valid input file, the  
 100555 *nm* utility shall report that fact, but not consider it an error condition.

100556 XSI The default base used when numeric values are written is unspecified. On XSI-conformant  
 100557 systems, it shall be decimal if the **-P** option is not specified.

100558 **OPTIONS**100559 The *nm* utility shall conform to XBD [Section 12.2](#) (on page 216).

100560 The following options shall be supported:

100561 **-A** Write the full pathname or library name of an object on each line.100562 XSI **-e** Write only external (global) and static symbol information.

100563 XSI **-f** Produce full output. Write redundant symbols (**.text**, **.data**, and **.bss**), normally  
 100564 suppressed.

100565 **-g** Write only external (global) symbol information.100566 XSI **-o** Write numeric values in octal (equivalent to **-t o**).100567 **-P** Write information in a portable output format, as specified in the STDOUT section.

100568 **-t *format*** Write each numeric value in the specified format. The format shall be dependent  
 100569 on the single character used as the *format* option-argument:

100570 XSI d decimal (default if **-P** is not specified).

100571 o octal.

100572 x hexadecimal (default if **-P** is specified).100573 **-u** Write only undefined symbols.100574 **-v** Sort output by value instead of by symbol name.100575 XSI **-x** Write numeric values in hexadecimal (equivalent to **-t x**).100576 **OPERANDS**

100577 The following operand shall be supported:

100578 *file* A pathname of an object file, executable file, or object-file library.100579 **STDIN**

100580 See the INPUT FILES section.

100581 **INPUT FILES**

100582 The input file shall be an object file, an object-file library whose format is the same as those  
 100583 produced by the *ar* utility for link editing, or an executable file. The *nm* utility may accept  
 100584 additional implementation-defined object library formats for the input file.

100585 **ENVIRONMENT VARIABLES**

100586 The following environment variables shall affect the execution of *nm*:

100587 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 100588 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 100589 variables used to determine the values of locale categories.)

100590 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 100591 internationalization variables.

100592 *LC\_COLLATE*  
 100593 Determine the locale for character collation information for the symbol-name and  
 100594 symbol-value collation sequences.

100595 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 100596 characters (for example, single-byte as opposed to multi-byte characters in  
 100597 arguments).

100598 *LC\_MESSAGES*  
 100599 Determine the locale that should be used to affect the format and contents of  
 100600 diagnostic messages written to standard error.

100601 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

100602 **ASYNCHRONOUS EVENTS**

100603 Default.

100604 **STDOUT**

100605 If symbolic information is present in the input files, then for each file or for each member of an  
 100606 archive, the *nm* utility shall write the following information to standard output. By default, the  
 100607 format is unspecified, but the output shall be sorted by symbol name according to the collation  
 100608 sequence in the current locale.

- 100609 • Library or object name, if *-A* is specified
- 100610 • Symbol name
- 100611 • Symbol type, which shall either be one of the following single characters or an  
 100612 implementation-defined type represented by a single character:
  - 100613 A Global absolute symbol.
  - 100614 a Local absolute symbol.
  - 100615 B Global “bss” (that is, uninitialized data space) symbol.
  - 100616 b Local bss symbol.
  - 100617 D Global data symbol.
  - 100618 d Local data symbol.
  - 100619 T Global text symbol.
  - 100620 t Local text symbol.
  - 100621 U Undefined symbol.
- 100622 • Value of the symbol
- 100623 • The size associated with the symbol, if applicable

100624 This information may be supplemented by additional information specific to the

100625 implementation.

100626 If the **-P** option is specified, the previous information shall be displayed using the following  
 100627 portable format. The three versions differ depending on whether **-t d**, **-t o**, or **-t x** was specified,  
 100628 respectively:

100629 "%s%s %s %d %d\n", <library/object name>, <name>, <type>,  
 100630 <value>, <size>

100631 "%s%s %s %o %o\n", <library/object name>, <name>, <type>,  
 100632 <value>, <size>

100633 "%s%s %s %x %x\n", <library/object name>, <name>, <type>,  
 100634 <value>, <size>

100635 where <library/object name> shall be formatted as follows:

- 100636 • If **-A** is not specified, <library/object name> shall be an empty string.
- 100637 • If **-A** is specified and the corresponding *file* operand does not name a library:  
 100638 "%s: ", <file>
- 100639 • If **-A** is specified and the corresponding *file* operand names a library. In this case,  
 100640 <object file> shall name the object file in the library containing the symbol being described:  
 100641 "%s[%s]: ", <file>, <object file>

100642 If **-A** is not specified, then if more than one *file* operand is specified or if only one *file* operand is  
 100643 specified and it names a library, *nm* shall write a line identifying the object containing the  
 100644 following symbols before the lines containing those symbols, in the form:

- 100645 • If the corresponding *file* operand does not name a library:  
 100646 "%s:\n", <file>
- 100647 • If the corresponding *file* operand names a library; in this case, <object file> shall be the  
 100648 name of the file in the library containing the following symbols:  
 100649 "%s[%s]:\n", <file>, <object file>

100650 If **-P** is specified, but **-t** is not, the format shall be as if **-t x** had been specified.

100651 **STDERR**

100652 The standard error shall be used only for diagnostic messages.

100653 **OUTPUT FILES**

100654 None.

100655 **EXTENDED DESCRIPTION**

100656 None.

100657 **EXIT STATUS**

100658 The following exit values shall be returned:

100659 0 Successful completion.

100660 >0 An error occurred.

100661 **CONSEQUENCES OF ERRORS**

100662 Default.

**100663 APPLICATION USAGE**

100664 Mechanisms for dynamic linking make this utility less meaningful when applied to an  
100665 executable file because a dynamically linked executable may omit numerous library routines  
100666 that would be found in a statically linked executable.

**100667 EXAMPLES**

100668 None.

**100669 RATIONALE**

100670 Historical implementations of *nm* have used different bases for numeric output and supplied  
100671 different default types of symbols that were reported. The *-t format* option, similar to that used  
100672 in *od* and *strings*, can be used to specify the numeric base; *-g* and *-u* can be used to restrict the  
100673 amount of output or the types of symbols included in the output.

100674 The compromise of using *-t format* versus using *-d*, *-o*, and other similar options was necessary  
100675 because of differences in the meaning of *-o* between implementations. The *-o* option from BSD  
100676 has been provided here as *-A* to avoid confusion with the *-o* from System V (which has been  
100677 provided here as *-t* and as *-o* on XSI-conformant systems).

100678 The option list was significantly reduced from that provided by historical implementations.

100679 The *nm* description is a subset of both the System V and BSD *nm* utilities with no specified  
100680 default output.

100681 It was recognized that mechanisms for dynamic linking make this utility less meaningful when  
100682 applied to an executable file (because a dynamically linked executable file may omit numerous  
100683 library routines that would be found in a statically linked executable file), but the value of *nm*  
100684 during software development was judged to outweigh other limitations.

100685 The default output format of *nm* is not specified because of differences in historical  
100686 implementations. The *-P* option was added to allow some type of portable output format. After  
100687 a comparison of the different formats used in SunOS, BSD, SVR3, and SVR4, it was decided to  
100688 create one that did not match the current format of any of these four systems. The format  
100689 devised is easy to parse by humans, easy to parse in shell scripts, and does not need to vary  
100690 depending on locale (because no English descriptions are included). All of the systems currently  
100691 have the information available to use this format.

100692 The format given in *nm* STDOUT uses <space> characters between the fields, which may be any  
100693 number of <blank> characters required to align the columns. The single-character types were  
100694 selected to match historical practice, and the requirement that implementation additions also be  
100695 single characters made parsing the information easier for shell scripts.

**100696 FUTURE DIRECTIONS**

100697 None.

**100698 SEE ALSO**

100699 *ar*, *c99*

100700 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

**100701 CHANGE HISTORY**

100702 First released in Issue 2.

**100703 Issue 6**

100704 This utility is marked as supported when both the User Portability Utilities option and the  
100705 Software Development Utilities option are supported.

100706 **Issue 7**

100707 The *nm* utility is removed from the User Portability Utilities option. User Portability Utilities is  
100708 now an option for interactive utilities.

100709 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

100710 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0125 [263] and XCU/TC1-2008/0126  
100711 [263] are applied.



100712 **NAME**

100713       nohup — invoke a utility immune to hangups

100714 **SYNOPSIS**100715       nohup *utility* [*argument...*]100716 **DESCRIPTION**

100717       The *nohup* utility shall invoke the utility named by the *utility* operand with arguments supplied  
 100718       as the *argument* operands. At the time the named *utility* is invoked, the SIGHUP signal shall be  
 100719       set to be ignored.

100720       If standard input is associated with a terminal, the *nohup* utility may redirect standard input  
 100721       from an unspecified file.

100722       If the standard output is a terminal, all output written by the named *utility* to its standard output  
 100723       shall be appended to the end of the file **nohup.out** in the current directory. If **nohup.out** cannot  
 100724       be created or opened for appending, the output shall be appended to the end of the file  
 100725       **nohup.out** in the directory specified by the *HOME* environment variable. If neither file can be  
 100726       created or opened for appending, *utility* shall not be invoked. If a file is created, the file's  
 100727       permission bits shall be set to S\_IRUSR | S\_IWUSR.

100728       If standard error is a terminal and standard output is open but is not a terminal, all output  
 100729       written by the named utility to its standard error shall be redirected to the same open file  
 100730       description as the standard output. If standard error is a terminal and standard output either is a  
 100731       terminal or is closed, the same output shall instead be appended to the end of the **nohup.out** file  
 100732       as described above.

100733 **OPTIONS**

100734       None.

100735 **OPERANDS**

100736       The following operands shall be supported:

100737       *utility*       The name of a utility that is to be invoked. If the *utility* operand names any of the  
 100738       special built-in utilities in [Section 2.14](#) (on page 2374), the results are undefined.

100739       *argument*   Any string to be supplied as an argument when invoking the utility named by the  
 100740       *utility* operand.

100741 **STDIN**

100742       Not used.

100743 **INPUT FILES**

100744       None.

100745 **ENVIRONMENT VARIABLES**100746       The following environment variables shall affect the execution of *nohup*:

100747       *HOME*       Determine the pathname of the user's home directory: if the output file **nohup.out**  
 100748       cannot be created in the current directory, the *nohup* utility shall use the directory  
 100749       named by *HOME* to create the file.

100750       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 100751       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 100752       variables used to determine the values of locale categories.)

100753       *LC\_ALL*      If set to a non-empty string value, override the values of all the other  
 100754       internationalization variables.

|        |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------|-------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 100755 | <i>LC_CTYPE</i>               | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 100756 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100757 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100758 | <i>LC_MESSAGES</i>            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100759 |                               | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 100760 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100761 | XSI <i>NLSPATH</i>            | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 100762 | <i>PATH</i>                   | Determine the search path that is used to locate the utility to be invoked. See XBD Chapter 8 (on page 173).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 100763 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100764 | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100765 |                               | The <i>nohup</i> utility shall take the standard action for all signals except that <i>SIGHUP</i> shall be ignored.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 100766 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100767 | <b>STDOUT</b>                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100768 |                               | If the standard output is not a terminal, the standard output of <i>nohup</i> shall be the standard output generated by the execution of the <i>utility</i> specified by the operands. Otherwise, nothing shall be written to the standard output.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 100769 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100770 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100771 | <b>STDERR</b>                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100772 |                               | If the standard output is a terminal, a message shall be written to the standard error, indicating the name of the file to which the output is being appended. The name of the file shall be either <b>nohup.out</b> or <b>\$HOME/nohup.out</b> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 100773 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100774 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100775 | <b>OUTPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100776 |                               | Output written by the named utility is appended to the file <b>nohup.out</b> (or <b>\$HOME/nohup.out</b> ), if the conditions hold as described in the DESCRIPTION.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 100777 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100778 | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100779 |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 100780 | <b>EXIT STATUS</b>            |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100781 |                               | The following exit values shall be returned:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 100782 | 126                           | The utility specified by <i>utility</i> was found but could not be invoked.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 100783 | 127                           | An error occurred in the <i>nohup</i> utility or the utility specified by <i>utility</i> could not be found.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 100784 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100785 |                               | Otherwise, the exit status of <i>nohup</i> shall be that of the utility specified by the <i>utility</i> operand.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 100786 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100787 |                               | Default.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 100788 | <b>APPLICATION USAGE</b>      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100789 |                               | The <i>command</i> , <i>env</i> , <i>nice</i> , <i>nohup</i> , <i>time</i> , and <i>xargs</i> utilities have been specified to use exit code 127 if an error occurs so that applications can distinguish “failure to find a utility” from “invoked utility exited with an error indication”. The value 127 was chosen because it is not commonly used for other meanings; most utilities use small values for “normal error conditions” and the values above 128 can be confused with termination due to receipt of a signal. The value 126 was chosen in a similar manner to indicate that the utility could be found, but not invoked. Some scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to <i>exec</i> the utility fail with [ENOENT], and uses 126 when any attempt to <i>exec</i> the utility fails for any other reason. |
| 100790 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100791 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100792 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100793 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100794 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100795 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100796 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100797 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 100798 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

**100799 EXAMPLES**

100800 It is frequently desirable to apply *nohup* to pipelines or lists of commands. This can be done by  
100801 placing pipelines and command lists in a single file; this file can then be invoked as a utility, and  
100802 the *nohup* applies to everything in the file.

100803 Alternatively, the following command can be used to apply *nohup* to a complex command:

100804 `nohup sh -c 'complex-command-line' </dev/null`

**100805 RATIONALE**

100806 The 4.3 BSD version ignores SIGTERM and SIGHUP, and if *./nohup.out* cannot be used, it fails  
100807 instead of trying to use *\$HOME/nohup.out*.

100808 The *cs*h utility has a built-in version of *nohup* that acts differently from the *nohup* defined in this  
100809 volume of POSIX.1-2008.

100810 The term *utility* is used, rather than *command*, to highlight the fact that shell compound  
100811 commands, pipelines, special built-ins, and so on, cannot be used directly. However, *utility*  
100812 includes user application programs and shell scripts, not just the standard utilities.

100813 Historical versions of the *nohup* utility use default file creation semantics. Some more recent  
100814 versions use the permissions specified here as an added security precaution.

100815 Some historical implementations ignore SIGQUIT in addition to SIGHUP; others ignore  
100816 SIGTERM. An early proposal allowed, but did not require, SIGQUIT to be ignored. Several  
100817 reviewers objected that *nohup* should only modify the handling of SIGHUP as required by this  
100818 volume of POSIX.1-2008.

100819 Historical versions of *nohup* did not affect standard input, but that causes problems in the  
100820 common scenario where the user logs into a system, types the command:

100821 `nohup make &`

100822 at the prompt, and then logs out. If standard input is not affected by *nohup*, the login session  
100823 may not terminate for quite some time, since standard input remains open until *make* exits. To  
100824 avoid this problem, POSIX.1-2008 allows implementations to redirect standard input if it is a  
100825 terminal. Since the behavior is implementation-defined, portable applications that may run into  
100826 the problem should redirect standard input themselves. For example, instead of:

100827 `nohup make &`

100828 an application can invoke:

100829 `nohup make </dev/null &`

**100830 FUTURE DIRECTIONS**

100831 None.

**100832 SEE ALSO**

100833 [Chapter 2](#) (on page 2335), *sh*

100834 [XBD Chapter 8](#) (on page 173)

100835 [XSH \*signal\(\)\*](#)

**100836 CHANGE HISTORY**

100837 First released in Issue 2.

100838 **Issue 7**  
100839

Austin Group Interpretations 1003.1-2001 #104, #105, and #106 are applied.

100840 **NAME**100841 `od` — dump files in various formats100842 **SYNOPSIS**

100843 `od [-v] [-A address_base] [-j skip] [-N count] [-t type_string]...`  
 100844 `[file...]`

100845 XSI `od [-bcdosx] [file] [[+]offset[.][b]]`100846 **DESCRIPTION**

100847 The *od* utility shall write the contents of its input files to standard output in a user-specified  
 100848 format.

100849 **OPTIONS**

100850 The *od* utility shall conform to XBD [Section 12.2](#) (on page 216), except that the order of  
 100851 XSI presentation of the `-t` options and the `-bcdosx` options is significant.

100852 The following options shall be supported:

100853 `-A address_base`

100854 Specify the input offset base. See the EXTENDED DESCRIPTION section. The  
 100855 application shall ensure that the *address\_base* option-argument is a character. The  
 100856 characters 'd', 'o', and 'x' specify that the offset base shall be written in  
 100857 decimal, octal, or hexadecimal, respectively. The character 'n' specifies that the  
 100858 offset shall not be written.

100859 XSI `-b` Interpret bytes in octal. This shall be equivalent to `-t o1`.

100860 XSI `-c` Interpret bytes as characters specified by the current setting of the *LC\_CTYPE*  
 100861 category. Certain non-graphic characters appear as C escapes: "NUL=\0",  
 100862 "BS=\b", "FF=\f", "NL=\n", "CR=\r", "HT=\t"; others appear as 3-digit octal  
 100863 numbers.

100864 XSI `-d` Interpret *words* (two-byte units) in unsigned decimal. This shall be equivalent to  
 100865 `-t u2`.

100866 `-j skip` Jump over *skip* bytes from the beginning of the input. The *od* utility shall read or  
 100867 seek past the first *skip* bytes in the concatenated input files. If the combined input is  
 100868 not at least *skip* bytes long, the *od* utility shall write a diagnostic message to  
 100869 standard error and exit with a non-zero exit status.

100870 By default, the *skip* option-argument shall be interpreted as a decimal number.  
 100871 With a leading 0x or 0X, the offset shall be interpreted as a hexadecimal number;  
 100872 otherwise, with a leading '0', the offset shall be interpreted as an octal number.  
 100873 Appending the character 'b', 'k', or 'm' to offset shall cause it to be interpreted  
 100874 as a multiple of 512, 1024, or 1048576 bytes, respectively. If the *skip* number is  
 100875 hexadecimal, any appended 'b' shall be considered to be the final hexadecimal  
 100876 digit.

100877 `-N count` Format no more than *count* bytes of input. By default, *count* shall be interpreted as  
 100878 a decimal number. With a leading 0x or 0X, *count* shall be interpreted as a  
 100879 hexadecimal number; otherwise, with a leading '0', it shall be interpreted as an  
 100880 octal number. If *count* bytes of input (after successfully skipping, if `-j skip` is  
 100881 specified) are not available, it shall not be considered an error; the *od* utility shall  
 100882 format the input that is available.

|        |                 |                                      |                                                                                                                             |
|--------|-----------------|--------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| 100883 | XSI             | <b>-o</b>                            | Interpret <i>words</i> (two-byte units) in octal. This shall be equivalent to <b>-t o2</b> .                                |
| 100884 | XSI             | <b>-s</b>                            | Interpret <i>words</i> (two-byte units) in signed decimal. This shall be equivalent to                                      |
| 100885 |                 | <b>-t d2</b> .                       |                                                                                                                             |
| 100886 |                 | <b>-t</b> <i>type_string</i>         |                                                                                                                             |
| 100887 |                 |                                      | Specify one or more output types. See the EXTENDED DESCRIPTION section. The                                                 |
| 100888 |                 |                                      | application shall ensure that the <i>type_string</i> option-argument is a string specifying                                 |
| 100889 |                 |                                      | the types to be used when writing the input data. The string shall consist of the                                           |
| 100890 |                 |                                      | type specification characters a, c, d, f, o, u, and x, specifying named character,                                          |
| 100891 |                 |                                      | character, signed decimal, floating point, octal, unsigned decimal, and                                                     |
| 100892 |                 |                                      | hexadecimal, respectively. The type specification characters d, f, o, u, and x can be                                       |
| 100893 |                 |                                      | followed by an optional unsigned decimal integer that specifies the number of                                               |
| 100894 |                 |                                      | bytes to be transformed by each instance of the output type. The type specification                                         |
| 100895 |                 |                                      | character f can be followed by an optional F, D, or L indicating that the conversion                                        |
| 100896 |                 |                                      | should be applied to an item of type <b>float</b> , <b>double</b> , or <b>long double</b> , respectively.                   |
| 100897 |                 |                                      | The type specification characters d, o, u, and x can be followed by an optional C, S,                                       |
| 100898 |                 |                                      | I, or L indicating that the conversion should be applied to an item of type <b>char</b> ,                                   |
| 100899 |                 |                                      | <b>short</b> , <b>int</b> , or <b>long</b> , respectively. Multiple types can be concatenated within the                    |
| 100900 |                 |                                      | same <i>type_string</i> and multiple <b>-t</b> options can be specified. Output lines shall be                              |
| 100901 |                 |                                      | written for each type specified in the order in which the type specification                                                |
| 100902 |                 |                                      | characters are specified.                                                                                                   |
| 100903 |                 | <b>-v</b>                            | Write all input data. Without the <b>-v</b> option, any number of groups of output lines,                                   |
| 100904 |                 |                                      | which would be identical to the immediately preceding group of output lines                                                 |
| 100905 |                 |                                      | (except for the byte offsets), shall be replaced with a line containing only an                                             |
| 100906 |                 |                                      | <asterisk> ( ' * ' ).                                                                                                       |
| 100907 | XSI             | <b>-x</b>                            | Interpret <i>words</i> (two-byte units) in hexadecimal. This shall be equivalent to <b>-t x2</b> .                          |
| 100908 | XSI             |                                      | Multiple types can be specified by using multiple <b>-bcdostx</b> options. Output lines are written for                     |
| 100909 |                 |                                      | each type specified in the order in which the types are specified.                                                          |
| 100910 | <b>OPERANDS</b> |                                      |                                                                                                                             |
| 100911 |                 |                                      | The following operands shall be supported:                                                                                  |
| 100912 | <i>file</i>     |                                      | A pathname of a file to be read. If no <i>file</i> operands are specified, the standard input                               |
| 100913 |                 |                                      | shall be used.                                                                                                              |
| 100914 |                 |                                      | If there are no more than two operands, none of the <b>-A</b> , <b>-j</b> , <b>-N</b> , <b>-t</b> , or <b>-v</b> options is |
| 100915 |                 |                                      | specified, and either of the following is true: the first character of the last operand                                     |
| 100916 |                 |                                      | is a <plus-sign> ( ' + ' ), or there are two operands and the first character of the last                                   |
| 100917 | XSI             |                                      | operand is numeric; the last operand shall be interpreted as an offset operand on                                           |
| 100918 |                 |                                      | XSI-conformant systems. Under these conditions, the results are unspecified on                                              |
| 100919 |                 |                                      | systems that are not XSI-conformant systems.                                                                                |
| 100920 | XSI             | <b>[+]<i>offset</i>[.][<i>b</i>]</b> | The <i>offset</i> operand specifies the offset in the file where dumping is to commence.                                    |
| 100921 |                 |                                      | This operand is normally interpreted as octal bytes. If ' . ' is appended, the offset                                       |
| 100922 |                 |                                      | shall be interpreted in decimal. If ' b ' is appended, the offset shall be interpreted                                      |
| 100923 |                 |                                      | in units of 512 bytes.                                                                                                      |
| 100924 | <b>STDIN</b>    |                                      |                                                                                                                             |
| 100925 |                 |                                      | The standard input shall be used if no <i>file</i> operands are specified, and shall be used if a <i>file</i>               |
| 100926 |                 |                                      | operand is ' - ' and the implementation treats the ' - ' as meaning standard input. Otherwise,                              |
| 100927 |                 |                                      | the standard input shall not be used. See the INPUT FILES section.                                                          |

100928 **INPUT FILES**

100929 The input files can be any file type.

100930 **ENVIRONMENT VARIABLES**100931 The following environment variables shall affect the execution of *od*:

100932 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 100933 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 100934 variables used to determine the values of locale categories.)

100935 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 100936 internationalization variables.

100937 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 100938 characters (for example, single-byte as opposed to multi-byte characters in  
 100939 arguments and input files).

100940 **LC\_MESSAGES**

100941 Determine the locale that should be used to affect the format and contents of  
 100942 diagnostic messages written to standard error.

100943 **LC\_NUMERIC**

100944 Determine the locale for selecting the radix character used when writing floating-  
 100945 point formatted output.

100946 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

100947 **ASYNCHRONOUS EVENTS**

100948 Default.

100949 **STDOUT**

100950 See the EXTENDED DESCRIPTION section.

100951 **STDERR**

100952 The standard error shall be used only for diagnostic messages.

100953 **OUTPUT FILES**

100954 None.

100955 **EXTENDED DESCRIPTION**

100956 The *od* utility shall copy sequentially each input file to standard output, transforming the input  
 100957 XSI data according to the output types specified by the **-t** option or the **-bcdosx** options. If no  
 100958 output type is specified, the default output shall be as if **-t oS** had been specified.

100959 The number of bytes transformed by the output type specifier *c* may be variable depending on  
 100960 the **LC\_CTYPE** category.

100961 The default number of bytes transformed by output type specifiers *d*, *f*, *o*, *u*, and *x* corresponds  
 100962 to the various C-language types as follows. If the *c99* compiler is present on the system, these  
 100963 specifiers shall correspond to the sizes used by default in that compiler. Otherwise, these sizes  
 100964 may vary among systems that conform to POSIX.1-2008.

- 100965 • For the type specifier characters *d*, *o*, *u*, and *x*, the default number of bytes shall
- 100966 correspond to the size of the underlying implementation's basic integer type. For these
- 100967 specifier characters, the implementation shall support values of the optional number of
- 100968 bytes to be converted corresponding to the number of bytes in the C-language types **char**,
- 100969 **short**, **int**, and **long**. These numbers can also be specified by an application as the
- 100970 characters '**C**', '**S**', '**I**', and '**L**', respectively. The implementation shall also support
- 100971 the values 1, 2, 4, and 8, even if it provides no C-Language types of those sizes. The



implementation shall support the decimal value corresponding to the C-language type **long long**. The byte order used when interpreting numeric values is implementation-defined, but shall correspond to the order in which a constant of the corresponding type is stored in memory on the system.

- For the type specifier character *f*, the default number of bytes shall correspond to the number of bytes in the underlying implementation's basic double precision floating-point data type. The implementation shall support values of the optional number of bytes to be converted corresponding to the number of bytes in the C-language types **float**, **double**, and **long double**. These numbers can also be specified by an application as the characters 'F', 'D', and 'L', respectively.

The type specifier character *a* specifies that bytes shall be interpreted as named characters from the International Reference Version (IRV) of the ISO/IEC 646:1991 standard. Only the least significant seven bits of each byte shall be used for this type specification. Bytes with the values listed in the following table shall be written using the corresponding names for those characters.

**Table 4-13** Named Characters in *od*

| Value | Name | Value | Name | Value | Name      | Value | Name |
|-------|------|-------|------|-------|-----------|-------|------|
| \000  | nul  | \001  | soh  | \002  | stx       | \003  | etx  |
| \004  | eot  | \005  | enq  | \006  | ack       | \007  | bel  |
| \010  | bs   | \011  | ht   | \012  | lf or nl* | \013  | vt   |
| \014  | ff   | \015  | cr   | \016  | so        | \017  | si   |
| \020  | dle  | \021  | dc1  | \022  | dc2       | \023  | dc3  |
| \024  | dc4  | \025  | nak  | \026  | syn       | \027  | etb  |
| \030  | can  | \031  | em   | \032  | sub       | \033  | esc  |
| \034  | fs   | \035  | gs   | \036  | rs        | \037  | us   |
| \040  | sp   | \177  | del  |       |           |       |      |

**Note:** The "\012" value may be written either as **lf** or **nl**.

The type specifier character *c* specifies that bytes shall be interpreted as characters specified by the current setting of the *LC\_CTYPE* locale category. Characters listed in the table in XBD Chapter 5 (on page 121) ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v') shall be written as the corresponding escape sequences, except that <backslash> shall be written as a single <backslash> and a NUL shall be written as '\0'. Other non-printable characters shall be written as one three-digit octal number for each byte in the character. Printable multi-byte characters shall be written in the area corresponding to the first byte of the character; the two-character sequence "\*\*\*" shall be written in the area corresponding to each remaining byte in the character, as an indication that the character is continued. When either the **-j skip** or **-N count** option is specified along with the *c* type specifier, and this results in an attempt to start or finish in the middle of a multi-byte character, the result is implementation-defined.

The input data shall be manipulated in blocks, where a block is defined as a multiple of the least common multiple of the number of bytes transformed by the specified output types. If the least common multiple is greater than 16, the results are unspecified. Each input block shall be written as transformed by each output type, one per written line, in the order that the output types were specified. If the input block size is larger than the number of bytes transformed by the output type, the output type shall sequentially transform the parts of the input block, and the output from each of the transformations shall be separated by one or more <blank> characters.

If, as a result of the specification of the **-N** option or end-of-file being reached on the last input file, input data only partially satisfies an output type, the input shall be extended sufficiently



101019 with null bytes to write the last byte of the input.

101020 Unless **-A n** is specified, the first output line produced for each input block shall be preceded by  
 101021 the input offset, cumulative across input files, of the next byte to be written. The format of the  
 101022 input offset is unspecified; however, it shall not contain any <blank> characters, shall start at the  
 101023 first character of the output line, and shall be followed by one or more <blank> characters. In  
 101024 addition, the offset of the byte following the last byte written shall be written after all the input  
 101025 data has been processed, but shall not be followed by any <blank> characters.

101026 If no **-A** option is specified, the input offset base is unspecified.

## 101027 EXIT STATUS

101028 The following exit values shall be returned:

101029 0 All input files were processed successfully.

101030 >0 An error occurred.

## 101031 CONSEQUENCES OF ERRORS

101032 Default.

## 101033 APPLICATION USAGE

101034 XSI-conformant applications are warned not to use filenames starting with '+' or a first  
 101035 operand starting with a numeric character so that the old functionality can be maintained by  
 101036 implementations, unless they specify one of the **-A**, **-j**, or **-N** options. To guarantee that one of  
 101037 these filenames is always interpreted as a filename, an application could always specify the  
 101038 address base format with the **-A** option.

## 101039 EXAMPLES

101040 If a file containing 128 bytes with decimal values zero to 127, in increasing order, is supplied as  
 101041 standard input to the command:

101042 `od -A d -t a`

101043 on an implementation using an input block size of 16 bytes, the standard output, independent of  
 101044 the current locale setting, would be similar to:

```

101045 0000000 nul soh stx etx eot enq ack bel bs ht nl vt ff cr so si
101046 0000016 dle dc1 dc2 dc3 dc4 nak syn etb can em sub esc fs gs rs us
101047 0000032 sp ! " # $ % & ' ( ) * + , - . /
101048 0000048 0 1 2 3 4 5 6 7 8 9 : ; < = > ?
101049 0000064 @ A B C D E F G H I J K L M N O
101050 0000080 P Q R S T U V W X Y Z [ \ ] ^ _
101051 0000096 ` a b c d e f g h i j k l m n o
101052 0000112 p q r s t u v w x y z { | } ~ del
101053 0000128
  
```

101054 Note that this volume of POSIX.1-2008 allows **nl** or **lf** to be used as the name for the  
 101055 ISO/IEC 646:1991 standard IRV character with decimal value 10. The IRV names this character  
 101056 **lf** (line feed), but traditional implementations have referred to this character as newline (**nl**) and  
 101057 the POSIX locale character set symbolic name for the corresponding character is a <newline>.

101058 The command:

101059 `od -A o -t o2x2x -N 18`

101060 on a system with 32-bit words and an implementation using an input block size of 16 bytes  
 101061 could write 18 bytes in approximately the following format:

101062 `0000000 032056 031440 041123 042040 052516 044530 020043 031464`

```

101063          342e   3320   4253   4420   554e   4958   2023   3334
101064          342e3320          42534420          554e4958          20233334
101065 0000020 032472
101066          353a
101067          353a0000
101068 0000022

```

The command:

```
od -A d -t f -t o4 -t x4 -N 24 -j 0x15
```

on a system with 64-bit doubles (for example, IEEE Std 754-1985 double precision floating-point format) would skip 21 bytes of input data and then write 24 bytes in approximately the following format:

```

101074 0000000 1.000000000000000e+00 1.573500000000000e+01
101075 07774000000 00000000000 10013674121 35341217270
101076 3ff00000 00000000 402f3851 eb851eb8
101077 0000016 1.406682300000000e+02
101078 10030312542 04370303230
101079 40619562 23e18698
101080 0000024

```

## RATIONALE

The *od* utility went through several names in early proposals, including *hd*, *xd*, and most recently *hexdump*. There were several objections to all of these based on the following reasons:

- The *hd* and *xd* names conflicted with historical utilities that behaved differently.
- The *hexdump* description was much more complex than needed for a simple dump utility.
- The *od* utility has been available on all historical implementations and there was no need to create a new name for a utility so similar to the historical *od* utility.

The original reasons for not standardizing historical *od* were also fairly widespread. Those reasons are given below along with rationale explaining why the standard developers believe that this version does not suffer from the indicated problem:

- The BSD and System V versions of *od* have diverged, and the intersection of features provided by both does not meet the needs of the user community. In fact, the System V version only provides a mechanism for dumping octal bytes and **shorts**, signed and unsigned decimal **shorts**, hexadecimal **shorts**, and ASCII characters. BSD added the ability to dump **floats**, **doubles**, named ASCII characters, and octal, signed decimal, unsigned decimal, and hexadecimal **longs**. The version presented here provides more normalized forms for dumping bytes, **shorts**, **ints**, and **longs** in octal, signed decimal, unsigned decimal, and hexadecimal; **float**, **double**, and **long double**; and named ASCII as well as current locale characters.
- It would not be possible to come up with a compatible superset of the BSD and System V flags that met the requirements of the standard developers. The historical default *od* output is the specified default output of this utility. None of the option letters chosen for this version of *od* conflict with any of the options to historical versions of *od*.
- On systems with different sizes for **short**, **int**, and **long**, there was no way to ask for dumps of **ints**, even in the BSD version. Because of the way options are named, the name space could not be extended to solve these problems. This is why the **-t** option was added (with type specifiers more closely matched to the *printf()* formats used in the rest of this volume of POSIX.1-2008) and the optional field sizes were added to the **d**, **f**, **o**, **u**, and **x** type

specifiers. It is also one of the reasons why the historical practice was not mandated as a required obsolescent form of *od*. (Although the old versions of *od* are not listed as an obsolescent form, implementations are urged to continue to recognize the older forms for several more years.) The *a*, *c*, *f*, *o*, and *x* types match the meaning of the corresponding format characters in the historical implementations of *od* except for the default sizes of the fields converted. The *d* format is signed in this volume of POSIX.1-2008 to match the *printf()* notation. (Historical versions of *od* used *d* as a synonym for *u* in this version. The System V implementation uses *s* for signed decimal; BSD uses *i* for signed decimal and *s* for null-terminated strings.) Other than *d* and *u*, all of the type specifiers match format characters in the historical BSD version of *od*.

The sizes of the C-language types **char**, **short**, **int**, **long**, **float**, **double**, and **long double** are used even though it is recognized that there may be zero or more than one compiler for the C language on an implementation and that they may use different sizes for some of these types. (For example, one compiler might use 2 bytes **shorts**, 2 bytes **ints**, and 4 bytes **longs**, while another compiler (or an option to the same compiler) uses 2 bytes **shorts**, 4 bytes **ints**, and 4 bytes **longs**.) Nonetheless, there has to be a basic size known by the implementation for these types, corresponding to the values reported by invocations of the *getconf* utility when called with *system\_var* operands {UCHAR\_MAX}, {USHORT\_MAX}, {UINT\_MAX}, and {ULONG\_MAX} for the types **char**, **short**, **int**, and **long**, respectively. There are similar constants required by the ISO C standard, but not required by the System Interfaces volume of POSIX.1-2008 or this volume of POSIX.1-2008. They are {FLT\_MANT\_DIG}, {DBL\_MANT\_DIG}, and {LDBL\_MANT\_DIG} for the types **float**, **double**, and **long double**, respectively. If the optional *c99* utility is provided by the implementation and used as specified by this volume of POSIX.1-2008, these are the sizes that would be provided. If an option is used that specifies different sizes for these types, there is no guarantee that the *od* utility is able to interpret binary data output by such a program correctly.

This volume of POSIX.1-2008 requires that the numeric values of these lengths be recognized by the *od* utility and that symbolic forms also be recognized. Thus, a conforming application can always look at an array of **unsigned long** data elements using *od -t uL*.

- The method of specifying the format for the address field based on specifying a starting offset in a file unnecessarily tied the two together. The **-A** option now specifies the address base and the **-S** option specifies a starting offset.
- It would be difficult to break the dependence on US ASCII to achieve an internationalized utility. It does not seem to be any harder for *od* to dump characters in the current locale than it is for the *ed* or *sed* **l** commands. The *c* type specifier does this without difficulty and is completely compatible with the historical implementations of the *c* format character when the current locale uses a superset of the ISO/IEC 646:1991 standard as a codeset. The *a* type specifier (from the BSD *a* format character) was left as a portable means to dump ASCII (or more correctly ISO/IEC 646:1991 standard (IRV)) so that headers produced by *pax* could be deciphered even on systems that do not use the ISO/IEC 646:1991 standard as a subset of their base codeset.

The use of **"\*\*"** as an indication of continuation of a multi-byte character in *c* specifier output was chosen based on seeing an implementation that uses this method. The continuation bytes have to be marked in a way that is not ambiguous with another single-byte or multi-byte character.

An early proposal used **-S** and **-n**, respectively, for the **-j** and **-N** options eventually selected. These were changed to avoid conflicts with historical implementations.

101158 The original standard specified **-t o2** as the default when no output type was given. This was  
101159 changed to **-t oS** (the length of a **short**) to accommodate a supercomputer implementation that  
101160 historically used 64 bits as its default (and that defined shorts as 64 bits). This change should not  
101161 affect conforming applications. The requirement to support lengths of 1, 2, and 4 was added at  
101162 the same time to address an historical implementation that had no two-byte data types in its C  
101163 compiler.

101164 The use of a basic integer data type is intended to allow the implementation to choose a word  
101165 size commonly used by applications on that architecture.

101166 Earlier versions of this standard allowed for implementations with bytes other than eight bits,  
101167 but this has been modified in this version.

#### 101168 **FUTURE DIRECTIONS**

101169 All option and operand interfaces marked XSI may be removed in a future version.

#### 101170 **SEE ALSO**

101171 *c99, sed*

101172 XBD [Chapter 5](#) (on page 121), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

#### 101173 **CHANGE HISTORY**

101174 First released in Issue 2.

#### 101175 **Issue 5**

101176 In the description of the **-c** option, the phrase “This is equivalent to **-t c.**” is deleted.

101177 The FUTURE DIRECTIONS section is modified.

#### 101178 **Issue 6**

101179 The *od* utility is changed to remove the assumption that **short** was a two-byte entity, as per the  
101180 revisions in the IEEE P1003.2b draft standard.

101181 The normative text is reworded to avoid use of the term “must” for application requirements.

101182 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/33 is applied, correcting the examples  
101183 which used an undefined **-n** option, which should have been **-N**.

101184 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/19 is applied, removing text describing  
101185 behavior on systems with bytes consisting of more than eight bits.

#### 101186 **Issue 7**

101187 Austin Group Interpretation 1003.1-2001 #092 is applied.

101188 SD5-XCU-ERN-37 is applied, updating the OPERANDS section.

101189 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

101190 **NAME**101191 `paste` — merge corresponding or subsequent lines of files101192 **SYNOPSIS**101193 `paste [-s] [-d list] file...`101194 **DESCRIPTION**101195 The *paste* utility shall concatenate the corresponding lines of the given input files, and write the  
101196 resulting lines to standard output.101197 The default operation of *paste* shall concatenate the corresponding lines of the input files. The  
101198 <newline> of every line except the line from the last input file shall be replaced with a <tab>.101199 If an end-of-file condition is detected on one or more input files, but not all input files, *paste* shall  
101200 behave as though empty lines were read from the files on which end-of-file was detected, unless  
101201 the `-s` option is specified.101202 **OPTIONS**101203 The *paste* utility shall conform to XBD [Section 12.2](#) (on page 216).

101204 The following options shall be supported:

101205 `-d list` Unless a <backslash> character appears in *list*, each character in *list* is an element  
101206 specifying a delimiter character. If a <backslash> character appears in *list*, the  
101207 <backslash> character and one or more characters following it are an element  
101208 specifying a delimiter character as described below. These elements specify one or  
101209 more delimiters to use, instead of the default <tab>, to replace the <newline> of  
101210 the input lines. The elements in *list* shall be used circularly; that is, when the list is  
101211 exhausted the first element from the list is reused. When the `-s` option is specified:

- 101212
- The last <newline> in a file shall not be modified.
  - The delimiter shall be reset to the first element of *list* after each *file* operand is  
101213 processed.

101214 When the `-s` option is not specified:

- 101215
- The <newline> characters in the file specified by the last *file* operand shall  
101216 not be modified.
  - The delimiter shall be reset to the first element of *list* each time a line is  
101217 processed from each file.

101218 If a <backslash> character appears in *list*, it and the character following it shall be  
101219 used to represent the following delimiter characters:101220 `\n` <newline>.101221 `\t` <tab>.101222 `\\` <backslash> character.101223 `\0` Empty string (not a null character). If '`\0`' is immediately followed by the  
101224 character '`x`', the character '`X`', or any character defined by the *LC\_CTYPE*  
101225 **digit** keyword (see XBD [Chapter 7](#), on page 135), the results are unspecified.

101226 If any other characters follow the &lt;backslash&gt;, the results are unspecified.

101227 `-s` Concatenate all of the lines from each input file into one line of output per file, in  
101228 command line order. The <newline> of every line except the last line in each input  
101229 file shall be replaced with a <tab>, unless otherwise specified by the `-d` option. If  
101230 an input file is empty, the output line corresponding to that file shall consist of  
101231 |  
101232 |

101233 only a <newline> character.

#### 101234 OPERANDS

101235 The following operand shall be supported:

101236 *file* A pathname of an input file. If '-' is specified for one or more of the *files*, the  
 101237 standard input shall be used; the standard input shall be read one line at a time,  
 101238 circularly, for each instance of '-'. Implementations shall support pasting of at  
 101239 least 12 *file* operands.

#### 101240 STDIN

101241 The standard input shall be used only if one or more *file* operands is '-'. See the INPUT FILES  
 101242 section.

#### 101243 INPUT FILES

101244 The input files shall be text files, except that line lengths shall be unlimited.

#### 101245 ENVIRONMENT VARIABLES

101246 The following environment variables shall affect the execution of *paste*:

101247 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 101248 (See XBD Section 8.2 (on page 174) the precedence of internationalization variables  
 101249 used to determine the values of locale categories.)

101250 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 101251 internationalization variables.

101252 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 101253 characters (for example, single-byte as opposed to multi-byte characters in  
 101254 arguments and input files).

101255 *LC\_MESSAGES*  
 101256 Determine the locale that should be used to affect the format and contents of  
 101257 diagnostic messages written to standard error.

101258 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

#### 101259 ASYNCHRONOUS EVENTS

101260 Default.

#### 101261 STDOUT

101262 Concatenated lines of input files shall be separated by the <tab> (or other characters under the  
 101263 control of the -d option) and terminated by a <newline>.

#### 101264 STDERR

101265 The standard error shall be used only for diagnostic messages.

#### 101266 OUTPUT FILES

101267 None.

#### 101268 EXTENDED DESCRIPTION

101269 None.

#### 101270 EXIT STATUS

101271 The following exit values shall be returned:

101272 0 Successful completion.

101273 >0 An error occurred.



101274 **CONSEQUENCES OF ERRORS**

101275 If one or more input files cannot be opened when the `-s` option is not specified, a diagnostic  
 101276 message shall be written to standard error, but no output is written to standard output. If the `-s`  
 101277 option is specified, the *paste* utility shall provide the default behavior described in [Section 1.4](#) (on  
 101278 page 2326).

101279 **APPLICATION USAGE**

101280 When the escape sequences of the *list* option-argument are used in a shell script, they must be  
 101281 quoted; otherwise, the shell treats the `<backslash>` as a special character.

101282 Conforming applications should only use the specific `<backslash>`-escaped delimiters presented  
 101283 in this volume of POSIX.1-2008. Historical implementations treat `'\x'`, where `'x'` is not in this  
 101284 list, as `'x'`, but future implementations are free to expand this list to recognize other common  
 101285 escapes similar to those accepted by *printf* and other standard utilities.

101286 Most of the standard utilities work on text files. The *cut* utility can be used to turn files with  
 101287 arbitrary line lengths into a set of text files containing the same data. The *paste* utility can be used  
 101288 to create (or recreate) files with arbitrary line lengths. For example, if *file* contains long lines:

```
101289 cut -b 1-500 -n file > file1
101290 cut -b 501- -n file > file2
```

101291 creates **file1** (a text file) with lines no longer than 500 bytes (plus the `<newline>`) and **file2** that  
 101292 contains the remainder of the data from *file*. Note that **file2** is not a text file if there are lines in  
 101293 *file* that are longer than 500 + `{LINE_MAX}` bytes. The original file can be recreated from **file1**  
 101294 and **file2** using the command:

```
101295 paste -d "\0" file1 file2 > file
```

101296 The commands:

```
101297 paste -d "\0" ...
101298 paste -d " " ...
```

101299 are not necessarily equivalent; the latter is not specified by this volume of POSIX.1-2008 and  
 101300 may result in an error. The construct `'\0'` is used to mean “no separator” because historical  
 101301 versions of *paste* did not follow the syntax guidelines, and the command:

```
101302 paste -d " " ...
```

101303 could not be handled properly by *getopt()*.

101304 **EXAMPLES**

101305 1. Write out a directory in four columns:

```
101306 ls | paste - - - -
```

101307 2. Combine pairs of lines from a file into single lines:

```
101308 paste -s -d "\t\n" file
```

101309 **RATIONALE**

101310 None.

101311 **FUTURE DIRECTIONS**

101312 None.

101313 **SEE ALSO**101314 [Section 1.4](#) (on page 2326), *cut*, *grep*, *pr*101315 XBD [Chapter 7](#) (on page 135), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)101316 **CHANGE HISTORY**

101317 First released in Issue 2.

101318 **Issue 6**

101319 The normative text is reworded to avoid use of the term “must” for application requirements.

101320 **Issue 7**

101321 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



101322 **NAME**

101323 patch — apply changes to files

101324 **SYNOPSIS**

101325 patch [-blNR] [-c|-e|-n|-u] [-d *dir*] [-D *define*] [-i *patchfile*]  
 101326 [-o *outfile*] [-p *num*] [-r *rejectfile*] [*file*]

101327 **DESCRIPTION**

101328 The *patch* utility shall read a source (patch) file containing any of four forms of difference (diff)  
 101329 listings produced by the *diff* utility (normal, copied context, unified context, or in the style of *ed*)  
 101330 and apply those differences to a file. By default, *patch* shall read from the standard input.

101331 The *patch* utility shall attempt to determine the type of the *diff* listing, unless overruled by a *-c*,  
 101332 *-e*, *-n*, or *-u* option.

101333 If the patch file contains more than one patch, *patch* shall attempt to apply each of them as if they  
 101334 came from separate patch files. (In this case, the application shall ensure that the name of the  
 101335 patch file is determinable for each *diff* listing.)

101336 **OPTIONS**

101337 The *patch* utility shall conform to XBD [Section 12.2](#) (on page 216).

101338 The following options shall be supported:

101339 **-b** Save a copy of the original contents of each modified file, before the differences are  
 101340 applied, in a file of the same name with the suffix **.orig** appended to it. If the file  
 101341 already exists, it shall be overwritten; if multiple patches are applied to the same  
 101342 file, the **.orig** file shall be written only for the first patch. When the *-o outfile* option  
 101343 is also specified, *file.orig* shall not be created but, if *outfile* already exists, *outfile.orig*  
 101344 shall be created.

101345 **-c** Interpret the patch file as a copied context difference (the output of the utility *diff*  
 101346 when the *-c* or *-C* options are specified).

101347 **-d dir** Change the current directory to *dir* before processing as described in the  
 101348 EXTENDED DESCRIPTION section.

101349 **-D define** Mark changes with one of the following C preprocessor constructs:

```
101350 #ifdef define
101351 ...
101352 #endif

101353 #ifndef define
101354 ...
101355 #endif
```

101356 optionally combined with the C preprocessor construct **#else**. If the patched file is  
 101357 processed with the C preprocessor, where the macro *define* is defined, the output  
 101358 shall contain the changes from the patch file; otherwise, the output shall not  
 101359 contain the patches specified in the patch file.

101360 **-e** Interpret the patch file as an *ed* script, rather than a *diff* script.

101361 **-i patchfile** Read the patch information from the file named by the pathname *patchfile*, rather  
 101362 than the standard input.

101363 **-l** (The letter ell.) Cause any sequence of <blank> characters in the difference script to  
 101364 match any sequence of <blank> characters in the input file. Other characters shall  
 101365 be matched exactly.

|        |                      |                                                                                                      |
|--------|----------------------|------------------------------------------------------------------------------------------------------|
| 101366 | <b>-n</b>            | Interpret the script as a normal difference.                                                         |
| 101367 | <b>-N</b>            | Ignore patches where the differences have already been applied to the file; by                       |
| 101368 |                      | default, already-applied patches shall be rejected.                                                  |
| 101369 | <b>-o outfile</b>    | Instead of modifying the files (specified by the <i>file</i> operand or the difference               |
| 101370 |                      | listings) directly, write a copy of the file referenced by each patch, with the                      |
| 101371 |                      | appropriate differences applied, to <i>outfile</i> . Multiple patches for a single file shall be     |
| 101372 |                      | applied to the intermediate versions of the file created by any previous patches,                    |
| 101373 |                      | and shall result in multiple, concatenated versions of the file being written to                     |
| 101374 |                      | <i>outfile</i> .                                                                                     |
| 101375 | <b>-p num</b>        | For all pathnames in the patch file that indicate the names of files to be patched,                  |
| 101376 |                      | delete <i>num</i> pathname components from the beginning of each pathname. If the                    |
| 101377 |                      | pathname in the patch file is absolute, any leading <slash> characters shall be                      |
| 101378 |                      | considered the first component (that is, <b>-p 1</b> shall remove the leading <slash>                |
| 101379 |                      | characters). Specifying <b>-p 0</b> shall cause the full pathname to be used. If <b>-p</b> is not    |
| 101380 |                      | specified, only the basename (the final pathname component) shall be used.                           |
| 101381 | <b>-R</b>            | Reverse the sense of the patch script; that is, assume that the difference script was                |
| 101382 |                      | created from the new version to the old version. The <b>-R</b> option cannot be used                 |
| 101383 |                      | with <i>ed</i> scripts. The <i>patch</i> utility shall attempt to reverse each portion of the script |
| 101384 |                      | before applying it. Rejected differences shall be saved in swapped format. If this                   |
| 101385 |                      | option is not specified, and until a portion of the patch file is successfully applied,              |
| 101386 |                      | <i>patch</i> attempts to apply each portion in its reversed sense as well as in its normal           |
| 101387 |                      | sense. If the attempt is successful, the user shall be prompted to determine whether                 |
| 101388 |                      | the <b>-R</b> option should be set.                                                                  |
| 101389 | <b>-r rejectfile</b> | Override the default reject filename. In the default case, the reject file shall have the            |
| 101390 |                      | same name as the output file, with the suffix <b>.rej</b> appended to it; see <a href="#">Patch</a>  |
| 101391 |                      | <a href="#">Application</a> (on page 3048).                                                          |
| 101392 | <b>-u</b>            | Interpret the patch file as a unified context difference (the output of the <i>diff</i> utility      |
| 101393 |                      | when the <b>-u</b> or <b>-U</b> options are specified).                                              |

#### 101394 OPERANDS

101395 The following operand shall be supported:

101396 *file* A pathname of a file to patch.

#### 101397 STDIN

101398 See the INPUT FILES section.

#### 101399 INPUT FILES

101400 Input files shall be text files.

#### 101401 ENVIRONMENT VARIABLES

101402 The following environment variables shall affect the execution of *patch*:

101403 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 101404 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 101405 used to determine the values of locale categories.)

101406 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 101407 internationalization variables.

101408 *LC\_COLLATE*  
 101409 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 101410 character collating elements used in the extended regular expression defined for

|        |                                                                                                                                                  |                                                                                       |
|--------|--------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------|
| 101411 |                                                                                                                                                  | the <b>yesexpr</b> locale keyword in the <i>LC_MESSAGES</i> category.                 |
| 101412 | <i>LC_CTYPE</i>                                                                                                                                  | Determine the locale for the interpretation of sequences of bytes of text data as     |
| 101413 |                                                                                                                                                  | characters (for example, single-byte as opposed to multi-byte characters in           |
| 101414 |                                                                                                                                                  | arguments and input files), and the behavior of character classes used in the         |
| 101415 |                                                                                                                                                  | extended regular expression defined for the <b>yesexpr</b> locale keyword in the      |
| 101416 |                                                                                                                                                  | <i>LC_MESSAGES</i> category.                                                          |
| 101417 | <i>LC_MESSAGES</i>                                                                                                                               |                                                                                       |
| 101418 |                                                                                                                                                  | Determine the locale used to process affirmative responses, and the locale used to    |
| 101419 |                                                                                                                                                  | affect the format and contents of diagnostic messages and prompts written to          |
| 101420 |                                                                                                                                                  | standard error.                                                                       |
| 101421 | XSI <i>NLSPATH</i>                                                                                                                               | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> . |
| 101422 | <i>LC_TIME</i>                                                                                                                                   | Determine the locale for recognizing the format of file timestamps written by the     |
| 101423 |                                                                                                                                                  | <i>diff</i> utility in a context-difference input file.                               |
| 101424 | <b>ASYNCHRONOUS EVENTS</b>                                                                                                                       |                                                                                       |
| 101425 | Default.                                                                                                                                         |                                                                                       |
| 101426 | <b>STDOUT</b>                                                                                                                                    |                                                                                       |
| 101427 | Not used.                                                                                                                                        |                                                                                       |
| 101428 | <b>STDERR</b>                                                                                                                                    |                                                                                       |
| 101429 | The standard error shall be used for diagnostic and informational messages.                                                                      |                                                                                       |
| 101430 | <b>OUTPUT FILES</b>                                                                                                                              |                                                                                       |
| 101431 | The output of the <i>patch</i> utility, the save files ( <b>.orig</b> suffixes), and the reject files ( <b>.rej</b> suffixes) shall              |                                                                                       |
| 101432 | be text files.                                                                                                                                   |                                                                                       |
| 101433 | <b>EXTENDED DESCRIPTION</b>                                                                                                                      |                                                                                       |
| 101434 | A patch file may contain patching instructions for more than one file; filenames shall be                                                        |                                                                                       |
| 101435 | determined as specified in <a href="#">Filename Determination</a> (on page 3048). When the <b>-b</b> option is                                   |                                                                                       |
| 101436 | specified, for each patched file, the original shall be saved in a file of the same name with the                                                |                                                                                       |
| 101437 | suffix <b>.orig</b> appended to it.                                                                                                              |                                                                                       |
| 101438 | For each patched file, a reject file may also be created as noted in <a href="#">Patch Application</a> (on page                                  |                                                                                       |
| 101439 | 3048). In the absence of a <b>-r</b> option, the name of this file shall be formed by appending the suffix                                       |                                                                                       |
| 101440 | <b>.rej</b> to the original filename.                                                                                                            |                                                                                       |
| 101441 | <b>Patch File Format</b>                                                                                                                         |                                                                                       |
| 101442 | The patch file shall contain zero or more lines of header information followed by one or more                                                    |                                                                                       |
| 101443 | patches. Each patch shall contain zero or more lines of filename identification in the format                                                    |                                                                                       |
| 101444 | produced by the <b>-c</b> , <b>-C</b> , <b>-u</b> , or <b>-U</b> options of the <i>diff</i> utility, and one or more sets of <i>diff</i> output, |                                                                                       |
| 101445 | which are customarily called <i>hunks</i> .                                                                                                      |                                                                                       |
| 101446 | The <i>patch</i> utility shall recognize the following expression in the header information:                                                     |                                                                                       |
| 101447 | <b>Index:</b> <i>pathname</i>                                                                                                                    |                                                                                       |
| 101448 | The file to be patched is named <i>pathname</i> .                                                                                                |                                                                                       |
| 101449 | If all lines (including headers) within a patch begin with the same leading sequence of <blank>                                                  |                                                                                       |
| 101450 | characters, the <i>patch</i> utility shall remove this sequence before proceeding. Within each patch, if                                         |                                                                                       |
| 101451 | the type of difference is common context, the <i>patch</i> utility shall recognize the following                                                 |                                                                                       |
| 101452 | expressions:                                                                                                                                     |                                                                                       |

101453 \*\*\* *filename timestamp*  
 101454 The patches arose from *filename*.

101455 --- *filename timestamp*  
 101456 The patches should be applied to *filename*.

101457 If the type of difference is unified context, the *patch* utility shall recognize the following  
 101458 expressions:

101459 --- *filename timestamp*  
 101460 The patches arose from *filename*.

101461 + + + *filename timestamp*  
 101462 The patches should be applied to *filename*.

101463 Each hunk within a patch shall be the *diff* output to change a line range within the original file.  
 101464 The line numbers for successive hunks within a patch shall occur in ascending order.

### 101465 Filename Determination

101466 If no *file* operand is specified, *patch* shall perform the following steps to determine the filename  
 101467 to use:

- 101468 1. If the type of *diff* is context, the *patch* utility shall delete pathname components (as  
 101469 specified by the **-p** option) from the filename on the line beginning with "\*\*\*\*" (if copied  
 101470 context) or "---" (if unified context), then test for the existence of this file relative to the  
 101471 current directory (or the directory specified with the **-d** option). If the file exists, the *patch*  
 101472 utility shall use this filename.
- 101473 2. If the type of *diff* is context, the *patch* utility shall delete the pathname components (as  
 101474 specified by the **-p** option) from the filename on the line beginning with "---" (if copied  
 101475 context) or "+ + +" (if unified context), then test for the existence of this file relative to the  
 101476 current directory (or the directory specified with the **-d** option). If the file exists, the *patch*  
 101477 utility shall use this filename.
- 101478 3. If the header information contains a line beginning with the string **Index:**, the *patch* utility  
 101479 shall delete pathname components (as specified by the **-p** option) from this line, then test  
 101480 for the existence of this file relative to the current directory (or the directory specified  
 101481 with the **-d** option). If the file exists, the *patch* utility shall use this filename.
- 101482 XSI 4. If an **SCCS** directory exists in the current directory, *patch* shall attempt to perform a **get -e**  
 101483 **SCCS/s.filename** command to retrieve an editable version of the file. If the file exists, the  
 101484 *patch* utility shall use this filename.
- 101485 5. The *patch* utility shall write a prompt to standard output and request a filename  
 101486 interactively from the controlling terminal (for example, **/dev/tty**).

### 101487 Patch Application

101488 If the **-c**, **-e**, **-n**, or **-u** option is present, the *patch* utility shall interpret information within each  
 101489 hunk as a copied context difference, an *ed* difference, a normal difference, or a unified context  
 101490 difference, respectively. In the absence of any of these options, the *patch* utility shall determine  
 101491 the type of difference based on the format of information within the hunk.

101492 For each hunk, the *patch* utility shall begin to search for the place to apply the patch at the line  
 101493 number at the beginning of the hunk, plus or minus any offset used in applying the previous  
 101494 hunk. If lines matching the hunk context are not found, *patch* shall scan both forwards and  
 101495 backwards at least 1 000 bytes for a set of lines that match the hunk context.

101496 If no such place is found and it is a context difference, then another scan shall take place,  
101497 ignoring the first and last line of context. If that fails, the first two and last two lines of context  
101498 shall be ignored and another scan shall be made. Implementations may search more extensively  
101499 for installation locations.

101500 If no location can be found, the *patch* utility shall append the hunk to the reject file. A rejected  
101501 hunk that is a copied context difference, an *ed* difference, or a normal difference shall be written  
101502 in copied-context-difference format regardless of the format of the patch file. It is  
101503 implementation-defined whether a rejected hunk that is a unified context difference is written in  
101504 copied-context-difference format or in unified-context-difference format. If the input was a  
101505 normal or *ed*-style difference, the reject file may contain differences with zero lines of context.  
101506 The line numbers on the hunks in the reject file may be different from the line numbers in the  
101507 patch file since they shall reflect the approximate locations for the failed hunks in the new file  
101508 rather than the old one.

101509 If the type of patch is an *ed* diff, the implementation may accomplish the patching by invoking  
101510 the *ed* utility.

#### 101511 EXIT STATUS

101512 The following exit values shall be returned:

- 101513 0 Successful completion.
- 101514 1 One or more lines were written to a reject file.
- 101515 >1 An error occurred.

#### 101516 CONSEQUENCES OF ERRORS

101517 Patches that cannot be correctly placed in the file shall be written to a reject file.

#### 101518 APPLICATION USAGE

101519 The **-R** option does not work with *ed* scripts because there is too little information to reconstruct  
101520 the reverse operation.

101521 The **-p** option makes it possible to customize a patch file to local user directory structures  
101522 without manually editing the patch file. For example, if the filename in the patch file was:

101523 `/curds/whey/src/blurfl/blurfl.c`

101524 Setting **-p 0** gives the entire pathname unmodified; **-p 1** gives:

101525 `curds/whey/src/blurfl/blurfl.c`

101526 without the leading <slash>, **-p 4** gives:

101527 `blurfl/blurfl.c`

101528 and not specifying **-p** at all gives:

101529 `blurfl.c` .

#### 101530 EXAMPLES

101531 None.

#### 101532 RATIONALE

101533 Some of the functionality in historical *patch* implementations was not specified. The following  
101534 documents those features present in historical implementations that have not been specified.

101535 A deleted piece of functionality was the '+' pseudo-option allowing an additional set of options  
101536 and a patch file operand to be given. This was seen as being insufficiently useful to standardize.

101537 In historical implementations, if the string "Prereq:" appeared in the header, the *patch* utility

would search for the corresponding version information (the string specified in the header, delimited by <blank> characters or the beginning or end of a line or the file) anywhere in the original file. This was deleted as too simplistic and insufficiently trustworthy a mechanism to standardize. For example, if:

Prereq: 1.2

were in the header, the presence of a delimited 1.2 anywhere in the file would satisfy the prerequisite.

The following options were dropped from historical implementations of *patch* as insufficiently useful to standardize:

**-b** The **-b** option historically provided a method for changing the name extension of the backup file from the default **.orig**. This option has been modified and retained in this volume of POSIX.1-2008.

**-F** The **-F** option specified the number of lines of a context diff to ignore when searching for a place to install a patch.

**-f** The **-f** option historically caused *patch* not to request additional information from the user.

**-r** The **-r** option historically provided a method of overriding the extension of the reject file from the default **.rej**.

**-s** The **-s** option historically caused *patch* to work silently unless an error occurred.

**-x** The **-x** option historically set internal debugging flags.

In some file system implementations, the saving of a **.orig** file may produce unwanted results. In the case of 12, 13, or 14-character filenames (on file systems supporting 14-character maximum filenames), the **.orig** file overwrites the new file. The reject file may also exceed this filename limit. It was suggested, due to some historical practice, that a <tilde> ('~') suffix be used instead of **.orig** and some other character instead of the **.rej** suffix. This was rejected because it is not obvious to the user which file is which. The suffixes **.orig** and **.rej** are clearer and more understandable.

The **-b** option has the opposite sense in some historical implementations—do not save the **.orig** file. The default case here is not to save the files, making *patch* behave more consistently with the other standard utilities.

The **-w** option in early proposals was changed to **-l** to match historical practice.

The **-N** option was included because without it, a non-interactive application cannot reject previously applied patches. For example, if a user is piping the output of *diff* into the *patch* utility, and the user only wants to patch a file to a newer version non-interactively, the **-N** option is required.

Changes to the **-l** option description were proposed to allow matching across <newline> characters in addition to just <blank> characters. Since this is not historical practice, and since some ambiguities could result, it is suggested that future developments in this area utilize another option letter, such as **-L**.

The **-u** option of GNU *patch* has been added, along with support for unified context formats.

101578 **FUTURE DIRECTIONS**

101579 None.

101580 **SEE ALSO**101581 *diff, ed*101582 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)101583 **CHANGE HISTORY**

101584 First released in Issue 4.

101585 **Issue 5**

101586 The FUTURE DIRECTIONS section is added.

101587 **Issue 6**

101588 This utility is marked as part of the User Portability Utilities option.

101589 The description of the **-D** option and the steps in [Filename Determination](#) (on page 3048) are  
101590 changed to match historical practice as defined in the IEEE P1003.2b draft standard.

101591 The normative text is reworded to avoid use of the term “must” for application requirements.

101592 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/34 is applied, clarifying the way that the  
101593 *patch* utility performs **ifdef** selection for the **-D** option.101594 **Issue 7**101595 The *patch* utility is moved from the User Portability Utilities option to the Base. User Portability  
101596 Utilities is now an option for interactive utilities.

101597 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

101598 SD5-XCU-ERN-103 and SD5-XCU-ERN-120 are applied, adding the **-u** option.101599 Austin Group Interpretation 1003.1-2001 #126 is applied, changing the description of the  
101600 *LC\_MESSAGES* and *LC\_CTYPE* environment variables and adding the *LC\_COLLATE*  
101601 environment variable.



101602 **NAME**

101603 pathchk — check pathnames

101604 **SYNOPSIS**101605 pathchk [-p] [-P] *pathname*...101606 **DESCRIPTION**

101607 The *pathchk* utility shall check that one or more pathnames are valid (that is, they could be used  
 101608 to access or create a file without causing syntax errors) and portable (that is, no filename  
 101609 truncation results). More extensive portability checks are provided by the **-p** and **-P** options.

101610 By default, the *pathchk* utility shall check each component of each *pathname* operand based on the  
 101611 underlying file system. A diagnostic shall be written for each *pathname* operand that:

- 101612 • Is longer than {PATH\_MAX} bytes (see **Pathname Variable Values** in XBD [<limits.h>](#))
- 101613 • Contains any component longer than {NAME\_MAX} bytes in its containing directory
- 101614 • Contains any component in a directory that is not searchable
- 101615 • Contains any byte sequence that is not valid in its containing directory

101616 The format of the diagnostic message is not specified, but shall indicate the error detected and  
 101617 the corresponding *pathname* operand.

101618 It shall not be considered an error if one or more components of a *pathname* operand do not exist  
 101619 as long as a file matching the pathname specified by the missing components could be created  
 101620 that does not violate any of the checks specified above.

101621 **OPTIONS**101622 The *pathchk* utility shall conform to XBD [Section 12.2](#) (on page 216).

101623 The following option shall be supported:

101624 **-p** Instead of performing checks based on the underlying file system, write a  
 101625 diagnostic for each *pathname* operand that:

- 101626 • Is longer than {\_POSIX\_PATH\_MAX} bytes (see **Minimum Values** in XBD  
 101627 [<limits.h>](#))
- 101628 • Contains any component longer than {\_POSIX\_NAME\_MAX} bytes
- 101629 • Contains any character in any component that is not in the portable filename  
 101630 character set

101631 **-P** Write a diagnostic for each *pathname* operand that:

- 101632 • Contains a component whose first character is the <hyphen-minus>  
 101633 character
- 101634 • Is empty

101635 **OPERANDS**

101636 The following operand shall be supported:

101637 *pathname* A pathname to be checked.101638 **STDIN**

101639 Not used.



101640 **INPUT FILES**

101641 None.

101642 **ENVIRONMENT VARIABLES**101643 The following environment variables shall affect the execution of *pathchk*:

101644 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 101645 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 101646 used to determine the values of locale categories.)

101647 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 101648 internationalization variables.

101649 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 101650 characters (for example, single-byte as opposed to multi-byte characters in  
 101651 arguments).

101652 *LC\_MESSAGES*

101653 Determine the locale that should be used to affect the format and contents of  
 101654 diagnostic messages written to standard error.

101655 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

101656 **ASYNCHRONOUS EVENTS**

101657 Default.

101658 **STDOUT**

101659 Not used.

101660 **STDERR**

101661 The standard error shall be used only for diagnostic messages.

101662 **OUTPUT FILES**

101663 None.

101664 **EXTENDED DESCRIPTION**

101665 None.

101666 **EXIT STATUS**

101667 The following exit values shall be returned:

101668 0 All *pathname* operands passed all of the checks.

101669 &gt;0 An error occurred.

101670 **CONSEQUENCES OF ERRORS**

101671 Default.

101672 **APPLICATION USAGE**

101673 The *test* utility can be used to determine whether a given pathname names an existing file; it  
 101674 does not, however, give any indication of whether or not any component of the pathname was  
 101675 truncated in a directory where the `_POSIX_NO_TRUNC` feature is not in effect. The *pathchk*  
 101676 utility does not check for file existence; it performs checks to determine whether a pathname  
 101677 does exist or could be created with no pathname component truncation.

101678 The *noclobber* option in the shell (see the *set* special built-in) can be used to atomically create a  
 101679 file. As with all file creation semantics in the System Interfaces volume of POSIX.1-2008, it  
 101680 guarantees atomic creation, but still depends on applications to agree on conventions and  
 101681 cooperate on the use of files after they have been created.

101682 To verify that a pathname meets the requirements of filename portability, applications should

use both the `-p` and `-P` options together.

#### EXAMPLES

To verify that all pathnames in an imported data interchange archive are legitimate and unambiguous on the current system:

```
# This example assumes that no pathnames in the archive
# contain <newline> characters.
pax -f archive | sed -e 's/^[[:alnum:]]/\\&/g' | xargs pathchk --
if [ $? -eq 0 ]
then
    pax -r -f archive
else
    echo Investigate problems before importing files.
    exit 1
fi
```

To verify that all files in the current directory hierarchy could be moved to any system conforming to the System Interfaces volume of POSIX.1-2008 that also supports the *pax* utility:

```
find . -exec pathchk -p -P {} +
if [ $? -eq 0 ]
then
    pax -w -f ../archive .
else
    echo Portable archive cannot be created.
    exit 1
fi
```

To verify that a user-supplied pathname names a readable file and that the application can create a file extending the given path without truncation and without overwriting any existing file:

```
case $- in
    *(*)    reset="";;
    *)      reset="set +C"
            set -C;;
esac
test -r "$path" && pathchk "$path.out" &&
rm "$path.out" > "$path.out"
if [ $? -ne 0 ]; then
    printf "%s: %s not found or %s.out fails \
creation checks.\n" $0 "$path" "$path"
    $reset      # Reset the noclobber option in case a trap
                # on EXIT depends on it.
    exit 1
fi
$reset
PROCESSING < "$path" > "$path.out"
```

The following assumptions are made in this example:

1. **PROCESSING** represents the code that is used by the application to use **\$path** once it is verified that **\$path.out** works as intended.

2. The state of the *noclobber* option is unknown when this code is invoked and should be set on exit to the state it was in when this code was invoked. (The **reset** variable is used in this example to restore the initial state.)

3. Note the usage of:

```
rm "$path.out" > "$path.out"
```

- a. The *pathchk* command has already verified, at this point, that **\$path.out** is not truncated.
- b. With the *noclobber* option set, the shell verifies that **\$path.out** does not already exist before invoking *rm*.
- c. If the shell succeeded in creating **\$path.out**, *rm* removes it so that the application can create the file again in the **PROCESSING** step.
- d. If the **PROCESSING** step wants the file to exist already when it is invoked, the:

```
rm "$path.out" > "$path.out"
```

should be replaced with:

```
> "$path.out"
```

which verifies that the file did not already exist, but leaves **\$path.out** in place for use by **PROCESSING**.

#### RATIONALE

The *pathchk* utility was new for the ISO POSIX-2:1993 standard. It, along with the *set -C(noclobber)* option added to the shell, replaces the *mktemp*, *validfnam*, and *create* utilities that appeared in early proposals. All of these utilities were attempts to solve several common problems:

- Verify the validity (for several different definitions of “valid”) of a pathname supplied by a user, generated by an application, or imported from an external source.
- Atomically create a file.
- Perform various string handling functions to generate a temporary filename.

The *create* utility, included in an early proposal, provided checking and atomic creation in a single invocation of the utility; these are orthogonal issues and need not be grouped into a single utility. Note that the *noclobber* option also provides a way of creating a lock for process synchronization; since it provides an atomic *create*, there is no race between a test for existence and the following creation if it did not exist.

Having a function like *tmpnam()* in the ISO C standard is important in many high-level languages. The shell programming language, however, has built-in string manipulation facilities, making it very easy to construct temporary filenames. The names needed obviously depend on the application, but are frequently of a form similar to:

```
$TMPDIR/application_abbreviation$$suffix
```

In cases where there is likely to be contention for a given suffix, a simple shell **for** or **while** loop can be used with the shell *noclobber* option to create a file without risk of collisions, as long as applications trying to use the same filename name space are cooperating on the use of files after they have been created.

For historical purposes, **-p** does not check for the use of the <hyphen-minus> character as the first character in a component of the pathname, or for an empty *pathname* operand.

101770 **FUTURE DIRECTIONS**

101771 None.

101772 **SEE ALSO**101773 [Section 2.7](#) (on page 2350), [set](#) (on page 2399), [test](#)101774 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216), [<limits.h>](#)101775 **CHANGE HISTORY**

101776 First released in Issue 4.

101777 **Issue 7**

101778 Austin Group Interpretations 1003.1-2001 #039, #040, and #094 are applied.

101779 SD5-XCU-ERN-121 is applied, updating the EXAMPLES section.

101780 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0127 [291] is applied.

101781 **NAME**

101782       pax — portable archive interchange

101783 **SYNOPSIS**101784       pax [-dv] [-c|-n] [-H|-L] [-o options] [-f archive] [-s replstr]...  
101785       [*pattern*...]101786       pax -r[-c|-n] [-dikuv] [-H|-L] [-f archive] [-o options]... [-p string]...  
101787       [-s replstr]... [*pattern*...]101788       pax -w [-dituvX] [-H|-L] [-b blocksize] [[-a] [-f archive]] [-o options]...  
101789       [-s replstr]... [-x format] [*file*...]101790       pax -r -w [-diklntuvX] [-H|-L] [-o options]... [-p string]...  
101791       [-s replstr]... [*file*...] *directory*101792 **DESCRIPTION**101793       The *pax* utility shall read, write, and write lists of the members of archive files and copy  
101794       directory hierarchies. A variety of archive formats shall be supported; see the **-x format** option.101795       The action to be taken depends on the presence of the **-r** and **-w** options. The four combinations  
101796       of **-r** and **-w** are referred to as the four modes of operation: **list**, **read**, **write**, and **copy** modes,  
101797       corresponding respectively to the four forms shown in the SYNOPSIS section.101798       **list**       In **list** mode (when neither **-r** nor **-w** are specified), *pax* shall write the names of  
101799       the members of the archive file read from the standard input, with pathnames  
101800       matching the specified patterns, to standard output. If a named file is of type  
101801       directory, the file hierarchy rooted at that file shall be listed as well.101802       **read**       In **read** mode (when **-r** is specified, but **-w** is not), *pax* shall extract the members of  
101803       the archive file read from the standard input, with pathnames matching the  
101804       specified patterns. If an extracted file is of type directory, the file hierarchy rooted  
101805       at that file shall be extracted as well. The extracted files shall be created performing  
101806       pathname resolution with the directory in which *pax* was invoked as the current  
101807       working directory.101808       If an attempt is made to extract a directory when the directory already exists, this  
101809       shall not be considered an error. If an attempt is made to extract a FIFO when the  
101810       FIFO already exists, this shall not be considered an error.101811       The ownership, access, and modification times, and file mode of the restored files  
101812       are discussed under the **-p** option.101813       **write**       In **write** mode (when **-w** is specified, but **-r** is not), *pax* shall write the contents of  
101814       the *file* operands to the standard output in an archive format. If no *file* operands are  
101815       specified, a list of files to copy, one per line, shall be read from the standard input  
101816       and each entry in this list shall be processed as if it had been a *file* operand on the  
101817       command line. A file of type directory shall include all of the files in the file  
101818       hierarchy rooted at the file.101819       **copy**       In **copy** mode (when both **-r** and **-w** are specified), *pax* shall copy the *file* operands  
101820       to the destination directory.101821       If no *file* operands are specified, a list of files to copy, one per line, shall be read  
101822       from the standard input. A file of type directory shall include all of the files in the  
101823       file hierarchy rooted at the file.101824       The effect of the **copy** shall be as if the copied files were written to a *pax* format |  
101825       archive file and then subsequently extracted, except that copying of sockets may be |

supported even if archiving them in write mode is not supported, and that there may be hard links between the original and the copied files. If the destination directory is a subdirectory of one of the files to be copied, the results are unspecified. If the destination directory is a file of a type not defined by the System Interfaces volume of POSIX.1-2008, the results are implementation-defined; otherwise, it shall be an error for the file named by the *directory* operand not to exist, not be writable by the user, or not be a file of type directory.

In **read** or **copy** modes, if intermediate directories are necessary to extract an archive member, *pax* shall perform actions equivalent to the *mkdir()* function defined in the System Interfaces volume of POSIX.1-2008, called with the following arguments:

- The intermediate directory used as the *path* argument
- The value of the bitwise-inclusive OR of S\_IRWXU, S\_IRWXG, and S\_IRWXO as the *mode* argument

If any specified *pattern* or *file* operands are not matched by at least one file or archive member, *pax* shall write a diagnostic message to standard error for each one that did not match and exit with a non-zero exit status.

The archive formats described in the EXTENDED DESCRIPTION section shall be automatically detected on input. The default output archive format shall be implementation-defined.

A single archive can span multiple files. The *pax* utility shall determine, in an implementation-defined manner, what file to read or write as the next file.

If the selected archive format supports the specification of linked files, it shall be an error if these files cannot be linked when the archive is extracted. For archive formats that do not store file contents with each name that causes a hard link, if the file that contains the data is not extracted during this *pax* session, either the data shall be restored from the original file, or a diagnostic message shall be displayed with the name of a file that can be used to extract the data. In traversing directories, *pax* shall detect infinite loops; that is, entering a previously visited directory that is an ancestor of the last file visited. When it detects an infinite loop, *pax* shall write a diagnostic message to standard error and shall terminate.

## OPTIONS

The *pax* utility shall conform to XBD [Section 12.2](#) (on page 216), except that the order of presentation of the **-o**, **-p**, and **-s** options is significant.

The following options shall be supported:

- r** Read an archive file from standard input.
- w** Write files to the standard output in the specified archive format.
- a** Append files to the end of the archive. It is implementation-defined which devices on the system support appending. Additional file formats unspecified by this volume of POSIX.1-2008 may impose restrictions on appending.
- b *blocksize*** Block the output at a positive decimal integer number of bytes per write to the archive file. Devices and archive formats may impose restrictions on blocking. Blocking shall be automatically determined on input. Conforming applications shall not specify a *blocksize* value larger than 32 256. Default blocking when creating archives depends on the archive format. (See the **-x** option below.)
- c** Match all file or archive members except those specified by the *pattern* or *file* operands.

|        |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 101870 | <b>-d</b>         | Cause files of type directory being copied or archived or archive members of type directory being extracted or listed to match only the file or archive member itself and not the file hierarchy rooted at the file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 101871 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101872 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101873 | <b>-f archive</b> | Specify the pathname of the input or output archive, overriding the default standard input (in <b>list</b> or <b>read</b> modes) or standard output ( <b>write</b> mode).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 101874 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101875 | <b>-H</b>         | If a symbolic link referencing a file of type directory is specified on the command line, <i>pax</i> shall archive the file hierarchy rooted in the file referenced by the link, using the name of the link as the root of the file hierarchy. Otherwise, if a symbolic link referencing a file of any other file type which <i>pax</i> can normally archive is specified on the command line, then <i>pax</i> shall archive the file referenced by the link, using the name of the link. The default behavior, when neither <b>-H</b> or <b>-L</b> are specified, shall be to archive the symbolic link itself.                                                                                                                                                                                                                       |
| 101876 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101877 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101878 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101879 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101880 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101881 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101882 | <b>-i</b>         | Interactively rename files or archive members. For each archive member matching a <i>pattern</i> operand or file matching a <i>file</i> operand, a prompt shall be written to the file <b>/dev/tty</b> . The prompt shall contain the name of the file or archive member, but the format is otherwise unspecified. A line shall then be read from <b>/dev/tty</b> . If this line is blank, the file or archive member shall be skipped. If this line consists of a single period, the file or archive member shall be processed with no modification to its name. Otherwise, its name shall be replaced with the contents of the line. The <i>pax</i> utility shall immediately exit with a non-zero exit status if end-of-file is encountered when reading a response or if <b>/dev/tty</b> cannot be opened for reading and writing. |
| 101883 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101884 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101885 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101886 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101887 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101888 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101889 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101890 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101891 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101892 |                   | The results of extracting a hard link to a file that has been renamed during extraction are unspecified.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 101893 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101894 | <b>-k</b>         | Prevent the overwriting of existing files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 101895 | <b>-l</b>         | (The letter ell.) In <b>copy</b> mode, hard links shall be made between the source and destination file hierarchies whenever possible. If specified in conjunction with <b>-H</b> or <b>-L</b> , when a symbolic link is encountered, the hard link created in the destination file hierarchy shall be to the file referenced by the symbolic link. If specified when neither <b>-H</b> nor <b>-L</b> is specified, when a symbolic link is encountered, the implementation shall create a hard link to the symbolic link in the source file hierarchy or copy the symbolic link to the destination.                                                                                                                                                                                                                                   |
| 101896 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101897 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101898 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101899 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101900 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101901 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101902 | <b>-L</b>         | If a symbolic link referencing a file of type directory is specified on the command line or encountered during the traversal of a file hierarchy, <i>pax</i> shall archive the file hierarchy rooted in the file referenced by the link, using the name of the link as the root of the file hierarchy. Otherwise, if a symbolic link referencing a file of any other file type which <i>pax</i> can normally archive is specified on the command line or encountered during the traversal of a file hierarchy, <i>pax</i> shall archive the file referenced by the link, using the name of the link. The default behavior, when neither <b>-H</b> or <b>-L</b> are specified, shall be to archive the symbolic link itself.                                                                                                            |
| 101903 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101904 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101905 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101906 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101907 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101908 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101909 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101910 | <b>-n</b>         | Select the first archive member that matches each <i>pattern</i> operand. No more than one archive member shall be matched for each pattern (although members of type directory shall still match the file hierarchy rooted at that file).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 101911 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101912 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101913 | <b>-o options</b> | Provide information to the implementation to modify the algorithm for extracting or writing files. The value of <i>options</i> shall consist of one or more <comma>-separated keywords of the form:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 101914 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101915 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 101916 |                   | <i>keyword</i> [[:]=value][,keyword[[:]=value], ...]                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |



Some keywords apply only to certain file formats, as indicated with each description. Use of keywords that are inapplicable to the file format being processed produces undefined results.

Keywords in the *options* argument shall be a string that would be a valid portable filename as described in XBD [Section 3.282](#) (on page 79).

**Note:** Keywords are not expected to be filenames, merely to follow the same character composition rules as portable filenames.

Keywords can be preceded with white space. The *value* field shall consist of zero or more characters; within *value*, the application shall precede any literal <comma> with a <backslash>, which shall be ignored, but preserves the <comma> as part of *value*. A <comma> as the final character, or a <comma> followed solely by white space as the final characters, in *options* shall be ignored. Multiple **-o** options can be specified; if keywords given to these multiple **-o** options conflict, the keywords and values appearing later in command line sequence shall take precedence and the earlier shall be silently ignored. The following keyword values of *options* shall be supported for the file formats as indicated:

**delete=pattern**

(Applicable only to the **-x pax** format.) When used in **write** or **copy** mode, *pax* shall omit from extended header records that it produces any keywords matching the string pattern. When used in **read** or **list** mode, *pax* shall ignore any keywords matching the string pattern in the extended header records. In both cases, matching shall be performed using the pattern matching notation described in [Section 2.13.1](#) (on page 2372) and [Section 2.13.2](#) (on page 2373). For example:

**-o delete=security.\***

would suppress security-related information. See [pax Extended Header](#) (on page 3071) for extended header record keyword usage.

When multiple **-odelete=pattern** options are specified, the patterns shall be additive; all keywords matching the specified string patterns shall be omitted from extended header records that *pax* produces.

**exthdr.name=string**

(Applicable only to the **-x pax** format.) This keyword allows user control over the name that is written into the **ustar** header blocks for the extended header produced under the circumstances described in [pax Header Block](#) (on page 3070). The name shall be the contents of *string*, after the following character substitutions have been made:

| <i>string</i><br>Includes: | Replaced by:                                                                                                       |
|----------------------------|--------------------------------------------------------------------------------------------------------------------|
| %d                         | The directory name of the file, equivalent to the result of the <i>dirname</i> utility on the translated pathname. |
| %f                         | The filename of the file, equivalent to the result of the <i>basename</i> utility on the translated pathname.      |
| %p                         | The process ID of the <i>pax</i> process.                                                                          |
| %%                         | A ' % ' character.                                                                                                 |

Any other ' % ' characters in *string* produce undefined results.

If no **-o exthdr.name=string** is specified, *pax* shall use the following default



value:  
`%d/PaxHeaders.%p/%f`

#### **globexthdr.name=string**

(Applicable only to the **-x pax** format.) When used in **write** or **copy** mode with the appropriate options, *pax* shall create global extended header records with **ustar** header blocks that will be treated as regular files by previous versions of *pax*. This keyword allows user control over the name that is written into the **ustar** header blocks for global extended header records. The name shall be the contents of *string*, after the following character substitutions have been made:

| <i>string</i><br>Includes: | Replaced by:                                                                                                       |
|----------------------------|--------------------------------------------------------------------------------------------------------------------|
| <code>%n</code>            | An integer that represents the sequence number of the global extended header record in the archive, starting at 1. |
| <code>%p</code>            | The process ID of the <i>pax</i> process.                                                                          |
| <code>%%</code>            | A <code>'%'</code> character.                                                                                      |

Any other `'%'` characters in *string* produce undefined results.

If no **-o globexthdr.name=string** is specified, *pax* shall use the following default value:

`$TMPDIR/GlobalHead.%p.%n`

where *\$TMPDIR* represents the value of the *TMPDIR* environment variable. If *TMPDIR* is not set, *pax* shall use **/tmp**.

#### **invalid=action**

(Applicable only to the **-x pax** format.) This keyword allows user control over the action *pax* takes upon encountering values in an extended header record that, in **read** or **copy** mode, are invalid in the destination hierarchy or, in **list** mode, cannot be written in the codeset and current locale of the implementation. The following are invalid values that shall be recognized by *pax*:

- In **read** or **copy** mode, a filename or link name that contains character encodings invalid in the destination hierarchy. (For example, the name may contain embedded NULs.)
- In **read** or **copy** mode, a filename or link name that is longer than the maximum allowed in the destination hierarchy (for either a pathname component or the entire pathname).
- In **list** mode, any character string value (filename, link name, user name, and so on) that cannot be written in the codeset and current locale of the implementation.

The following mutually-exclusive values of the *action* argument are supported:

**binary** In **write** mode, *pax* shall generate a **hdrcharset=BINARY** extended header record for each file with a filename, link name, group name, owner name, or any other field in an extended header record that cannot be translated to the UTF-8 codeset, allowing the archive to contain the files with unencoded extended header record values. In **read** or **copy** mode, *pax* shall

|        |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------|-----------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 102008 |                       | use the values specified in the header without translation, regardless of whether this may overwrite an existing file with a valid name. In <b>list</b> mode, <i>pax</i> shall behave identically to the <b>bypass</b> action.                                                                                                                                                                                                                                                                                                                                                                                          |
| 102009 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102010 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102011 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102012 | <b>bypass</b>         | In <b>read</b> or <b>copy</b> mode, <i>pax</i> shall bypass the file, causing no change to the destination hierarchy. In <b>list</b> mode, <i>pax</i> shall write all requested valid values for the file, but its method for writing invalid values is unspecified.                                                                                                                                                                                                                                                                                                                                                    |
| 102013 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102014 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102015 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102016 | <b>rename</b>         | In <b>read</b> or <b>copy</b> mode, <i>pax</i> shall act as if the <b>-i</b> option were in effect for each file with invalid filename or link name values, allowing the user to provide a replacement name interactively. In <b>list</b> mode, <i>pax</i> shall behave identically to the <b>bypass</b> action.                                                                                                                                                                                                                                                                                                        |
| 102017 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102018 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102019 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102020 | <b>UTF-8</b>          | When used in <b>read</b> , <b>copy</b> , or <b>list</b> mode and a filename, link name, owner name, or any other field in an extended header record cannot be translated from the <b>pax</b> UTF-8 codeset format to the codeset and current locale of the implementation, <i>pax</i> shall use the actual UTF-8 encoding for the name. If a <b>hdrcharset</b> extended header record is in effect for this file, the character set specified by that record shall be used instead of UTF-8. If a <b>hdrcharset=BINARY</b> extended header record is in effect for this file, no translation shall be performed.        |
| 102021 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102022 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102023 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102024 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102025 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102026 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102027 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102028 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102029 | <b>write</b>          | In <b>read</b> or <b>copy</b> mode, <i>pax</i> shall write the file, translating the name, regardless of whether this may overwrite an existing file with a valid name. In <b>list</b> mode, <i>pax</i> shall behave identically to the <b>bypass</b> action.                                                                                                                                                                                                                                                                                                                                                           |
| 102030 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102031 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102032 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102033 |                       | If no <b>-o invalid=option</b> is specified, <i>pax</i> shall act as if <b>-o invalid=bypass</b> were specified. Any overwriting of existing files that may be allowed by the <b>-o invalid=</b> actions shall be subject to permission ( <b>-p</b> ) and modification time ( <b>-u</b> ) restrictions, and shall be suppressed if the <b>-k</b> option is also specified.                                                                                                                                                                                                                                              |
| 102034 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102035 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102036 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102037 | <b>linkdata</b>       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102038 |                       | (Applicable only to the <b>-x pax</b> format.) In <b>write</b> mode, <i>pax</i> shall write the contents of a file to the archive even when that file is merely a hard link to a file whose contents have already been written to the archive.                                                                                                                                                                                                                                                                                                                                                                          |
| 102039 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102040 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102041 | <b>listopt=format</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102042 |                       | This keyword specifies the output format of the table of contents produced when the <b>-v</b> option is specified in <b>list</b> mode. See <a href="#">List Mode Format Specifications</a> (on page 3065). To avoid ambiguity, the <b>listopt=format</b> shall be the only or final <b>keyword=value</b> pair in a <b>-o</b> option-argument; all characters in the remainder of the option-argument shall be considered part of the format string. When multiple <b>-olistopt=format</b> options are specified, the format strings shall be considered a single, concatenated string, evaluated in command line order. |
| 102043 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102044 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102045 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102046 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102047 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102048 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102049 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102050 | <b>times</b>          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102051 |                       | (Applicable only to the <b>-x pax</b> format.) When used in <b>write</b> or <b>copy</b> mode, <i>pax</i> shall include <b>atime</b> and <b>mtime</b> extended header records for each file. See <a href="#">pax Extended Header File Times</a> (on page 3074).                                                                                                                                                                                                                                                                                                                                                          |
| 102052 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102053 |                       |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102054 |                       | In addition to these keywords, if the <b>-x pax</b> format is specified, any of the                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |

keywords and values defined in [pax Extended Header](#) (on page 3071), including implementation extensions, can be used in **-o** option-arguments, in either of two modes:

**keyword=value**

When used in **write** or **copy** mode, these keyword/value pairs shall be included at the beginning of the archive as **typeflag g** global extended header records. When used in **read** or **list** mode, these keyword/value pairs shall act as if they had been at the beginning of the archive as **typeflag g** global extended header records.

**keyword:=value**

When used in **write** or **copy** mode, these keyword/value pairs shall be included as records at the beginning of a **typeflag x** extended header for each file. (This shall be equivalent to the `<equals-sign>` form except that it creates no **typeflag g** global extended header records.) When used in **read** or **list** mode, these keyword/value pairs shall act as if they were included as records at the end of each extended header; thus, they shall override any global or file-specific extended header record keywords of the same names. For example, in the command:

```
pax -r -o "
  gname:=mygroup,
  " <archive
```

the group name will be forced to a new value for all files read from the archive.

The precedence of **-o** keywords over various fields in the archive is described in [pax Extended Header Keyword Precedence](#) (on page 3074). If the **-o delete=pattern**, **-o keyword=value**, or **-o keyword:=value** options are used to override or remove any extended header data needed to find files in an archive (e.g., **-o delete=size** for a file whose size cannot be represented in a **ustar** header or **-o size=100** for a file whose size is not 100 bytes), the behavior is undefined.

**-p string**

Specify one or more file characteristic options (privileges). The *string* option-argument shall be a string specifying file characteristics to be retained or discarded on extraction. The string shall consist of the specification characters **a**, **e**, **m**, **o**, and **p**. Other implementation-defined characters can be included. Multiple characteristics can be concatenated within the same string and multiple **-p** options can be specified. The meaning of the specification characters are as follows:

- a** Do not preserve file access times.
- e** Preserve the user ID, group ID, file mode bits (see [XBD Section 3.169](#), on page 60), access time, modification time, and any other implementation-defined file characteristics.
- m** Do not preserve file modification times.
- o** Preserve the user ID and group ID.
- p** Preserve the file mode bits. Other implementation-defined file mode attributes may be preserved.

In the preceding list, “preserve” indicates that an attribute stored in the archive shall be given to the extracted file, subject to the permissions of the invoking

process. The access and modification times of the file shall be preserved unless otherwise specified with the `-p` option or not stored in the archive. All attributes that are not preserved shall be determined as part of the normal file creation action (see [Section 1.1.1.4](#), on page 2318).

If neither the `e` nor the `o` specification character is specified, or the user ID and group ID are not preserved for any reason, *pax* shall not set the `S_ISUID` and `S_ISGID` bits of the file mode.

If the preservation of any of these items fails for any reason, *pax* shall write a diagnostic message to standard error. Failure to preserve these items shall affect the final exit status, but shall not cause the extracted file to be deleted.

If file characteristic letters in any of the *string* option-arguments are duplicated or conflict with each other, the ones given last shall take precedence. For example, if `-p eme` is specified, file modification times are preserved.

`-s replstr` Modify file or archive member names named by *pattern* or *file* operands according to the substitution expression *replstr*, using the syntax of the *ed* utility. The concepts of “address” and “line” are meaningless in the context of the *pax* utility, and shall not be supplied. The format shall be:

```
-s /old/new/[gp]
```

where as in *ed*, *old* is a basic regular expression and *new* can contain an `<ampersand>`, `'\n'` (where *n* is a digit) back-references, or subexpression matching. The *old* string shall also be permitted to contain `<newline>` characters.

Any non-null character can be used as a delimiter (`'/'` shown here). Multiple `-s` expressions can be specified; the expressions shall be applied in the order specified, terminating with the first successful substitution. The optional trailing `'g'` is as defined in the *ed* utility. The optional trailing `'p'` shall cause successful substitutions to be written to standard error. File or archive member names that substitute to the empty string shall be ignored when reading and writing archives.

`-t` When reading files from the file system, and if the user has the permissions required by *utime()* to do so, set the access time of each file read to the access time that it had before being read by *pax*.

`-u` Ignore files that are older (having a less recent file modification time) than a pre-existing file or archive member with the same name. In **read** mode, an archive member with the same name as a file in the file system shall be extracted if the archive member is newer than the file. In **write** mode, an archive file member with the same name as a file in the file system shall be superseded if the file is newer than the archive member. If `-a` is also specified, this is accomplished by appending to the archive; otherwise, it is unspecified whether this is accomplished by actual replacement in the archive or by appending to the archive. In **copy** mode, the file in the destination hierarchy shall be replaced by the file in the source hierarchy or by a link to the file in the source hierarchy if the file in the source hierarchy is newer.

`-v` In **list** mode, produce a verbose table of contents (see the **STDOUT** section). Otherwise, write archive member pathnames to standard error (see the **STDERR** section).

|        |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------|------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 102145 | <b>-x format</b> | Specify the output archive format. The <i>pax</i> utility shall support the following formats:                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 102146 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102147 | <b>cpio</b>      | The <b>cpio</b> interchange format; see the EXTENDED DESCRIPTION section. The default <i>blocksize</i> for this format for character special archive files shall be 5120. Implementations shall support all <i>blocksize</i> values less than or equal to 32768 that are multiples of 512.                                                                                                                                                                                                                                                     |
| 102148 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102149 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102150 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102151 | <b>pax</b>       | The <b>pax</b> interchange format; see the EXTENDED DESCRIPTION section. The default <i>blocksize</i> for this format for character special archive files shall be 5120. Implementations shall support all <i>blocksize</i> values less than or equal to 32768 that are multiples of 512.                                                                                                                                                                                                                                                      |
| 102152 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102153 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102154 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102155 | <b>ustar</b>     | The <b>tar</b> interchange format; see the EXTENDED DESCRIPTION section. The default <i>blocksize</i> for this format for character special archive files shall be 10240. Implementations shall support all <i>blocksize</i> values less than or equal to 32768 that are multiples of 512.                                                                                                                                                                                                                                                     |
| 102156 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102157 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102158 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102159 |                  | Implementation-defined formats shall specify a default block size as well as any other block sizes supported for character special archive files.                                                                                                                                                                                                                                                                                                                                                                                              |
| 102160 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102161 |                  | Any attempt to append to an archive file in a format different from the existing archive format shall cause <i>pax</i> to exit immediately with a non-zero exit status.                                                                                                                                                                                                                                                                                                                                                                        |
| 102162 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102163 | <b>-X</b>        | When traversing the file hierarchy specified by a pathname, <i>pax</i> shall not descend into directories that have a different device ID ( <i>st_dev</i> ; see the System Interfaces volume of POSIX.1-2008, <i>stat()</i> ).                                                                                                                                                                                                                                                                                                                 |
| 102164 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102165 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102166 |                  | Specifying more than one of the mutually-exclusive options <b>-H</b> and <b>-L</b> shall not be considered an error and the last option specified shall determine the behavior of the utility.                                                                                                                                                                                                                                                                                                                                                 |
| 102167 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102168 |                  | The options that operate on the names of files or archive members ( <b>-c</b> , <b>-i</b> , <b>-n</b> , <b>-s</b> , <b>-u</b> , and <b>-v</b> ) shall interact as follows. In <b>read</b> mode, the archive members shall be selected based on the user-specified <i>pattern</i> operands as modified by the <b>-c</b> , <b>-n</b> , and <b>-u</b> options. Then, any <b>-s</b> and <b>-i</b> options shall modify, in that order, the names of the selected files. The <b>-v</b> option shall write names resulting from these modifications. |
| 102169 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102170 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102171 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102172 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102173 |                  | In <b>write</b> mode, the files shall be selected based on the user-specified pathnames as modified by the <b>-n</b> and <b>-u</b> options. Then, any <b>-s</b> and <b>-i</b> options shall modify, in that order, the names of these selected files. The <b>-v</b> option shall write names resulting from these modifications.                                                                                                                                                                                                               |
| 102174 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102175 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102176 |                  | If both the <b>-u</b> and <b>-n</b> options are specified, <i>pax</i> shall not consider a file selected unless it is newer than the file to which it is compared.                                                                                                                                                                                                                                                                                                                                                                             |
| 102177 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102178 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102179 |                  | <b>List Mode Format Specifications</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102180 |                  | In <b>list</b> mode with the <b>-o listopt=format</b> option, the <i>format</i> argument shall be applied for each selected file. The <i>pax</i> utility shall append a <newline> to the <b>listopt</b> output for each selected file.                                                                                                                                                                                                                                                                                                         |
| 102181 |                  | The <i>format</i> argument shall be used as the <i>format</i> string described in XBD Chapter 5 (on page 121), with the exceptions 1. through 6. defined in the EXTENDED DESCRIPTION section of <i>printf</i> , plus the following exceptions:                                                                                                                                                                                                                                                                                                 |
| 102182 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102183 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102184 |                  | 7. The sequence ( <i>keyword</i> ) can occur before a format conversion specifier. The conversion argument is defined by the value of <i>keyword</i> . The implementation shall support the following keywords:                                                                                                                                                                                                                                                                                                                                |
| 102185 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102186 |                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |

- 102187 — Any of the Field Name entries in Table 4-14 (on page 3075) and Table 4-16 (on page  
102188 3078). The implementation may support the *cpio* keywords without the leading *c\_* in  
102189 addition to the form required by Table 4-16 (on page 3078).
- 102190 — Any keyword defined for the extended header in pax Extended Header (on page  
102191 3071).
- 102192 — Any keyword provided as an implementation-defined extension within the extended  
102193 header defined in pax Extended Header (on page 3071).

102194 For example, the sequence "%(charset)s" is the string value of the name of the character  
102195 set in the extended header.

102196 The result of the keyword conversion argument shall be the value from the applicable  
102197 header field or extended header, without any trailing NULs.

102198 All keyword values used as conversion arguments shall be translated from the UTF-8  
102199 encoding (or alternative encoding specified by any **hdrcharset** extended header record) to  
102200 the character set appropriate for the local file system, user database, and so on, as  
102201 applicable.

- 102202 8. An additional conversion specifier character, T, shall be used to specify time formats. The T  
102203 conversion specifier character can be preceded by the sequence (*keyword=subformat*), where  
102204 *subformat* is a date format as defined by *date* operands. The default *keyword* shall be **mtime**  
102205 and the default subformat shall be:

102206 %b %e %H:%M %Y

- 102207 9. An additional conversion specifier character, M, shall be used to specify the file mode string  
102208 as defined in *ls* Standard Output. If (*keyword*) is omitted, the **mode** keyword shall be used.  
102209 For example, %.1M writes the single character corresponding to the <entry type> field of the  
102210 *ls -l* command.

- 102211 10. An additional conversion specifier character, D, shall be used to specify the device for block  
102212 or special files, if applicable, in an implementation-defined format. If not applicable, and  
102213 (*keyword*) is specified, then this conversion shall be equivalent to %(*keyword*)u. If not  
102214 applicable, and (*keyword*) is omitted, then this conversion shall be equivalent to <space>.

- 102215 11. An additional conversion specifier character, F, shall be used to specify a pathname. The F  
102216 conversion character can be preceded by a sequence of <comma>-separated keywords:

102217 (*keyword*[,*keyword*] ... )

102218 The values for all the keywords that are non-null shall be concatenated together, each  
102219 separated by a ' / '. The default shall be (**path**) if the keyword **path** is defined; otherwise,  
102220 the default shall be (**prefix,name**).

- 102221 12. An additional conversion specifier character, L, shall be used to specify a symbolic link  
102222 expansion. If the current file is a symbolic link, then %L shall expand to:

102223 "%s -> %s", <value of keyword>, <contents of link>

102224 Otherwise, the %L conversion specification shall be the equivalent of %F.

## 102225 OPERANDS

102226 The following operands shall be supported:

102227 *directory* The destination directory pathname for **copy** mode.



|        |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
|--------|---------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 102228 | <i>file</i>                                                                                                         | A pathname of a file to be copied or archived.                                                                                                                                                                                                                                                                                                                                          |
| 102229 | <i>pattern</i>                                                                                                      | A pattern matching one or more pathnames of archive members. A pattern must be given in the name-generating notation of the pattern matching notation in <a href="#">Section 2.13</a> (on page 2372), including the filename expansion rules in <a href="#">Section 2.13.3</a> (on page 2373). The default, if no <i>pattern</i> is specified, is to select all members in the archive. |
| 102230 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102231 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102232 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102233 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102234 | <b>STDIN</b>                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102235 | In <b>write</b> mode, the standard input shall be used only if no <i>file</i> operands are specified. It shall be a |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102236 | file containing a list of pathnames, each terminated by a <newline> character.                                      |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102237 | In <b>list</b> and <b>read</b> modes, if <b>-f</b> is not specified, the standard input shall be an archive file.   |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102238 | Otherwise, the standard input shall not be used.                                                                    |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102239 | <b>INPUT FILES</b>                                                                                                  |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102240 | The input file named by the <i>archive</i> option-argument, or standard input when the archive is read              |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102241 | from there, shall be a file formatted according to one of the specifications in the EXTENDED                        |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102242 | DESCRIPTION section or some other implementation-defined format.                                                    |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102243 | The file <b>/dev/tty</b> shall be used to write prompts and read responses.                                         |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102244 | <b>ENVIRONMENT VARIABLES</b>                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102245 | The following environment variables shall affect the execution of <i>pax</i> :                                      |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102246 | <i>LANG</i>                                                                                                         | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) the precedence of internationalization variables used to determine the values of locale categories.)                                                                                                                                          |
| 102247 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102248 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102249 | <i>LC_ALL</i>                                                                                                       | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                                                |
| 102250 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102251 | <i>LC_COLLATE</i>                                                                                                   | Determine the locale for the behavior of ranges, equivalence classes, and multi-character collating elements used in the pattern matching expressions for the <i>pattern</i> operand, the basic regular expression for the <b>-s</b> option, and the extended regular expression defined for the <b>yesexpr</b> locale keyword in the <i>LC_MESSAGES</i> category.                      |
| 102252 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102253 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102254 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102255 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102256 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102257 | <i>LC_CTYPE</i>                                                                                                     | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files), the behavior of character classes used in the extended regular expression defined for the <b>yesexpr</b> locale keyword in the <i>LC_MESSAGES</i> category, and pattern matching.             |
| 102258 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102259 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102260 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102261 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102262 | <i>LC_MESSAGES</i>                                                                                                  | Determine the locale used to process affirmative responses, and the locale used to affect the format and contents of diagnostic messages and prompts written to standard error.                                                                                                                                                                                                         |
| 102263 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102264 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102265 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102266 | <i>LC_TIME</i>                                                                                                      | Determine the format and contents of date and time strings when the <b>-v</b> option is specified.                                                                                                                                                                                                                                                                                      |
| 102267 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |
| 102268 | XSI <i>NLSPATH</i>                                                                                                  | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                                                   |
| 102269 | <i>TMPDIR</i>                                                                                                       | Determine the pathname that provides part of the default global extended header record file, as described for the <b>-o globexthdr=</b> keyword in the OPTIONS section.                                                                                                                                                                                                                 |
| 102270 |                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                         |

102271 *TZ* Determine the timezone used to calculate date and time strings when the *-v* option  
 102272 is specified. If *TZ* is unset or null, an unspecified default timezone shall be used.

## 102273 ASYNCHRONOUS EVENTS

102274 Default.

## 102275 STDOUT

102276 In **write** mode, if *-f* is not specified, the standard output shall be the archive formatted  
 102277 according to one of the specifications in the EXTENDED DESCRIPTION section, or some other  
 102278 implementation-defined format (see *-x format*).

102279 In **list** mode, when the *-olistopt=format* has been specified, the selected archive members shall  
 102280 be written to standard output using the format described under [List Mode Format Specifications](#)  
 102281 (on page 3065). In **list** mode without the *-olistopt=format* option, the table of contents of the  
 102282 selected archive members shall be written to standard output using the following format:

102283 "%s\n", <pathname>

102284 If the *-v* option is specified in **list** mode, the table of contents of the selected archive members  
 102285 shall be written to standard output using the following formats.

102286 For pathnames representing hard links to previous members of the archive:

102287 "%sΔ==Δ%s\n", <ls -l listing>, <linkname>

102288 For all other pathnames:

102289 "%s\n", <ls -l listing>

102290 where <ls -l listing> shall be the format specified by the *ls* utility with the *-l* option. When  
 102291 writing pathnames in this format, it is unspecified what is written for fields for which the  
 102292 underlying archive format does not have the correct information, although the correct number of  
 102293 <blank>-separated fields shall be written.

102294 In **list** mode, standard output shall not be buffered more than a pathname (plus any associated  
 102295 information and a <newline> terminator) at a time.

## 102296 STDERR

102297 If *-v* is specified in **read**, **write**, or **copy** modes, *pax* shall write the pathnames it processes to the  
 102298 standard error output using the following format:

102299 "%s\n", <pathname>

102300 These pathnames shall be written as soon as processing is begun on the file or archive member,  
 102301 and shall be flushed to standard error. The trailing <newline>, which shall not be buffered, is  
 102302 written when the file has been read or written.

102303 If the *-s* option is specified, and the replacement string has a trailing 'p', substitutions shall be  
 102304 written to standard error in the following format:

102305 "%sΔ>>Δ%s\n", <original pathname>, <new pathname>

102306 In all operating modes of *pax*, optional messages of unspecified format concerning the input  
 102307 archive format and volume number, the number of files, blocks, volumes, and media parts as  
 102308 well as other diagnostic messages may be written to standard error.

102309 In all formats, for both standard output and standard error, it is unspecified how non-printable  
 102310 characters in pathnames or link names are written.

102311 When using the *-xpax* archive format, if a filename, link name, group name, owner name, or any  
 102312 other field in an extended header record cannot be translated between the codeset in use for that



102313 extended header record and the character set of the current locale, *pax* shall write a diagnostic  
102314 message to standard error, shall process the file as described for the **-o invalid=** option, and then  
102315 shall continue processing with the next file.

## 102316 OUTPUT FILES

102317 In **read** mode, the extracted output files shall be of the archived file type. In **copy** mode, the  
102318 copied output files shall be the type of the file being copied. In either mode, existing files in the  
102319 destination hierarchy shall be overwritten only when all permission (**-p**), modification time (**-u**),  
102320 and invalid-value (**-o invalid=**) tests allow it.

102321 In **write** mode, the output file named by the **-f** option-argument shall be a file formatted  
102322 according to one of the specifications in the EXTENDED DESCRIPTION section, or some other  
102323 implementation-defined format.

## 102324 EXTENDED DESCRIPTION

### 102325 **pax Interchange Format**

102326 A *pax* archive tape or file produced in the **-xpax** format shall contain a series of blocks. The  
102327 physical layout of the archive shall be identical to the **ustar** format described in **ustar**  
102328 **Interchange Format** (on page 3075). Each file archived shall be represented by the following  
102329 sequence:

- 102330 • An optional header block with extended header records. This header block is of the form  
102331 described in **pax Header Block** (on page 3070), with a *typeflag* value of **x** or **g**. The  
102332 extended header records, described in **pax Extended Header** (on page 3071), shall be  
102333 included as the data for this header block.
- 102334 • A header block that describes the file. Any fields in the preceding optional extended  
102335 header shall override the associated fields in this header block for this file.
- 102336 • Zero or more blocks that contain the contents of the file.

102337 At the end of the archive file there shall be two 512-byte blocks filled with binary zeros,  
102338 interpreted as an end-of-archive indicator.

102339 A schematic of an example archive with global extended header records and two actual files is  
102340 shown in **Figure 4-1** (on page 3070). In the example, the second file in the archive has no  
102341 extended header preceding it, presumably because it has no need for extended attributes.

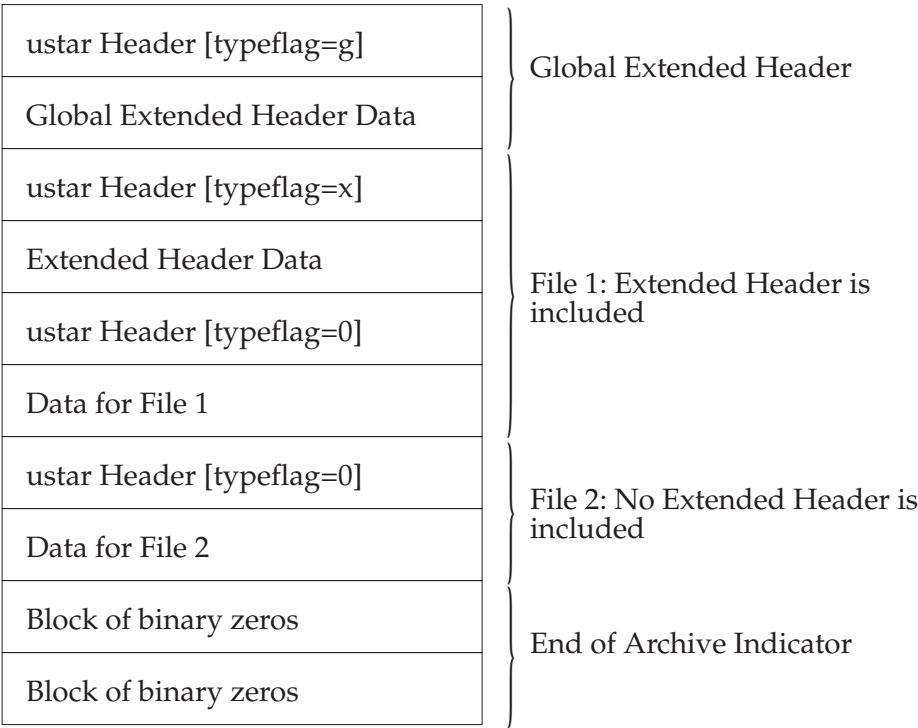


Figure 4-1 pax Format Archive Example

**pax Header Block**

The **pax** header block shall be identical to the **ustar** header block described in **ustar Interchange Format** (on page 3075), except that two additional *typeflag* values are defined:

- ⌘ Represents extended header records for the following file in the archive (which shall have its own **ustar** header block). The format of these extended header records shall be as described in **pax Extended Header** (on page 3071).
- g Represents global extended header records for the following files in the archive. The format of these extended header records shall be as described in **pax Extended Header** (on page 3071). Each value shall affect all subsequent files that do not override that value in their own extended header record and until another global extended header record is reached that provides another value for the same field. The *typeflag* **g** global headers should not be used with interchange media that could suffer partial data loss in transporting the archive.

For both of these types, the *size* field shall be the size of the extended header records in octets. The other fields in the header block are not meaningful to this version of the *pax* utility. However, if this archive is read by a *pax* utility conforming to the ISO POSIX-2: 1993 standard, the header block fields are used to create a regular file that contains the extended header records as data. Therefore, header block field values should be selected to provide reasonable file access to this regular file.

A further difference from the **ustar** header block is that data blocks for files of *typeflag* 1 (the digit one) (hard link) may be included, which means that the size field may be greater than zero. Archives created by *pax -o linkdata* shall include these data blocks with the hard links.

**pax Extended Header**

A **pax** extended header contains values that are inappropriate for the **ustar** header block because of limitations in that format: fields requiring a character encoding other than that described in the ISO/IEC 646:1991 standard, fields representing file attributes not described in the **ustar** header, and fields whose format or length do not fit the requirements of the **ustar** header. The values in an extended header add attributes to the following file (or files; see the description of the *typeflag* **g** header block) or override values in the following header block(s), as indicated in the following list of keywords.

An extended header shall consist of one or more records, each constructed as follows:

```
"%d %s=%s\n", <length>, <keyword>, <value>
```

The extended header records shall be encoded according to the ISO/IEC 10646-1:2000 standard UTF-8 encoding. The *<length>* field, *<blank>*, *<equals-sign>*, and *<newline>* shown shall be limited to the portable character set, as encoded in UTF-8. The *<keyword>* fields can be any UTF-8 characters. The *<length>* field shall be the decimal length of the extended header record in octets, including the trailing *<newline>*. If there is a **hdrcharset** extended header in effect for a file, the *value* field for any **gname**, **linkpath**, **path**, and **uname** extended header records shall be encoded using the character set specified by the **hdrcharset** extended header record; otherwise, the *value* field shall be encoded using UTF-8. The *value* field for all other keywords specified by POSIX.1-2008 shall be encoded using UTF-8.

The *<keyword>* field shall be one of the entries from the following list or a keyword provided as an implementation extension. Keywords consisting entirely of lowercase letters, digits, and periods are reserved for future standardization. A keyword shall not include an *<equals-sign>*. (In the following list, the notations “file(s)” or “block(s)” is used to acknowledge that a keyword affects the following single file after a *typeflag* **x** extended header, but possibly multiple files after *typeflag* **g**. Any requirements in the list for *pax* to include a record when in **write** or **copy** mode shall apply only when such a record has not already been provided through the use of the **-o** option. When used in **copy** mode, *pax* shall behave as if an archive had been created with applicable extended header records and then extracted.)

**atime** The file access time for the following file(s), equivalent to the value of the *st\_atime* member of the **stat** structure for a file, as described by the *stat()* function. The access time shall be restored if the process has appropriate privileges required to do so. The format of the *<value>* shall be as described in [pax Extended Header File Times](#) (on page 3074).

**charset** The name of the character set used to encode the data in the following file(s). The entries in the following table are defined to refer to known standards; additional names may be agreed on between the originator and recipient.

102400  
102401  
102402  
102403  
102404  
102405  
102406  
102407  
102408  
102409  
102410  
102411  
102412  
102413  
102414  
102415  
102416  
102417

| <value>                 | Formal Standard               |
|-------------------------|-------------------------------|
| ISO-IRΔ646Δ1990         | ISO/IEC 646: 1990             |
| ISO-IRΔ8859Δ1Δ1998      | ISO/IEC 8859-1: 1998          |
| ISO-IRΔ8859Δ2Δ1999      | ISO/IEC 8859-2: 1999          |
| ISO-IRΔ8859Δ3Δ1999      | ISO/IEC 8859-3: 1999          |
| ISO-IRΔ8859Δ4Δ1998      | ISO/IEC 8859-4: 1998          |
| ISO-IRΔ8859Δ5Δ1999      | ISO/IEC 8859-5: 1999          |
| ISO-IRΔ8859Δ6Δ1999      | ISO/IEC 8859-6: 1999          |
| ISO-IRΔ8859Δ7Δ1987      | ISO/IEC 8859-7: 1987          |
| ISO-IRΔ8859Δ8Δ1999      | ISO/IEC 8859-8: 1999          |
| ISO-IRΔ8859Δ9Δ1999      | ISO/IEC 8859-9: 1999          |
| ISO-IRΔ8859Δ10Δ1998     | ISO/IEC 8859-10: 1998         |
| ISO-IRΔ8859Δ13Δ1998     | ISO/IEC 8859-13: 1998         |
| ISO-IRΔ8859Δ14Δ1998     | ISO/IEC 8859-14: 1998         |
| ISO-IRΔ8859Δ15Δ1999     | ISO/IEC 8859-15: 1999         |
| ISO-IRΔ10646Δ2000       | ISO/IEC 10646: 2000           |
| ISO-IRΔ10646Δ2000ΔUTF-8 | ISO/IEC 10646, UTF-8 encoding |
| BINARY                  | None.                         |

102418  
102419  
102420

The encoding is included in an extended header for information only; when *pax* is used as described in POSIX.1-2008, it shall not translate the file data into any other encoding. The **BINARY** entry indicates unencoded binary data.

102421  
102422

When used in **write** or **copy** mode, it is implementation-defined whether *pax* includes a **charset** extended header record for a file.

102423  
102424

**comment**

A series of characters used as a comment. All characters in the <value> field shall be ignored by *pax*.

102425  
102426  
102427  
102428  
102429

**gid**

The group ID of the group that owns the file, expressed as a decimal number using digits from the ISO/IEC 646: 1991 standard. This record shall override the *gid* field in the following header block(s). When used in **write** or **copy** mode, *pax* shall include a *gid* extended header record for each file whose group ID is greater than 2 097 151 (octal 7 777 777).

102430  
102431  
102432  
102433  
102434  
102435  
102436  
102437  
102438  
102439

**gname**

The group of the file(s), formatted as a group name in the group database. This record shall override the *gid* and *gname* fields in the following header block(s), and any *gid* extended header record. When used in **read**, **copy**, or **list** mode, *pax* shall translate the name from the encoding in the header record to the character set appropriate for the group database on the receiving system. If any of the characters cannot be translated, and if neither the **-oinvalid=UTF-8** option nor the **-oinvalid=binary** option is specified, the results are implementation-defined. When used in **write** or **copy** mode, *pax* shall include a **gname** extended header record for each file whose group name cannot be represented entirely with the letters and digits of the portable character set.

102440  
102441  
102442  
102443

**hdrcharset**

The name of the character set used to encode the value field of the **gname**, **linkpath**, **path**, and **uname** *pax* extended header records. The entries in the following table are defined to refer to known standards; additional names may be agreed between the originator and the recipient.

102444  
102445  
102446

| <value>                 | Formal Standard               |
|-------------------------|-------------------------------|
| ISO-IRΔ10646Δ2000ΔUTF-8 | ISO/IEC 10646, UTF-8 encoding |
| BINARY                  | None.                         |

|        |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|--------|---------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 102447 |                     | If no <b>hdrcharset</b> extended header record is specified, the default character set used to encode all values in extended header records shall be the ISO/IEC 10646-1: 2000 standard UTF-8 encoding.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 102448 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102449 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102450 |                     | The <b>BINARY</b> entry indicates that all values recorded in extended headers for affected files are unencoded binary data from the underlying system.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 102451 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102452 | <b>linkpath</b>     | The pathname of a link being created to another file, of any type, previously archived. This record shall override the <i>linkname</i> field in the following <b>ustar</b> header block(s). The following <b>ustar</b> header block shall determine the type of link created. If <i>typeflag</i> of the following header block is 1, it shall be a hard link. If <i>typeflag</i> is 2, it shall be a symbolic link and the <b>linkpath</b> value shall be the contents of the symbolic link. The <i>pax</i> utility shall translate the name of the link (contents of the symbolic link) from the encoding in the header to the character set appropriate for the local file system. When used in <b>write</b> or <b>copy</b> mode, <i>pax</i> shall include a <b>linkpath</b> extended header record for each link whose pathname cannot be represented entirely with the members of the portable character set other than NUL. |
| 102453 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102454 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102455 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102456 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102457 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102458 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102459 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102460 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102461 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102462 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102463 | <b>mtime</b>        | The file modification time of the following file(s), equivalent to the value of the <i>st_mtime</i> member of the <b>stat</b> structure for a file, as described in the <i>stat()</i> function. This record shall override the <i>mtime</i> field in the following header block(s). The modification time shall be restored if the process has appropriate privileges required to do so. The format of the <i>&lt;value&gt;</i> shall be as described in <b>pax Extended Header File Times</b> (on page 3074).                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 102464 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102465 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102466 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102467 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102468 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102469 | <b>path</b>         | The pathname of the following file(s). This record shall override the <i>name</i> and <i>prefix</i> fields in the following header block(s). The <i>pax</i> utility shall translate the pathname of the file from the encoding in the header to the character set appropriate for the local file system.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 102470 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102471 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102472 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102473 |                     | When used in <b>write</b> or <b>copy</b> mode, <i>pax</i> shall include a <i>path</i> extended header record for each file whose pathname cannot be represented entirely with the members of the portable character set other than NUL.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 102474 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102475 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102476 | <b>realtime.any</b> | The keywords prefixed by “realtime.” are reserved for future standardization.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 102477 | <b>security.any</b> | The keywords prefixed by “security.” are reserved for future standardization.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 102478 | <b>size</b>         | The size of the file in octets, expressed as a decimal number using digits from the ISO/IEC 646:1991 standard. This record shall override the <i>size</i> field in the following header block(s). When used in <b>write</b> or <b>copy</b> mode, <i>pax</i> shall include a <i>size</i> extended header record for each file with a size value greater than 8 589 934 591 (octal 77 777 777 777).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102479 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102480 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102481 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102482 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102483 | <b>uid</b>          | The user ID of the file owner, expressed as a decimal number using digits from the ISO/IEC 646:1991 standard. This record shall override the <i>uid</i> field in the following header block(s). When used in <b>write</b> or <b>copy</b> mode, <i>pax</i> shall include a <i>uid</i> extended header record for each file whose owner ID is greater than 2 097 151 (octal 7 777 777).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102484 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102485 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102486 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102487 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102488 | <b>uname</b>        | The owner of the following file(s), formatted as a user name in the user database. This record shall override the <i>uid</i> and <i>uname</i> fields in the following header block(s), and any <i>uid</i> extended header record. When used in <b>read</b> , <b>copy</b> , or <b>list</b> mode, <i>pax</i> shall translate the name from the encoding in the header record to the character set appropriate for the user database on the receiving system. If any of the characters                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 102489 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102490 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102491 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 102492 |                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |

cannot be translated, and if neither the **-oinvalid=UTF-8** option nor the **-oinvalid=binary** option is specified, the results are implementation-defined. When used in **write** or **copy** mode, *pax* shall include a **uname** extended header record for each file whose user name cannot be represented entirely with the letters and digits of the portable character set.

If the *<value>* field is zero length, it shall delete any header block field, previously entered extended header value, or global extended header value of the same name.

If a keyword in an extended header record (or in a **-o** option-argument) overrides or deletes a corresponding field in the **ustar** header block, *pax* shall ignore the contents of that header block field.

Unlike the **ustar** header block fields, NULs shall not delimit *<value>*s; all characters within the *<value>* field shall be considered data for the field. None of the length limitations of the **ustar** header block fields in Table 4-14 (on page 3075) shall apply to the extended header records.

### pax Extended Header Keyword Precedence

This section describes the precedence in which the various header records and fields and command line options are selected to apply to a file in the archive. When *pax* is used in **read** or **list** modes, it shall determine a file attribute in the following sequence:

1. If **-odelete=keyword-prefix** is used, the affected attributes shall be determined from step 7., if applicable, or ignored otherwise.
2. If **-okeyword:=** is used, the affected attributes shall be ignored.
3. If **-okeyword:=value** is used, the affected attribute shall be assigned the value.
4. If there is a *typeflag* **x** extended header record, the affected attribute shall be assigned the *<value>*. When extended header records conflict, the last one given in the header shall take precedence.
5. If **-okeyword=value** is used, the affected attribute shall be assigned the value.
6. If there is a *typeflag* **g** global extended header record, the affected attribute shall be assigned the *<value>*. When global extended header records conflict, the last one given in the global header shall take precedence.
7. Otherwise, the attribute shall be determined from the **ustar** header block.

### pax Extended Header File Times

The *pax* utility shall write an **mtime** record for each file in **write** or **copy** modes if the file's modification time cannot be represented exactly in the **ustar** header logical record described in *ustar Interchange Format* (on page 3075). This can occur if the time is out of **ustar** range, or if the file system of the underlying implementation supports non-integer time granularities and the time is not an integer. All of these time records shall be formatted as a decimal representation of the time in seconds since the Epoch. If a *<period>* ( ' . ' ) decimal point character is present, the digits to the right of the point shall represent the units of a subsecond timing granularity, where the first digit is tenths of a second and each subsequent digit is a tenth of the previous digit. In **read** or **copy** mode, the *pax* utility shall truncate the time of a file to the greatest value that is not greater than the input header file time. In **write** or **copy** mode, the *pax* utility shall output a time exactly if it can be represented exactly as a decimal number, and otherwise shall generate only enough digits so that the same time shall be recovered if the file is extracted on a system whose underlying implementation supports the same time granularity.



**ustar Interchange Format**

A **ustar** archive tape or file shall contain a series of logical records. Each logical record shall be a fixed-size logical record of 512 octets (see below). Although this format may be thought of as being stored on 9-track industry-standard 12.7 mm (0.5 in) magnetic tape, other types of transportable media are not excluded. Each file archived shall be represented by a header logical record that describes the file, followed by zero or more logical records that give the contents of the file. At the end of the archive file there shall be two 512-octet logical records filled with binary zeros, interpreted as an end-of-archive indicator.

The logical records may be grouped for physical I/O operations, as described under the **-bblocksize** and **-x ustar** options. Each group of logical records may be written with a single operation equivalent to the *write()* function. On magnetic tape, the result of this write shall be a single tape physical block. The last physical block shall always be the full size, so logical records after the two zero logical records may contain undefined data.

The header logical record shall be structured as shown in the following table. All lengths and offsets are in decimal.

**Table 4-14** ustar Header Block

| Field Name      | Octet Offset | Length (in Octets) |
|-----------------|--------------|--------------------|
| <i>name</i>     | 0            | 100                |
| <i>mode</i>     | 100          | 8                  |
| <i>uid</i>      | 108          | 8                  |
| <i>gid</i>      | 116          | 8                  |
| <i>size</i>     | 124          | 12                 |
| <i>mtime</i>    | 136          | 12                 |
| <i>chksum</i>   | 148          | 8                  |
| <i>typeflag</i> | 156          | 1                  |
| <i>linkname</i> | 157          | 100                |
| <i>magic</i>    | 257          | 6                  |
| <i>version</i>  | 263          | 2                  |
| <i>uname</i>    | 265          | 32                 |
| <i>gname</i>    | 297          | 32                 |
| <i>devmajor</i> | 329          | 8                  |
| <i>devminor</i> | 337          | 8                  |
| <i>prefix</i>   | 345          | 155                |

All characters in the header logical record shall be represented in the coded character set of the ISO/IEC 646: 1991 standard. For maximum portability between implementations, names should be selected from characters represented by the portable filename character set as octets with the most significant bit zero. If an implementation supports the use of characters outside of <slash> and the portable filename character set in names for files, users, and groups, one or more implementation-defined encodings of these characters shall be provided for interchange purposes.

However, the *pax* utility shall never create filenames on the local system that cannot be accessed via the procedures described in POSIX.1-2008. If a filename is found on the medium that would create an invalid filename, it is implementation-defined whether the data from the file is stored on the file hierarchy and under what name it is stored. The *pax* utility may choose to ignore these files as long as it produces an error indicating that the file is being ignored.

Each field within the header logical record is contiguous; that is, there is no padding used. Each

character on the archive medium shall be stored contiguously.

The fields *magic*, *uname*, and *gname* are character strings each terminated by a NUL character. The fields *name*, *linkname*, and *prefix* are NUL-terminated character strings except when all characters in the array contain non-NUL characters including the last character. The *version* field is two octets containing the characters "00" (zero-zero). The *typeflag* contains a single character. All other fields are leading zero-filled octal numbers using digits from the ISO/IEC 646:1991 standard IRV. Each numeric field is terminated by one or more <space> or NUL characters.

The *name* and the *prefix* fields shall produce the pathname of the file. A new pathname shall be formed, if *prefix* is not an empty string (its first character is not NUL), by concatenating *prefix* (up to the first NUL character), a <slash> character, and *name*; otherwise, *name* is used alone. In either case, *name* is terminated at the first NUL character. If *prefix* begins with a NUL character, it shall be ignored. In this manner, pathnames of at most 256 characters can be supported. If a pathname does not fit in the space provided, *pax* shall notify the user of the error, and shall not store any part of the file—header or data—on the medium.

The *linkname* field, described below, shall not use the *prefix* to produce a pathname. As such, a *linkname* is limited to 100 characters. If the name does not fit in the space provided, *pax* shall notify the user of the error, and shall not attempt to store the link on the medium.

The *mode* field provides 12 bits encoded in the ISO/IEC 646:1991 standard octal digit representation. The encoded bits shall represent the following values:

**Table 4-15** *ustar mode* Field

| Bit Value | POSIX.1-2008 Bit | Description                                     |
|-----------|------------------|-------------------------------------------------|
| 04 000    | S_ISUID          | Set UID on execution.                           |
| 02 000    | S_ISGID          | Set GID on execution.                           |
| 01 000    | <reserved>       | Reserved for future standardization.            |
| 00 400    | S_IRUSR          | Read permission for file owner class.           |
| 00 200    | S_IWUSR          | Write permission for file owner class.          |
| 00 100    | S_IXUSR          | Execute/search permission for file owner class. |
| 00 040    | S_IRGRP          | Read permission for file group class.           |
| 00 020    | S_IWGRP          | Write permission for file group class.          |
| 00 010    | S_IXGRP          | Execute/search permission for file group class. |
| 00 004    | S_IROTH          | Read permission for file other class.           |
| 00 002    | S_IWOTH          | Write permission for file other class.          |
| 00 001    | S_IXOTH          | Execute/search permission for file other class. |

When appropriate privileges are required to set one of these mode bits, and the user restoring the files from the archive does not have appropriate privileges, the mode bits for which the user does not have appropriate privileges shall be ignored. Some of the mode bits in the archive format are not mentioned elsewhere in this volume of POSIX.1-2008. If the implementation does not support those bits, they may be ignored.

The *uid* and *gid* fields are the user and group ID of the owner and group of the file, respectively.

The *size* field is the size of the file in octets. If the *typeflag* field is set to specify a file to be of type 1 (a link) or 2 (a symbolic link), the *size* field shall be specified as zero. If the *typeflag* field is set to specify a file of type 5 (directory), the *size* field shall be interpreted as described under the definition of that record type. No data logical records are stored for types 1, 2, or 5. If the *typeflag* field is set to 3 (character special file), 4 (block special file), or 6 (FIFO), the meaning of the *size* field is unspecified by this volume of POSIX.1-2008, and no data logical records shall be stored on the medium. Additionally, for type 6, the *size* field shall be ignored when reading. If the



- 102628        *typeflag* field is set to any other value, the number of logical records written following the header  
 102629        shall be  $(size+511)/512$ , ignoring any fraction in the result of the division.
- 102630        The *mtime* field shall be the modification time of the file at the time it was archived. It is the  
 102631        ISO/IEC 646:1991 standard representation of the octal value of the modification time obtained  
 102632        from the *stat()* function.
- 102633        The *chksum* field shall be the ISO/IEC 646:1991 standard IRV representation of the octal value of  
 102634        the simple sum of all octets in the header logical record. Each octet in the header shall be treated  
 102635        as an unsigned value. These values shall be added to an unsigned integer, initialized to zero, the  
 102636        precision of which is not less than 17 bits. When calculating the checksum, the *chksum* field is  
 102637        treated as if it were all <space> characters.
- 102638        The *typeflag* field specifies the type of file archived. If a particular implementation does not  
 102639        recognize the type, or the user does not have appropriate privileges to create that type, the file  
 102640        shall be extracted as if it were a regular file if the file type is defined to have a meaning for the  
 102641        *size* field that could cause data logical records to be written on the medium (see the previous  
 102642        description for *size*). If conversion to a regular file occurs, the *pax* utility shall produce an error  
 102643        indicating that the conversion took place. All of the *typeflag* fields shall be coded in the  
 102644        ISO/IEC 646:1991 standard IRV:
- |        |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
|--------|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 102645 | 0    | Represents a regular file. For backwards-compatibility, a <i>typeflag</i> value of binary zero (' $\backslash 0$ ') should be recognized as meaning a regular file when extracting files from the archive. Archives written with this version of the archive file format create regular files with a <i>typeflag</i> value of the ISO/IEC 646:1991 standard IRV ' $\backslash 0$ '.                                        |
| 102646 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102647 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102648 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102649 | 1    | Represents a file linked to another file, of any type, previously archived. Such files are identified by having the same device and file serial numbers, and pathnames that refer to different directory entries. All such files shall be archived as linked files. The linked-to name is specified in the <i>linkname</i> field with a NUL-character terminator if it is less than 100 octets in length.                  |
| 102650 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102651 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102652 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102653 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102654 | 2    | Represents a symbolic link. The contents of the symbolic link shall be stored in the <i>linkname</i> field.                                                                                                                                                                                                                                                                                                                |
| 102655 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102656 | 3, 4 | Represent character special files and block special files respectively. In this case the <i>devmajor</i> and <i>devminor</i> fields shall contain information defining the device, the format of which is unspecified by this volume of POSIX.1-2008. Implementations may map the device specifications to their own local specification or may ignore the entry.                                                          |
| 102657 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102658 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102659 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102660 | 5    | Specifies a directory or subdirectory. On systems where disk allocation is performed on a directory basis, the <i>size</i> field shall contain the maximum number of octets (which may be rounded to the nearest disk block allocation unit) that the directory may hold. A <i>size</i> field of zero indicates no such limiting. Systems that do not support limiting in this manner should ignore the <i>size</i> field. |
| 102661 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102662 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102663 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102664 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102665 | 6    | Specifies a FIFO special file. Note that the archiving of a FIFO file archives the existence of this file and not its contents.                                                                                                                                                                                                                                                                                            |
| 102666 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102667 | 7    | Reserved to represent a file to which an implementation has associated some high-performance attribute. Implementations without such extensions should treat this file as a regular file (type 0).                                                                                                                                                                                                                         |
| 102668 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102669 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102670 | A-Z  | The letters ' $\backslash A$ ' to ' $\backslash Z$ ', inclusive, are reserved for custom implementations. All other values are reserved for future versions of this standard.                                                                                                                                                                                                                                              |
| 102671 |      |                                                                                                                                                                                                                                                                                                                                                                                                                            |
- 102672        It is unspecified whether files with pathnames that refer to the same directory entry are archived  
 102673        as linked files or as separate files. If they are archived as linked files, this means that attempting

to extract both pathnames from the resulting archive will always cause an error (unless the **-u** option is used) because the link cannot be created.

It is unspecified whether files with the same device and file serial numbers being appended to an archive are treated as linked files to members that were in the archive before the append.

Attempts to archive a socket shall produce a diagnostic message when **ustar** interchange format is used, but may be allowed when **pax** interchange format is used. Handling of other file types is implementation-defined.

The *magic* field is the specification that this archive was output in this archive format. If this field contains **ustar** (the five characters from the ISO/IEC 646:1991 standard IRV shown followed by NUL), the *uname* and *gname* fields shall contain the ISO/IEC 646:1991 standard IRV representation of the owner and group of the file, respectively (truncated to fit, if necessary). When the file is restored by a privileged, protection-preserving version of the utility, the user and group databases shall be scanned for these names. If found, the user and group IDs contained within these files shall be used rather than the values contained within the *uid* and *gid* fields.

### cpio Interchange Format

The octet-oriented **cpio** archive format shall be a series of entries, each comprising a header that describes the file, the name of the file, and then the contents of the file.

An archive may be recorded as a series of fixed-size blocks of octets. This blocking shall be used only to make physical I/O more efficient. The last group of blocks shall always be at the full size.

For the octet-oriented **cpio** archive format, the individual entry information shall be in the order indicated and described by the following table; see also the **<cpio.h>** header.

**Table 4-16** Octet-Oriented cpio Archive Entry

| Header Field Name    | Length (in Octets) | Interpreted as  |
|----------------------|--------------------|-----------------|
| <i>c_magic</i>       | 6                  | Octal number    |
| <i>c_dev</i>         | 6                  | Octal number    |
| <i>c_ino</i>         | 6                  | Octal number    |
| <i>c_mode</i>        | 6                  | Octal number    |
| <i>c_uid</i>         | 6                  | Octal number    |
| <i>c_gid</i>         | 6                  | Octal number    |
| <i>c_nlink</i>       | 6                  | Octal number    |
| <i>c_rdev</i>        | 6                  | Octal number    |
| <i>c_mtime</i>       | 11                 | Octal number    |
| <i>c_namesize</i>    | 6                  | Octal number    |
| <i>c_filesize</i>    | 11                 | Octal number    |
| Filename Field Name  | Length             | Interpreted as  |
| <i>c_name</i>        | <i>c_namesize</i>  | Pathname string |
| File Data Field Name | Length             | Interpreted as  |
| <i>c_filedata</i>    | <i>c_filesize</i>  | Data            |

**cpio Header**

For each file in the archive, a header as defined previously shall be written. The information in the header fields is written as streams of the ISO/IEC 646:1991 standard characters interpreted as octal numbers. The octal numbers shall be extended to the necessary length by appending the ISO/IEC 646:1991 standard IRV zeros at the most-significant-digit end of the number; the result is written to the most-significant digit of the stream of octets first. The fields shall be interpreted as follows:

- c\_magic* Identify the archive as being a transportable archive by containing the identifying value "070707".
- c\_dev, c\_ino* Contains values that uniquely identify the file within the archive (that is, no files contain the same pair of *c\_dev* and *c\_ino* values unless they are links to the same file). The values shall be determined in an unspecified manner.
- c\_mode* Contains the file type and access permissions as defined in the following table.

**Table 4-17** Values for cpio *c\_mode* Field

| File Permissions Name | Value    | Indicates              |
|-----------------------|----------|------------------------|
| C_IRUSR               | 000 400  | Read by owner          |
| C_IWUSR               | 000 200  | Write by owner         |
| C_IXUSR               | 000 100  | Execute by owner       |
| C_IRGRP               | 000 040  | Read by group          |
| C_IWGRP               | 000 020  | Write by group         |
| C_IXGRP               | 000 010  | Execute by group       |
| C_IROTH               | 000 004  | Read by others         |
| C_IWOTH               | 000 002  | Write by others        |
| C_IXOTH               | 000 001  | Execute by others      |
| C_ISUID               | 004 000  | Set <i>uid</i>         |
| C_ISGID               | 002 000  | Set <i>gid</i>         |
| C_ISVTX               | 001 000  | Reserved               |
| File Type Name        | Value    | Indicates              |
| C_ISDIR               | 040 000  | Directory              |
| C_ISFIFO              | 010 000  | FIFO                   |
| C_ISREG               | 0100 000 | Regular file           |
| C_ISLNK               | 0120 000 | Symbolic link          |
| C_ISBLK               | 060 000  | Block special file     |
| C_ISCHR               | 020 000  | Character special file |
| C_ISSOCK              | 0140 000 | Socket                 |
| C_ISCTG               | 0110 000 | Reserved               |

Directories, FIFOs, symbolic links, and regular files shall be supported on a system conforming to this volume of POSIX.1-2008; additional values defined previously are reserved for compatibility with existing systems. Additional file types may be supported; however, such files should not be written to archives intended to be transported to other systems.

- c\_uid* Contains the user ID of the owner.

|        |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------|-------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 102756 | <i>c_gid</i>      | Contains the group ID of the group.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 102757 | <i>c_nlink</i>    | Contains a number greater than or equal to the number of links in the archive referencing the file. If the <i>-a</i> option is used to append to a <i>cpio</i> archive, then the <i>pax</i> utility need not account for the files in the existing part of the archive when calculating the <i>c_nlink</i> values for the appended part of the archive, and need not alter the <i>c_nlink</i> values in the existing part of the archive if additional files with the same <i>c_dev</i> and <i>c_ino</i> values are appended to the archive. |
| 102758 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 102759 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 102760 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 102761 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 102762 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 102763 | <i>c_rdev</i>     | Contains implementation-defined information for character or block special files.                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 102764 | <i>c_mtime</i>    | Contains the latest time of modification of the file at the time the archive was created.                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 102765 |                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 102766 | <i>c_namesize</i> | Contains the length of the pathname, including the terminating NUL character.                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 102767 | <i>c_filesiz</i>  | Contains the length in octets of the data section following the header structure.                                                                                                                                                                                                                                                                                                                                                                                                                                                            |

## 102768 **cpio Filename**

102769 The *c\_name* field shall contain the pathname of the file. The length of this field in octets is the  
102770 value of *c\_namesize*.

102771 If a filename is found on the medium that would create an invalid pathname, it is  
102772 implementation-defined whether the data from the file is stored on the file hierarchy and under  
102773 what name it is stored.

102774 All characters shall be represented in the ISO/IEC 646:1991 standard IRV. For maximum  
102775 portability between implementations, names should be selected from characters represented by  
102776 the portable filename character set as octets with the most significant bit zero. If an  
102777 implementation supports the use of characters outside the portable filename character set in  
102778 names for files, users, and groups, one or more implementation-defined encodings of these  
102779 characters shall be provided for interchange purposes. However, the *pax* utility shall never create  
102780 filenames on the local system that cannot be accessed via the procedures described previously in  
102781 this volume of POSIX.1-2008. If a filename is found on the medium that would create an invalid  
102782 filename, it is implementation-defined whether the data from the file is stored on the local file  
102783 system and under what name it is stored. The *pax* utility may choose to ignore these files as long  
102784 as it produces an error indicating that the file is being ignored.

## 102785 **cpio File Data**

102786 Following *c\_name*, there shall be *c\_filesiz* octets of data. Interpretation of such data occurs in a  
102787 manner dependent on the file. For regular files, the data shall consist of the contents of the file.  
102788 For symbolic links, the data shall consist of the contents of the symbolic link. If *c\_filesiz* is zero,  
102789 no data shall be contained in *c\_filedata*.

102790 When restoring from an archive:

- 102791 • If the user does not have appropriate privileges to create a file of the specified type, *pax*  
102792 shall ignore the entry and write an error message to standard error.
- 102793 • Only regular files and symbolic links have data to be restored. Presuming a regular file  
102794 meets any selection criteria that might be imposed on the format-reading utility by the  
102795 user, such data shall be restored.
- 102796 • If a user does not have appropriate privileges to set a particular mode flag, the flag shall be  
102797 ignored. Some of the mode flags in the archive format are not mentioned elsewhere in this  
102798 volume of POSIX.1-2008. If the implementation does not support those flags, they may be

102799 ignored.

## 102800 **cpio Special Entries**

102801 FIFO special files, directories, and the trailer shall be recorded with *c\_filesize* equal to zero.  
102802 Symbolic links shall be recorded with *c\_filesize* equal to the length of the contents of the symbolic  
102803 link. For other special files, *c\_filesize* is unspecified by this volume of POSIX.1-2008. The header  
102804 for the next file entry in the archive shall be written directly after the last octet of the file entry  
102805 preceding it. A header denoting the filename **TRAILER!!!** shall indicate the end of the archive;  
102806 the contents of octets in the last block of the archive following such a header are undefined.

## 102807 **EXIT STATUS**

102808 The following exit values shall be returned:

102809 0 All files were processed successfully.

102810 >0 An error occurred.

## 102811 **CONSEQUENCES OF ERRORS**

102812 If *pax* cannot create a file or a link when reading an archive or cannot find a file when writing an  
102813 archive, or cannot preserve the user ID, group ID, or file mode when the **-p** option is specified, a  
102814 diagnostic message shall be written to standard error and a non-zero exit status shall be  
102815 returned, but processing shall continue. In the case where *pax* cannot create a link to a file, *pax*  
102816 shall not, by default, create a second copy of the file.

102817 If the extraction of a file from an archive is prematurely terminated by a signal or error, *pax* may  
102818 have only partially extracted the file or (if the **-n** option was not specified) may have extracted a  
102819 file of the same name as that specified by the user, but which is not the file the user wanted.  
102820 Additionally, the file modes of extracted directories may have additional bits from the S\_IRWXU  
102821 mask set as well as incorrect modification and access times.

## 102822 **APPLICATION USAGE**

102823 Caution is advised when using the **-a** option to append to a *cpio* format archive. If any of the  
102824 files being appended happen to be given the same *c\_dev* and *c\_ino* values as a file in the existing  
102825 part of the archive, then they may be treated as links to that file on extraction. Thus, it is risky to  
102826 use **-a** with *cpio* format except when it is done on the same system that the original archive was  
102827 created on, and with the same *pax* utility, and in the knowledge that there has been little or no  
102828 file system activity since the original archive was created that could lead to any of the files  
102829 appended being given the same *c\_dev* and *c\_ino* values as an unrelated file in the existing part of  
102830 the archive. Also, when (intentionally) appending additional links to a file in the existing part of  
102831 the archive, the *c\_nlink* values in the modified archive can be smaller than the number of links to  
102832 the file in the archive, which may mean that the links are not preserved on extraction.

102833 The **-p** (privileges) option was invented to reconcile differences between historical *tar* and *cpio*  
102834 implementations. In particular, the two utilities use **-m** in diametrically opposed ways. The **-p**  
102835 option also provides a consistent means of extending the ways in which future file attributes can  
102836 be addressed, such as for enhanced security systems or high-performance files. Although it may  
102837 seem complex, there are really two modes that are most commonly used:

102838 **-p e** “Preserve everything”. This would be used by the historical superuser, someone with  
102839 all appropriate privileges, to preserve all aspects of the files as they are recorded in the  
102840 archive. The **e** flag is the sum of **o** and **p**, and other implementation-defined attributes.

102841 **-p p** “Preserve” the file mode bits. This would be used by the user with regular privileges  
102842 who wished to preserve aspects of the file other than the ownership. The file times are  
102843 preserved by default, but two other flags are offered to disable these and use the time  
102844 of extraction.

The one pathname per line format of standard input precludes pathnames containing <newline> characters. Although such pathnames violate the portable filename guidelines, they may exist and their presence may inhibit usage of *pax* within shell scripts. This problem is inherited from historical archive programs. The problem can be avoided by listing filename arguments on the command line instead of on standard input.

It is almost certain that appropriate privileges are required for *pax* to accomplish parts of this volume of POSIX.1-2008. Specifically, creating files of type block special or character special, restoring file access times unless the files are owned by the user (the **-t** option), or preserving file owner, group, and mode (the **-p** option) all probably require appropriate privileges.

In **read** mode, implementations are permitted to overwrite files when the archive has multiple members with the same name. This may fail if permissions on the first version of the file do not permit it to be overwritten.

The **cpio** and **ustar** formats can only support files up to 8 589 934 592 bytes ( $8 * 2^{30}$ ) in size.

When archives containing binary header information are listed, the filenames printed may cause strange behavior on some terminals.

When all of the following are true:

1. A file of type directory is being placed into an archive.
2. The **ustar** archive format is being used.
3. The pathname of the directory is less than or equal to 155 bytes long (it will fit in the *prefix* field in the **ustar** header block).
4. The last component of the pathname of the directory is longer than 100 bytes long (it will not fit in the *name* field in the **ustar** header block).

some implementations of the *pax* utility will place the entire directory pathname in the *prefix* field, set the *name* field to an empty string, and place the directory in the archive. Other implementations of the *pax* utility will give an error under these conditions because the *name* field is not large enough to hold the last component of the directory name. This standard allows either behavior. However, when extracting a directory from a **ustar** format archive, this standard requires that all implementations be able to extract a directory even if the *name* field contains an empty string as long as the *prefix* field does not also contain an empty string.

## EXAMPLES

The following command:

```
pax -w -f /dev/rmt/lm .
```

copies the contents of the current directory to tape drive 1, medium density (assuming historical System V device naming procedures—the historical BSD device name would be **/dev/rmt9**).

The following commands:

```
mkdir newdir
```

```
pax -rw olddir newdir
```

copy the *olddir* directory hierarchy to *newdir*.

```
pax -r -s ',^//*usr//*,,' -f a.pax
```

reads the archive **a.pax**, with all files rooted in **/usr** in the archive extracted relative to the current directory.

Using the option:



```

102887 -o listopt="%M %(atime)T %(size)D %(name)s"
102888 overrides the default output description in Standard Output and instead writes:
102889 -rw-rw--- Jan 12 15:53 2003 1492 /usr/foo/bar
102890 Using the options:
102891 -o listopt='%L\t%(size)D\n%.7' \
102892 -o listopt='(name)s\n%(atime)T\n%T'
102893 overrides the default output description in Standard Output and instead writes:
102894 /usr/foo/bar -> /tmp 1492
102895 /usr/fo
102896 Jan 12 15:53 1991
102897 Jan 31 15:53 2003

```

## 102898 RATIONALE

102899 The *pax* utility was new for the ISO POSIX-2:1993 standard. It represents a peaceful compromise  
 102900 between advocates of the historical *tar* and *cpio* utilities.

102901 A fundamental difference between *cpio* and *tar* was in the way directories were treated. The *cpio*  
 102902 utility did not treat directories differently from other files, and to select a directory and its  
 102903 contents required that each file in the hierarchy be explicitly specified. For *tar*, a directory  
 102904 matched every file in the file hierarchy it rooted.

102905 The *pax* utility offers both interfaces; by default, directories map into the file hierarchy they root.  
 102906 The **-d** option causes *pax* to skip any file not explicitly referenced, as *cpio* historically did. The *tar*  
 102907 **-style** behavior was chosen as the default because it was believed that this was the more  
 102908 common usage and because *tar* is the more commonly available interface, as it was historically  
 102909 provided on both System V and BSD implementations.

102910 The data interchange format specification in this volume of POSIX.1-2008 requires that processes  
 102911 with “appropriate privileges” shall always restore the ownership and permissions of extracted  
 102912 files exactly as archived. If viewed from the historic equivalence between superuser and  
 102913 “appropriate privileges”, there are two problems with this requirement. First, users running as  
 102914 superusers may unknowingly set dangerous permissions on extracted files. Second, it is  
 102915 needlessly limiting, in that superusers cannot extract files and own them as superuser unless the  
 102916 archive was created by the superuser. (It should be noted that restoration of ownerships and  
 102917 permissions for the superuser, by default, is historical practice in *cpio*, but not in *tar*.) In order to  
 102918 avoid these two problems, the *pax* specification has an additional “privilege” mechanism, the **-p**  
 102919 option. Only a *pax* invocation with the privileges needed, and which has the **-p** option set using  
 102920 the **e** specification character, has appropriate privileges to restore full ownership and permission  
 102921 information.

102922 Note also that this volume of POSIX.1-2008 requires that the file ownership and access  
 102923 permissions shall be set, on extraction, in the same fashion as the *creat()* function when provided  
 102924 with the mode stored in the archive. This means that the file creation mask of the user is applied  
 102925 to the file permissions.

102926 Users should note that directories may be created by *pax* while extracting files with permissions  
 102927 that are different from those that existed at the time the archive was created. When extracting  
 102928 sensitive information into a directory hierarchy that no longer exists, users are encouraged to set  
 102929 their file creation mask appropriately to protect these files during extraction.

102930 The table of contents output is written to standard output to facilitate pipeline processing.

102931 An early proposal had hard links displaying for all pathnames. This was removed because it

complicates the output of the case where `-v` is not specified and does not match historical *cpio* usage. The hard-link information is available in the `-v` display.

The description of the `-l` option allows implementations to make hard links to symbolic links. Earlier versions of this standard did not specify any way to create a hard link to a symbolic link, but many implementations provided this capability as an extension. If there are hard links to symbolic links when an archive is created, the implementation is required to archive the hard link in the archive (unless `-H` or `-L` is specified). When in **read** mode and in **copy** mode, implementations supporting hard links to symbolic links should use them when appropriate.

The archive formats inherited from the POSIX.1-1990 standard have certain restrictions that have been brought along from historical usage. For example, there are restrictions on the length of pathnames stored in the archive. When *pax* is used in **copy(-rw)** mode (copying directory hierarchies), the ability to use extensions from the **-xpax** format overcomes these restrictions.

The default *blocksize* value of 5120 bytes for *cpio* was selected because it is one of the standard block-size values for *cpio*, set when the `-B` option is specified. (The other default block-size value for *cpio* is 512 bytes, and this was considered to be too small.) The default block value of 10240 bytes for *tar* was selected because that is the standard block-size value for BSD *tar*. The maximum block size of 32768 bytes ( $2^{15}$ –512 bytes) is the largest multiple of 512 bytes that fits into a signed 16-bit tape controller transfer register. There are known limitations in some historical systems that would prevent larger blocks from being accepted. Historical values were chosen to improve compatibility with historical scripts using *dd* or similar utilities to manipulate archives. Also, default block sizes for any file type other than character special file has been deleted from this volume of POSIX.1-2008 as unimportant and not likely to affect the structure of the resulting archive.

Implementations are permitted to modify the block-size value based on the archive format or the device to which the archive is being written. This is to provide implementations with the opportunity to take advantage of special types of devices, and it should not be used without a great deal of consideration as it almost certainly decreases archive portability.

The intended use of the `-n` option was to permit extraction of one or more files from the archive without processing the entire archive. This was viewed by the standard developers as offering significant performance advantages over historical implementations. The `-n` option in early proposals had three effects; the first was to cause special characters in patterns to not be treated specially. The second was to cause only the first file that matched a pattern to be extracted. The third was to cause *pax* to write a diagnostic message to standard error when no file was found matching a specified pattern. Only the second behavior is retained by this volume of POSIX.1-2008, for many reasons. First, it is in general not acceptable for a single option to have multiple effects. Second, the ability to make pattern matching characters act as normal characters is useful for parts of *pax* other than file extraction. Third, a finer degree of control over the special characters is useful because users may wish to normalize only a single special character in a single filename. Fourth, given a more general escape mechanism, the previous behavior of the `-n` option can be easily obtained using the `-s` option or a *sed* script. Finally, writing a diagnostic message when a pattern specified by the user is unmatched by any file is useful behavior in all cases.

In this version, the `-n` was removed from the **copy** mode synopsis of *pax*; it is inapplicable because there are no pattern operands specified in this mode.

There is another method than *pax* for copying subtrees in POSIX.1-2008 described as part of the *cp* utility. Both methods are historical practice: *cp* provides a simpler, more intuitive interface, while *pax* offers a finer granularity of control. Each provides additional functionality to the other; in particular, *pax* maintains the hard-link structure of the hierarchy while *cp* does not. It is



the intention of the standard developers that the results be similar (using appropriate option combinations in both utilities). The results are not required to be identical; there seemed insufficient gain to applications to balance the difficulty of implementations having to guarantee that the results would be exactly identical.

A single archive may span more than one file. It is suggested that implementations provide informative messages to the user on standard error whenever the archive file is changed.

The **-d** option (do not create intermediate directories not listed in the archive) found in early proposals was originally provided as a complement to the historic **-d** option of *cpio*. It has been deleted.

The **-s** option in early proposals specified a subset of the substitution command from the *ed* utility. As there was no reason for only a subset to be supported, the **-s** option is now compatible with the current *ed* specification. Since the delimiter can be any non-null character, the following usage with single `<space>` characters is valid:

```
pax -s " foo bar " ...
```

The **-t** description is worded so as to note that this may cause the access time update caused by some other activity (which occurs while the file is being read) to be overwritten.

The default behavior of *pax* with regard to file modification times is the same as historical implementations of *tar*. It is not the historical behavior of *cpio*.

Because the **-i** option uses **/dev/tty**, utilities without a controlling terminal are not able to use this option.

The **-y** option, found in early proposals, has been deleted because a line containing a single `<period>` for the **-i** option has equivalent functionality. The special lines for the **-i** option (a single `<period>` and the empty line) are historical practice in *cpio*.

In early drafts, a **-echarmap** option was included to increase portability of files between systems using different coded character sets. This option was omitted because it was apparent that consensus could not be formed for it. In this version, the use of UTF-8 should be an adequate substitute.

The ISO POSIX-2:1993 standard and ISO POSIX-1 standard requirements for *pax*, however, made it very difficult to create a single archive containing files created using extended characters provided by different locales. This version adds the **hdrcharset** keyword to make it possible to archive files in these cases without dropping files due to translation errors.

Translating filenames and other attributes from a locale's encoding to UTF-8 and then back again can lose information, as the resulting filename might not be byte-for-byte equivalent to the original. To avoid this problem, users can specify the **-o hdrcharset=binary** option, which will cause the resulting archive to use binary format for all names and attributes. Such archives are not portable among hosts that use different native encodings (e.g., EBCDIC *versus* ASCII-based encodings), but they will allow interchange among the vast majority of POSIX file systems in practical use. Also, the **-o hdrcharset=binary** option will cause *pax* in **copy** mode to behave more like other standard utilities such as *cp*.

If the values specified by the **-o exthdr.name=value**, **-o globexthdr.name=value**, or by **\$TMPDIR** (if **-o globexthdr.name** is not specified) require a character encoding other than that described in the ISO/IEC 646:1991 standard, a **path** extended header record will have to be created for the file. If a **hdrcharset** extended header record is active for such headers, it will determine the codeset used for the value field in these extended **path** header records. These **path** extended header records always need to be created when writing an archive even if **hdrcharset=binary** has been specified and would contain the same (binary) data that appears in

the **ustar** header record prefix and *name* fields. (In other words, an extended header **path** record is always required to be generated if the *prefix* or *name* fields contain non-ASCII characters even when **hdrcharset=binary** is also in effect for that file.)

The **-k** option was added to address international concerns about the dangers involved in the character set transformations of **-e** (if the target character set were different from the source, the filenames might be transformed into names matching existing files) and also was made more general to protect files transferred between file systems with different {NAME\_MAX} values (truncating a filename on a smaller system might also inadvertently overwrite existing files). As stated, it prevents any overwriting, even if the target file is older than the source. This version adds more granularity of options to solve this problem by introducing the **-oinvalid=option**—specifically the **UTF-8** and **binary** actions. (Note that an existing file is still subject to overwriting in this case. The **-k** option closes that loophole.)

Some of the file characteristics referenced in this volume of POSIX.1-2008 might not be supported by some archive formats. For example, neither the **tar** nor **cpio** formats contain the file access time. For this reason, the *e* specification character has been provided, intended to cause all file characteristics specified in the archive to be retained.

It is required that extracted directories, by default, have their access and modification times and permissions set to the values specified in the archive. This has obvious problems in that the directories are almost certainly modified after being extracted and that directory permissions may not permit file creation. One possible solution is to create directories with the mode specified in the archive, as modified by the *umask* of the user, with sufficient permissions to allow file creation. After all files have been extracted, *pax* would then reset the access and modification times and permissions as necessary.

The list-mode formatting description borrows heavily from the one defined by the *printf* utility. However, since there is no separate operand list to get conversion arguments, the format was extended to allow specifying the name of the conversion argument as part of the conversion specification.

The *T* conversion specifier allows time fields to be displayed in any of the date formats. Unlike the *ls* utility, *pax* does not adjust the format when the date is less than six months in the past. This makes parsing the output more predictable.

The *D* conversion specifier handles the ability to display the major/minor or file size, as with *ls*, by using *%-8(size)D*.

The *L* conversion specifier handles the *ls* display for symbolic links.

Conversion specifiers were added to generate existing known types used for *ls*.

### pax Interchange Format

The new POSIX data interchange format was developed primarily to satisfy international concerns that the **ustar** and **cpio** formats did not provide for file, user, and group names encoded in characters outside a subset of the ISO/IEC 646:1991 standard. The standard developers realized that this new POSIX data interchange format should be very extensible because there were other requirements they foresaw in the near future:

- Support international character encodings and locale information
- Support security information (ACLs, and so on)
- Support future file types, such as realtime or contiguous files

- Include data areas for implementation use
- Support systems with words larger than 32 bits and timers with subsecond granularity

The following were not goals for this format because these are better handled by separate utilities or are inappropriate for a portable format:

- Encryption
- Compression
- Data translation between locales and codesets
- *inode* storage

The format chosen to support the goals is an extension of the **ustar** format. Of the two formats previously available, only the **ustar** format was selected for extensions because:

- It was easier to extend in an upwards-compatible way. It offered version flags and header block type fields with room for future standardization. The **cpio** format, while possessing a more flexible file naming methodology, could not be extended without breaking some theoretical implementation or using a dummy filename that could be a legitimate filename.
- Industry experience since the original “tar wars” fought in developing the ISO POSIX-1 standard has clearly been in favor of the **ustar** format, which is generally the default output format selected for *pax* implementations on new systems.

The new format was designed with one additional goal in mind: reasonable behavior when an older *tar* or *pax* utility happened to read an archive. Since the POSIX.1-1990 standard mandated that a “format-reading utility” had to treat unrecognized *typeflag* values as regular files, this allowed the format to include all the extended information in a pseudo-regular file that preceded each real file. An option is given that allows the archive creator to set up reasonable names for these files on the older systems. Also, the normative text suggests that reasonable file access values be used for this **ustar** header block. Making these header files inaccessible for convenient reading and deleting would not be reasonable. File permissions of 600 or 700 are suggested.

The **ustar** *typeflag* field was used to accommodate the additional functionality of the new format rather than magic or version because the POSIX.1-1990 standard (and, by reference, the previous version of *pax*), mandated the behavior of the format-reading utility when it encountered an unknown *typeflag*, but was silent about the other two fields.

Early proposals for the first version of this standard contained a proposed archive format that was based on compatibility with the standard for tape files (ISO 1001, similar to the format used historically on many mainframes and minicomputers). This format was overly complex and required considerable overhead in volume and header records. Furthermore, the standard developers felt that it would not be acceptable to the community of POSIX developers, so it was later changed to be a format more closely related to historical practice on POSIX systems.

The prefix and name split of pathnames in **ustar** was replaced by the single path extended header record for simplicity.

The concept of a global extended header (*typeflagg*) was controversial. If this were applied to an archive being recorded on magnetic tape, a few unreadable blocks at the beginning of the tape could be a serious problem; a utility attempting to extract as many files as possible from a damaged archive could lose a large percentage of file header information in this case. However, if the archive were on a reliable medium, such as a CD-ROM, the global extended header offers considerable potential size reductions by eliminating redundant information. Thus, the text warns against using the global method for unreliable media and provides a method for

implanting global information in the extended header for each file, rather than in the *typeflag* g records.

No facility for data translation or filtering on a per-file basis is included because the standard developers could not invent an interface that would allow this in an efficient manner. If a filter, such as encryption or compression, is to be applied to all the files, it is more efficient to apply the filter to the entire archive as a single file. The standard developers considered interfaces that would invoke a shell script for each file going into or out of the archive, but the system overhead in this approach was considered to be too high.

One such approach would be to have **filter=** records that give a pathname for an executable. When the program is invoked, the file and archive would be open for standard input/output and all the header fields would be available as environment variables or command-line arguments. The standard developers did discuss such schemes, but they were omitted from POSIX.1-2008 due to concerns about excessive overhead. Also, the program itself would need to be in the archive if it were to be used portably.

There is currently no portable means of identifying the character set(s) used for a file in the file system. Therefore, *pax* has not been given a mechanism to generate charset records automatically. The only portable means of doing this is for the user to write the archive using the **-ocharset=string** command line option. This assumes that all of the files in the archive use the same encoding. The “implementation-defined” text is included to allow for a system that can identify the encodings used for each of its files.

The table of standards that accompanies the charset record description is acknowledged to be very limited. Only a limited number of character set standards is reasonable for maximal interchange. Any character set is, of course, possible by prior agreement. It was suggested that EBCDIC be listed, but it was omitted because it is not defined by a formal standard. Formal standards, and then only those with reasonably large followings, can be included here, simply as a matter of practicality. The *<value>s* represent names of officially registered character sets in the format required by the ISO 2375:1985 standard.

The normal *<comma>* or *<blank>*-separated list rules are not followed in the case of keyword options to allow ease of argument parsing for *getopts*.

Further information on character encodings is in [pax Archive Character Set Encoding/Decoding](#) (on page 3090).

The standard developers have reserved keyword name space for vendor extensions. It is suggested that the format to be used is:

*VENDOR.keyword*

where *VENDOR* is the name of the vendor or organization in all uppercase letters. It is further suggested that the keyword following the *<period>* be named differently than any of the standard keywords so that it could be used for future standardization, if appropriate, by omitting the *VENDOR* prefix.

The *<length>* field in the extended header record was included to make it simpler to step through the records, even if a record contains an unknown format (to a particular *pax*) with complex interactions of special characters. It also provides a minor integrity checkpoint within the records to aid a program attempting to recover files from a damaged archive.

There are no extended header versions of the *devmajor* and *devminor* fields because the unspecified format **ustar** header field should be sufficient. If they are not, vendor-specific extended keywords (such as *VENDOR.devmajor*) should be used.

Device and *i*-number labeling of files was not adopted from *cpio*; files are interchanged strictly

on a symbolic name basis, as in **ustar**.

Just as with the **ustar** format descriptions, the new format makes no special arrangements for multi-volume archives. Each of the *pax* archive types is assumed to be inside a single POSIX file and splitting that file over multiple volumes (diskettes, tape cartridges, and so on), processing their labels, and mounting each in the proper sequence are considered to be implementation details that cannot be described portably.

The **pax** format is intended for interchange, not only for backup on a single (family of) systems. It is not as densely packed as might be possible for backup:

- It contains information as coded characters that could be coded in binary.
- It identifies extended records with name fields that could be omitted in favor of a fixed-field layout.
- It translates names into a portable character set and identifies locale-related information, both of which are probably unnecessary for backup.

The requirements on restoring from an archive are slightly different from the historical wording, allowing for non-monolithic privilege to bring forward as much as possible. In particular, attributes such as “high performance file” might be broadly but not universally granted while set-user-ID or *chown()* might be much more restricted. There is no implication in POSIX.1-2008 that the security information be honored after it is restored to the file hierarchy, in spite of what might be improperly inferred by the silence on that topic. That is a topic for another standard.

Links are recorded in the fashion described here because a link can be to any file type. It is desirable in general to be able to restore part of an archive selectively and restore all of those files completely. If the data is not associated with each link, it is not possible to do this. However, the data associated with a file can be large, and when selective restoration is not needed, this can be a significant burden. The archive is structured so that files that have no associated data can always be restored by the name of any link name of any link, and the user may choose whether data is recorded with each instance of a file that contains data. The format permits mixing of both types of links in a single archive; this can be done for special needs, and *pax* is expected to interpret such archives on input properly, despite the fact that there is no *pax* option that would force this mixed case on output. (When **-o linkdata** is used, the output must contain the duplicate data, but the implementation is free to include it or omit it when **-o linkdata** is not used.)

The time values are included as extended header records for those implementations needing more than the eleven octal digits allowed by the **ustar** format. Portable file timestamps cannot be negative. If *pax* encounters a file with a negative timestamp in **copy** or **write** mode, it can reject the file, substitute a non-negative timestamp, or generate a non-portable timestamp with a leading ‘-’. Even though some implementations can support finer file-time granularities than seconds, the normative text requires support only for seconds since the Epoch because the ISO POSIX-1 standard states them that way. The **ustar** format includes only *mtime*; the new format adds *atime* and *ctime* for symmetry. The *atime* access time restored to the file system will be affected by the **-p a** and **-p e** options. The *ctime* creation time (actually *inode* modification time) is described with appropriate privileges so that it can be ignored when writing to the file system. POSIX does not provide a portable means to change file creation time. Nothing is intended to prevent a non-portable implementation of *pax* from restoring the value.

The *gid*, *size*, and *uid* extended header records were included to allow expansion beyond the sizes specified in the regular *tar* header. New file system architectures are emerging that will exhaust the 12-digit size field. There are probably not many systems requiring more than 8 digits for user and group IDs, but the extended header values were included for completeness,



allowing overrides for all of the decimal values in the *tar* header.

The standard developers intended to describe the effective results of *pax* with regard to file ownerships and permissions; implementations are not restricted in timing or sequencing the restoration of such, provided the results are as specified.

Much of the text describing the extended headers refers to use in “**write** or **copy** modes”. The **copy** mode references are due to the normative text: “The effect of the copy shall be as if the copied files were written to an archive file and then subsequently extracted ...”. There is certainly no way to test whether *pax* is actually generating the extended headers in **copy** mode, but the effects must be as if it had.

### **pax Archive Character Set Encoding/Decoding**

There is a need to exchange archives of files between systems of different native codesets. Filenames, group names, and user names must be preserved to the fullest extent possible when an archive is read on the receiving platform. Translation of the contents of files is not within the scope of the *pax* utility.

There will also be the need to represent characters that are not available on the receiving platform. These unsupported characters cannot be automatically folded to the local set of characters due to the chance of collisions. This could result in overwriting previous extracted files from the archive or pre-existing files on the system.

For these reasons, the codeset used to represent characters within the extended header records of the *pax* archive must be sufficiently rich to handle all commonly used character sets. The fields requiring translation include, at a minimum, filenames, user names, group names, and link pathnames. Implementations may wish to have localized extended keywords that use non-portable characters.

The standard developers considered the following options:

- The archive creator specifies the well-defined name of the source codeset. The receiver must then recognize the codeset name and perform the appropriate translations to the destination codeset.
- The archive creator includes within the archive the character mapping table for the source codeset used to encode extended header records. The receiver must then read the character mapping table and perform the appropriate translations to the destination codeset.
- The archive creator translates the extended header records in the source codeset into a canonical form. The receiver must then perform the appropriate translations to the destination codeset.

The approach that incorporates the name of the source codeset poses the problem of codeset name registration, and makes the archive useless to *pax* archive decoders that do not recognize that codeset.

Because parts of an archive may be corrupted, the standard developers felt that including the character map of the source codeset was too fragile. The loss of this one key component could result in making the entire archive useless. (The difference between this and the global extended header decision was that the latter has a workaround—duplicating extended header records on unreliable media—but this would be too burdensome for large character set maps.)

Both of the above approaches also put an undue burden on the *pax* archive receiver to handle the cross-product of all source and destination codesets.

To simplify the translation from the source codeset to the canonical form and from the canonical form to the destination codeset, the standard developers decided that the internal representation should be a stateless encoding. A stateless encoding is one where each codepoint has the same meaning, without regard to the decoder being in a specific state. An example of a stateful encoding would be the Japanese Shift-JIS; an example of a stateless encoding would be the ISO/IEC 646: 1991 standard (equivalent to 7-bit ASCII).

For these reasons, the standard developers decided to adopt a canonical format for the representation of file information strings. The obvious, well-endorsed candidate is the ISO/IEC 10646-1: 2000 standard (based in part on Unicode), which can be used to represent the characters of virtually all standardized character sets. The standard developers initially agreed upon using UCS2 (16-bit Unicode) as the internal representation. This repertoire of characters provides a sufficiently rich set to represent all commonly-used codesets.

However, the standard developers found that the 16-bit Unicode representation had some problems. It forced the issue of standardizing byte ordering. The 2-byte length of each character made the extended header records twice as long for the case of strings coded entirely from historical 7-bit ASCII. For these reasons, the standard developers chose the UTF-8 defined in the ISO/IEC 10646-1: 2000 standard. This multi-byte representation encodes UCS2 or UCS4 characters reliably and deterministically, eliminating the need for a canonical byte ordering. In addition, NUL octets and other characters possibly confusing to POSIX file systems do not appear, except to represent themselves. It was realized that certain national codesets take up more space after the encoding, due to their placement within the UCS range; it was felt that the usefulness of the encoding of the names outweighs the disadvantage of size increase for file, user, and group names.

The encoding of UTF-8 is as follows:

| UCS4 Hex Encoding | UTF-8 Binary Encoding                                 |
|-------------------|-------------------------------------------------------|
| 00000000–0000007F | 0xxxxxxx                                              |
| 00000080–000007FF | 110xxxxx 10xxxxxx                                     |
| 00000800–0000FFFF | 1110xxxx 10xxxxxx 10xxxxxx                            |
| 00010000–001FFFFF | 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx                   |
| 00200000–03FFFFFF | 111110xx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx          |
| 04000000–7FFFFFFF | 1111110x 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx 10xxxxxx |

where each 'x' represents a bit value from the character being translated.

### ustar Interchange Format

The description of the **ustar** format reflects numerous enhancements over pre-1988 versions of the historical *tar* utility. The goal of these changes was not only to provide the functional enhancements desired, but also to retain compatibility between new and old versions. This compatibility has been retained. Archives written using the old archive format are compatible with the new format.

Implementors should be aware that the previous file format did not include a mechanism to archive directory type files. For this reason, the convention of using a filename ending with <slash> was adopted to specify a directory on the archive.

The total size of the *name* and *prefix* fields have been set to meet the minimum requirements for {PATH\_MAX}. If a pathname will fit within the *name* field, it is recommended that the pathname be stored there without the use of the *prefix* field. Although the name field is known to be too small to contain {PATH\_MAX} characters, the value was not changed in this version of the archive file format to retain backwards-compatibility, and instead the prefix was introduced.

Also, because of the earlier version of the format, there is no way to remove the restriction on the *linkname* field being limited in size to just that of the *name* field.

The *size* field is required to be meaningful in all implementation extensions, although it could be zero. This is required so that the data blocks can always be properly counted.

It is suggested that if device special files need to be represented that cannot be represented in the standard format, that one of the extension types (**A-Z**) be used, and that the additional information for the special file be represented as data and be reflected in the *size* field.

Attempting to restore a special file type, where it is converted to ordinary data and conflicts with an existing filename, need not be specially detected by the utility. If run as an ordinary user, *pax* should not be able to overwrite the entries in, for example, */dev* in any case (whether the file is converted to another type or not). If run as a privileged user, it should be able to do so, and it would be considered a bug if it did not. The same is true of ordinary data files and similarly named special files; it is impossible to anticipate the needs of the user (who could really intend to overwrite the file), so the behavior should be predictable (and thus regular) and rely on the protection system as required.

The value 7 in the *typeflag* field is intended to define how contiguous files can be stored in a **ustar** archive. POSIX.1-2008 does not require the contiguous file extension, but does define a standard way of archiving such files so that all conforming systems can interpret these file types in a meaningful and consistent manner. On a system that does not support extended file types, the *pax* utility should do the best it can with the file and go on to the next.

The file protection modes are those conventionally used by the *ls* utility. This is extended beyond the usage in the ISO POSIX-2 standard to support the “shared text” or “sticky” bit. It is intended that the conformance document should not document anything beyond the existence of and support of such a mode. Further extensions are expected to these bits, particularly with overloading the set-user-ID and set-group-ID flags.

### **cpio Interchange Format**

The reference to appropriate privileges in the **cpio** format refers to an error on standard output; the **ustar** format does not make comparable statements.

The model for this format was the historical System V *cpio-c* data interchange format. This model documents the portable version of the **cpio** format and not the binary version. It has the flexibility to transfer data of any type described within POSIX.1-2008, yet is extensible to transfer data types specific to extensions beyond POSIX.1-2008 (for example, contiguous files). Because it describes existing practice, there is no question of maintaining upwards-compatibility.

### **cpio Header**

There has been some concern that the size of the *c\_ino* field of the header is too small to handle those systems that have very large *inode* numbers. However, the *c\_ino* field in the header is used strictly as a hard-link resolution mechanism for archives. It is not necessarily the same value as the *inode* number of the file in the location from which that file is extracted.

The name *c\_magic* is based on historical usage.



**cpio Filename**

For most historical implementations of the *cpio* utility, {PATH\_MAX} octets can be used to describe the pathname without the addition of any other header fields (the NUL character would be included in this count). {PATH\_MAX} is the minimum value for pathname size, documented as 256 bytes. However, an implementation may use *c\_namesize* to determine the exact length of the pathname. With the current description of the **<cpio.h>** header, this pathname size can be as large as a number that is described in six octal digits.

Two values are documented under the *c\_mode* field values to provide for extensibility for known file types:

**0110 000** Reserved for contiguous files. The implementation may treat the rest of the information for this archive like a regular file. If this file type is undefined, the implementation may create the file as a regular file.

This provides for extensibility of the **cpio** format while allowing for the ability to read old archives. Files of an unknown type may be read as “regular files” on some implementations. On a system that does not support extended file types, the *pax* utility should do the best it can with the file and go on to the next.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

Chapter 2 (on page 2335), *cp*, *ed*, *getopts*, *ls*, *printf*

XBD Section 3.169 (on page 60), Chapter 5 (on page 121), Chapter 8 (on page 173), Section 12.2 (on page 216), **<cpio.h>**, **<tar.h>**

XSH *chown()*, *creat()*, *fstatat()*, *mkdir()*, *mkfifo()*, *utime()*, *write()*

**CHANGE HISTORY**

First released in Issue 4.

**Issue 5**

A note is added to the APPLICATION USAGE indicating that the **cpio** and **tar** formats can only support files up to 8 gigabytes in size.

**Issue 6**

The *pax* utility is aligned with the IEEE P1003.2b draft standard:

- Support has been added for symbolic links in the options and interchange formats.
- A new format has been devised, based on extensions to **ustar**.
- References to the “extended” **tar** and **cpio** formats derived from the POSIX.1-1990 standard have been changed to remove the “extended” adjective because this could cause confusion with the extended **tar** header added in this version. (All references to **tar** are actually to **ustar**.)

The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

IEEE PASC Interpretation 1003.2 #168 is applied, clarifying that *mkdir()* and *mkfifo()* calls can ignore an [EEXIST] error when extracting an archive.

IEEE PASC Interpretation 1003.2 #180 is applied, clarifying how extracted files are created when in **read** mode.

IEEE PASC Interpretation 1003.2 #181 is applied, clarifying the description of the **-t** option.

- 103378 IEEE PASC Interpretation 1003.2 #195 is applied.
- 103379 IEEE PASC Interpretation 1003.2 #206 is applied, clarifying the handling of links for the **-H**, **-L**,  
103380 and **-I** options.
- 103381 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/35 is applied, adding the process ID of  
103382 the *pax* process into certain fields. This change provides a method for the implementation to  
103383 ensure that different instances of *pax* extracting a file named */a/b/foo* will not collide when  
103384 processing the extended header information associated with **foo**.
- 103385 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/36 is applied, changing **-x B** to **-x pax** in  
103386 the OPTIONS section.
- 103387 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/20 is applied, updating the SYNOPSIS to  
103388 be consistent with the normative text.
- 103389 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/21 is applied, updating the  
103390 DESCRIPTION to describe the behavior when files to be linked are symbolic links and the  
103391 system is not capable of making hard links to symbolic links.
- 103392 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/22 is applied, updating the OPTIONS  
103393 section to describe the behavior for how multiple **-odelete=pattern** options are to be handled.
- 103394 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/23 is applied, updating the **write** option  
103395 within the OPTIONS section.
- 103396 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/24 is applied, adding a paragraph into  
103397 the OPTIONS section that states that specifying more than one of the mutually-exclusive options  
103398 (**-H** and **-L**) is not considered an error and that the last option specified will determine the  
103399 behavior of the utility.
- 103400 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/25 is applied, removing the *ctime*  
103401 paragraph within the EXTENDED DESCRIPTION. There is a contradiction in the definition of  
103402 the *ctime* keyword for the *pax* extended header, in that the *st\_ctime* member of the **stat** structure  
103403 does not refer to a file creation time. No field in the standard **stat** structure from **<sys/stat.h>**  
103404 includes a file creation time.
- 103405 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/26 is applied, making it clear that *typeflag*  
103406 1 (**ustar** Interchange Format) applies not only to files that are hard-linked, but also to files that  
103407 are aliased via symbolic links.
- 103408 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/27 is applied, clarifying the *cpio c\_nlink*  
103409 field.
- 103410 **Issue 7**
- 103411 Austin Group Interpretations 1003.1-2001 #011, #036, #086, and #109 are applied.
- 103412 Austin Group Interpretation 1003.1-2001 #126 is applied, changing the description of the  
103413 *LC\_MESSAGES* environment variable.
- 103414 SD5-XCU-ERN-2 is applied, making **-c** and **-n** mutually-exclusive in the SYNOPSIS.
- 103415 SD5-XCU-ERN-3 is applied, revising the default behavior of **-H** and **-L**.
- 103416 SD5-XCU-ERN-5, SD5-XCU-ERN-6, SD5-XCU-ERN-7, SD5-XCU-ERN-60 are applied.
- 103417 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.
- 103418 The *pax* utility is no longer allowed to create separate identical symbolic links when extracting  
103419 linked symbolic links from an archive.

103420  
103421  
103422

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0128 [260], XCU/TC1-2008/0129 [261], XCU/TC1-2008/0130 [261], XCU/TC1-2008/0131 [313], and XCU/TC1-2008/0132 [233] are applied.

103423 **NAME**103424 `pr` — print files103425 **SYNOPSIS**

103426 `pr [+page] [-column] [-adFmrt] [-e[char][gap]] [-h header] [-i[char][gap]]`  
 103427 XSI `[-l lines] [-n[char][width]] [-o offset] [-s[char]] [-w width] [ -fp]`  
 103428 `[file...]`

103429 **DESCRIPTION**

103430 The *pr* utility is a printing and pagination filter. If multiple input files are specified, each shall be  
 103431 read, formatted, and written to standard output. By default, the input shall be separated into  
 103432 66-line pages, each with:

- 103433 • A 5-line header that includes the page number, date, time, and the pathname of the file
- 103434 • A 5-line trailer consisting of blank lines

103435 If standard output is associated with a terminal, diagnostic messages shall be deferred until the  
 103436 *pr* utility has completed processing.

103437 When options specifying multi-column output are specified, output text columns shall be of  
 103438 equal width; input lines that do not fit into a text column shall be truncated. By default, text  
 103439 columns shall be separated with at least one <blank>.

103440 **OPTIONS**

103441 The *pr* utility shall conform to XBD [Section 12.2](#) (on page 216), except that: the *page* option has a  
 103442 '+' delimiter; *page* and *column* can be multi-digit numbers; some of the option-arguments are  
 103443 optional; and some of the option-arguments cannot be specified as separate arguments from the  
 103444 preceding option letter. In particular, the *-s* option does not allow the option letter to be  
 103445 separated from its argument, and the options *-e*, *-i*, and *-n* require that both arguments, if  
 103446 present, not be separated from the option letter.

103447 The following options shall be supported. In the following option descriptions, *column*, *lines*,  
 103448 *offset*, *page*, and *width* are positive decimal integers; *gap* is a non-negative decimal integer.

- 103449 **+page** Begin output at page number *page* of the formatted input.
- 103450 **-column** Produce multi-column output that is arranged in *column* columns (the default shall  
 103451 be 1) and is written down each column in the order in which the text is received  
 103452 from the input file. This option should not be used with *-m*. The options *-e* and *-i*  
 103453 shall be assumed for multiple text-column output. Whether or not text columns are  
 103454 produced with identical vertical lengths is unspecified, but a text column shall  
 103455 never exceed the length of the page (see the *-l* option). When used with *-t*, use the  
 103456 minimum number of lines to write the output.
- 103457 **-a** Modify the effect of the *-column* option so that the columns are filled across the  
 103458 page in a round-robin order (for example, when *column* is 2, the first input line  
 103459 heads column 1, the second heads column 2, the third is the second line in column  
 103460 1, and so on).
- 103461 **-d** Produce output that is double-spaced; append an extra <newline> following every  
 103462 <newline> found in the input.
- 103463 **-e[char][gap]**  
 103464 Expand each input <tab> to the next greater column position specified by the  
 103465 formula  $n*gap+1$ , where *n* is an integer > 0. If *gap* is zero or is omitted, it shall  
 103466 default to 8. All <tab> characters in the input shall be expanded into the  
 103467 appropriate number of <space> characters. If any non-digit character, *char*, is  
 103468 specified, it shall be used as the input <tab>. If the first character of the *-e* option-

|        |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|--------|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 103469 |     | argument is a digit, the entire option-argument shall be assumed to be <i>gap</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 103470 | XSI | <b>-f</b> Use a <form-feed> for new pages, instead of the default behavior that uses a sequence of <newline> characters. Pause before beginning the first page if the standard output is associated with a terminal.                                                                                                                                                                                                                                                                                                                                                |
| 103471 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103472 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103473 |     | <b>-F</b> Use a <form-feed> for new pages, instead of the default behavior that uses a sequence of <newline> characters.                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 103474 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103475 |     | <b>-h header</b> Use the string <i>header</i> to replace the contents of the <i>file</i> operand in the page header.                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 103476 |     | <b>-i[char][gap]</b> In output, replace <space> characters with <tab> characters wherever one or more adjacent <space> characters reach column positions <i>gap</i> +1, 2* <i>gap</i> +1, 3* <i>gap</i> +1, and so on. If <i>gap</i> is zero or is omitted, default tab settings at every eighth column position shall be assumed. If any non-digit character, <i>char</i> , is specified, it shall be used as the output <tab>. If the first character of the <b>-i</b> option-argument is a digit, the entire option-argument shall be assumed to be <i>gap</i> . |
| 103477 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103478 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103479 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103480 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103481 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103482 |     | <b>-l lines</b> Override the 66-line default and reset the page length to <i>lines</i> . If <i>lines</i> is not greater than the sum of both the header and trailer depths (in lines), the <i>pr</i> utility shall suppress both the header and trailer, as if the <b>-t</b> option were in effect.                                                                                                                                                                                                                                                                 |
| 103483 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103484 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103485 |     | <b>-m</b> Merge files. Standard output shall be formatted so the <i>pr</i> utility writes one line from each file specified by a <i>file</i> operand, side by side into text columns of equal fixed widths, in terms of the number of column positions. Implementations shall support merging of at least nine <i>file</i> operands.                                                                                                                                                                                                                                |
| 103486 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103487 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103488 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103489 |     | <b>-n[char][width]</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 103490 |     | Provide <i>width</i> -digit line numbering (default for <i>width</i> shall be 5). The number shall occupy the first <i>width</i> column positions of each text column of default output or each line of <b>-m</b> output. If <i>char</i> (any non-digit character) is given, it shall be appended to the line number to separate it from whatever follows (default for <i>char</i> is a <tab>).                                                                                                                                                                     |
| 103491 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103492 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103493 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103494 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103495 |     | <b>-o offset</b> Each line of output shall be preceded by offset <space> characters. If the <b>-o</b> option is not specified, the default offset shall be zero. The space taken is in addition to the output line width (see the <b>-w</b> option below).                                                                                                                                                                                                                                                                                                          |
| 103496 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103497 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103498 |     | <b>-p</b> Pause before beginning each page if the standard output is directed to a terminal ( <i>pr</i> shall write an <alert> to standard error and wait for a <carriage-return> to be read on <i>/dev/tty</i> ).                                                                                                                                                                                                                                                                                                                                                  |
| 103499 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103500 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103501 |     | <b>-r</b> Write no diagnostic reports on failure to open files.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103502 |     | <b>-s[char]</b> Separate text columns by the single character <i>char</i> instead of by the appropriate number of <space> characters (default for <i>char</i> shall be <tab>).                                                                                                                                                                                                                                                                                                                                                                                      |
| 103503 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103504 |     | <b>-t</b> Write neither the five-line identifying header nor the five-line trailer usually supplied for each page. Quit writing after the last line of each file without spacing to the end of the page.                                                                                                                                                                                                                                                                                                                                                            |
| 103505 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103506 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103507 |     | <b>-w width</b> Set the width of the line to <i>width</i> column positions for multiple text-column output only. If the <b>-w</b> option is not specified and the <b>-s</b> option is not specified, the default width shall be 72. If the <b>-w</b> option is not specified and the <b>-s</b> option is specified, the default width shall be 512.                                                                                                                                                                                                                 |
| 103508 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103509 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103510 |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 103511 |     | For single column output, input lines shall not be truncated.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |

103512 **OPERANDS**

103513 The following operand shall be supported:

103514 *file* A pathname of a file to be written. If no *file* operands are specified, or if a *file*  
 103515 operand is '-', the standard input shall be used.

103516 **STDIN**

103517 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '-'.  
 103518 See the INPUT FILES section.

103519 **INPUT FILES**

103520 The input files shall be text files.

103521 The file `/dev/tty` shall be used to read responses required by the `-p` option.103522 **ENVIRONMENT VARIABLES**103523 The following environment variables shall affect the execution of *pr*:

103524 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 103525 (See XBD Section 8.2 (on page 174) the precedence of internationalization variables  
 103526 used to determine the values of locale categories.)

103527 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 103528 internationalization variables.

103529 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 103530 characters (for example, single-byte as opposed to multi-byte characters in  
 103531 arguments and input files) and which characters are defined as printable (character  
 103532 class **print**). Non-printable characters are still written to standard output, but are  
 103533 not counted for the purpose for column-width and line-length calculations.

103534 *LC\_MESSAGES*  
 103535 Determine the locale that should be used to affect the format and contents of  
 103536 diagnostic messages written to standard error.

103537 *LC\_TIME* Determine the format of the date and time for use in writing header lines.103538 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

103539 *TZ* Determine the timezone used to calculate date and time strings written in header  
 103540 lines. If *TZ* is unset or null, an unspecified default timezone shall be used.

103541 **ASYNCHRONOUS EVENTS**

103542 If *pr* receives an interrupt while writing to a terminal, it shall flush all accumulated error  
 103543 messages to the screen before terminating.

103544 **STDOUT**

103545 The *pr* utility output shall be a paginated version of the original file (or files). This pagination  
 103546 shall be accomplished using either <form-feed> characters or a sequence of <newline>  
 103547 XSI characters, as controlled by the `-F` or `-f` option. Page headers shall be generated unless the `-t`  
 103548 option is specified. The page headers shall be of the form:

103549 "`\n\n%s %s Page %d\n\n\n`", <output of date>, <file>, <page number>

103550 In the POSIX locale, the <output of date> field, representing the date and time of last modification  
 103551 of the input file (or the current date and time if the input file is standard input), shall be  
 103552 equivalent to the output of the following command as it would appear if executed at the given  
 103553 time:

103554 `date "+%b %e %H:%M %Y"`

103555 without the trailing <newline>, if the page being written is from standard input. If the page  
 103556 being written is not from standard input, in the POSIX locale, the same format shall be used, but  
 103557 the time used shall be the modification time of the file corresponding to *file* instead of the current  
 103558 time. When the *LC\_TIME* locale category is not set to the POSIX locale, a different format and  
 103559 order of presentation of this field may be used.

103560 If the standard input is used instead of a *file* operand, the <*file*> field shall be replaced by a null  
 103561 string.

103562 If the **-h** option is specified, the <*file*> field shall be replaced by the *header* argument.

#### 103563 **STDERR**

103564 The standard error shall be used for diagnostic messages and for alerting the terminal when **-p**  
 103565 is specified.

#### 103566 **OUTPUT FILES**

103567 None.

#### 103568 **EXTENDED DESCRIPTION**

103569 None.

#### 103570 **EXIT STATUS**

103571 The following exit values shall be returned:

103572 0 Successful completion.

103573 >0 An error occurred.

#### 103574 **CONSEQUENCES OF ERRORS**

103575 Default.

#### 103576 **APPLICATION USAGE**

103577 A conforming application must protect its first operand, if it starts with a <plus-sign>, by  
 103578 preceding it with the "--" argument that denotes the end of the options. For example, *pr+x*  
 103579 could be interpreted as an invalid page number or a *file* operand.

#### 103580 **EXAMPLES**

103581 1. Print a numbered list of all files in the current directory:

103582 `ls -a | pr -n -h "Files in $(pwd)."`

103583 2. Print **file1** and **file2** as a double-spaced, three-column listing headed by "file list":

103584 `pr -3d -h "file list" file1 file2`

103585 3. Write **file1** on **file2**, expanding tabs to columns 10, 19, 28, ...:

103586 `pr -e9 -t <file1 >file2`

#### 103587 **RATIONALE**

103588 This utility is one of those that does not follow the Utility Syntax Guidelines because of its  
 103589 historical origins. The standard developers could have added new options that obeyed the  
 103590 guidelines (and marked the old options obsolescent) or devised an entirely new utility; there are  
 103591 examples of both actions in this volume of POSIX.1-2008. Because of its widespread use by  
 103592 historical applications, the standard developers decided to exempt this version of *pr* from many  
 103593 of the guidelines.

103594 Implementations are required to accept option-arguments to the **-h**, **-l**, **-o**, and **-w** options  
 103595 whether presented as part of the same argument or as a separate argument to *pr*, as suggested by  
 103596 the Utility Syntax Guidelines. The **-n** and **-s** options, however, are specified as in historical



practice because they are frequently specified without their optional arguments. If a <blank> were allowed before the option-argument in these cases, a *file* operand could mistakenly be interpreted as an option-argument in historical applications.

The text about the minimum number of lines in multi-column output was included to ensure that a best effort is made in balancing the length of the columns. There are known historical implementations in which, for example, 60-line files are listed by *pr* -2 as one column of 56 lines and a second of 4. Although this is not a problem when a full page with headers and trailers is produced, it would be relatively useless when used with -t.

Historical implementations of the *pr* utility have differed in the action taken for the -f option. BSD uses it as described here for the -F option; System V uses it to change trailing <newline> characters on each page to a <form-feed> and, if standard output is a TTY device, sends an <alert> to standard error and reads a line from /dev/tty before the first page. There were strong arguments from both sides of this issue concerning historical practice and as a result the -F option was added. XSI-conformant systems support the System V historical actions for the -f option.

The <output of date> field in the -l format is specified only for the POSIX locale. As noted, the format can be different in other locales. No mechanism for defining this is present in this volume of POSIX.1-2008, as the appropriate vehicle is a message catalog; that is, the format should be specified as a “message”.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*expand*, *lp*

XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

#### CHANGE HISTORY

First released in Issue 2.

#### Issue 6

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The -p option is added.

The normative text is reworded to avoid use of the term “must” for application requirements.

#### Issue 7

PASC Interpretation 1003.2-92 #151 (SD5-XCU-ERN-44) is applied.

Austin Group Interpretation 1003.1-2001 #093 is applied.

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



103632 **NAME**

103633       printf — write formatted output

103634 **SYNOPSIS**103635       printf *format* [*argument...*]103636 **DESCRIPTION**

103637       The *printf* utility shall write formatted operands to the standard output. The *argument* operands  
 103638       shall be formatted under control of the *format* operand.

103639 **OPTIONS**

103640       None.

103641 **OPERANDS**

103642       The following operands shall be supported:

103643       *format*       A string describing the format to use to write the remaining operands. See the  
 103644       EXTENDED DESCRIPTION section.

103645       *argument*   The strings to be written to standard output, under the control of *format*. See the  
 103646       EXTENDED DESCRIPTION section.

103647 **STDIN**

103648       Not used.

103649 **INPUT FILES**

103650       None.

103651 **ENVIRONMENT VARIABLES**103652       The following environment variables shall affect the execution of *printf*:

103653       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 103654       (See XBD Section 8.2 (on page 174) the precedence of internationalization variables  
 103655       used to determine the values of locale categories.)

103656       *LC\_ALL*      If set to a non-empty string value, override the values of all the other  
 103657       internationalization variables.

103658       *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 103659       characters (for example, single-byte as opposed to multi-byte characters in  
 103660       arguments).

103661       *LC\_MESSAGES*

103662       Determine the locale that should be used to affect the format and contents of  
 103663       diagnostic messages written to standard error.

103664       *LC\_NUMERIC*

103665       Determine the locale for numeric formatting. It shall affect the format of numbers  
 103666       written using the e, E, f, g, and G conversion specifier characters (if supported).

103667 XSI       *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

103668 **ASYNCHRONOUS EVENTS**

103669       Default.

103670 **STDOUT**

103671       See the EXTENDED DESCRIPTION section.

103672 **STDERR**

103673 The standard error shall be used only for diagnostic messages.

103674 **OUTPUT FILES**

103675 None.

103676 **EXTENDED DESCRIPTION**

103677 The *format* operand shall be used as the *format* string described in XBD Chapter 5 (on page 121)  
103678 with the following exceptions:

- 103679 1. A <space> in the format string, in any context other than a flag of a conversion  
103680 specification, shall be treated as an ordinary character that is copied to the output.
- 103681 2. A 'Δ' character in the format string shall be treated as a 'Δ' character, not as a <space>.
- 103682 3. In addition to the escape sequences shown in XBD Chapter 5 (on page 121) ('\\', '\a',  
103683 '\b', '\f', '\n', '\r', '\t', '\v'), "\ddd", where *ddd* is a one, two, or three-digit  
103684 octal number, shall be written as a byte with the numeric value specified by the octal  
103685 number.
- 103686 4. The implementation shall not precede or follow output from the d or u conversion  
103687 specifiers with <blank> characters not specified by the *format* operand.
- 103688 5. The implementation shall not precede output from the o conversion specifier with zeros  
103689 not specified by the *format* operand.
- 103690 6. The a, A, e, E, f, F, g, and G conversion specifiers need not be supported.
- 103691 7. An additional conversion specifier character, b, shall be supported as follows. The  
103692 argument shall be taken to be a string that can contain <backslash>-escape sequences. |  
103693 The following <backslash>-escape sequences shall be supported:
  - 103694 — The escape sequences listed in XBD Chapter 5 (on page 121) ('\\', '\a', '\b',  
103695 '\f', '\n', '\r', '\t', '\v'), which shall be converted to the characters they  
103696 represent
  - 103697 — "\0ddd", where *ddd* is a zero, one, two, or three-digit octal number that shall be  
103698 converted to a byte with the numeric value specified by the octal number
  - 103699 — '\c', which shall not be written and shall cause *printf* to ignore any remaining  
103700 characters in the string operand containing it, any remaining string operands, and  
103701 any additional characters in the *format* operand

103702 The interpretation of a <backslash> followed by any other sequence of characters is  
103703 unspecified.

103704 Bytes from the converted string shall be written until the end of the string or the number  
103705 of bytes indicated by the precision specification is reached. If the precision is omitted, it  
103706 shall be taken to be infinite, so all bytes up to the end of the converted string shall be  
103707 written.

- 103708 8. For each conversion specification that consumes an argument, the next *argument* operand |  
103709 shall be evaluated and converted to the appropriate type for the conversion as specified  
103710 below.
- 103711 9. The *format* operand shall be reused as often as necessary to satisfy the *argument* operands. |  
103712 Any extra b, c, or s conversion specifiers shall be evaluated as if a null string argument  
103713 were supplied; other extra conversion specifications shall be evaluated as if a zero  
103714 argument were supplied. If the *format* operand contains no conversion specifications and  
103715 *argument* operands are present, the results are unspecified.

10. If a character sequence in the *format* operand begins with a '%' character, but does not form a valid conversion specification, the behavior is unspecified.
11. The argument to the *c* conversion specifier can be a string containing zero or more bytes. If it contains one or more bytes, the first byte shall be written and any additional bytes shall be ignored. If the argument is an empty string, it is unspecified whether nothing is written or a null byte is written.

The *argument* operands shall be treated as strings if the corresponding conversion specifier is *b*, *c*, or *s*, and shall be evaluated as if by the *strtod()* function if the corresponding conversion specifier is *a*, *A*, *e*, *E*, *f*, *F*, *g*, or *G*. Otherwise, they shall be evaluated as unsuffixed C integer constants, as described by the ISO C standard, with the following extensions:

- A leading <plus-sign> or <hyphen-minus> shall be allowed.
- If the leading character is a single-quote or double-quote, the value shall be the numeric value in the underlying codeset of the character following the single-quote or double-quote.
- Suffixed integer constants may be allowed.

If an *argument* operand cannot be completely converted into an internal value appropriate to the corresponding conversion specification, a diagnostic message shall be written to standard error and the utility shall not exit with a zero exit status, but shall continue processing any remaining operands and shall write the value accumulated at the time the error was detected to standard output.

It shall not be considered an error if an *argument* operand is not completely used for a *b*, *c*, or *s* conversion.

## EXIT STATUS

The following exit values shall be returned:

- 0 Successful completion.
- >0 An error occurred.

## CONSEQUENCES OF ERRORS

Default.

## APPLICATION USAGE

The floating-point formatting conversion specifications of *printf()* are not required because all arithmetic in the shell is integer arithmetic. The *awk* utility performs floating-point calculations and provides its own **printf** function. The *bc* utility can perform arbitrary-precision floating-point arithmetic, but does not provide extensive formatting capabilities. (This *printf* utility cannot really be used to format *bc* output; it does not support arbitrary precision.) Implementations are encouraged to support the floating-point conversions as an extension.

Note that this *printf* utility, like the *printf()* function defined in the System Interfaces volume of POSIX.1-2008 on which it is based, makes no special provision for dealing with multi-byte characters when using the *%c* conversion specification or when a precision is specified in a *%b* or *%s* conversion specification. Applications should be extremely cautious using either of these features when there are multi-byte characters in the character set.

No provision is made in this volume of POSIX.1-2008 which allows field widths and precisions to be specified as '\*' since the '\*' can be replaced directly in the *format* operand using shell variable substitution. Implementations can also provide this feature as an extension if they so choose.

Hexadecimal character constants as defined in the ISO C standard are not recognized in the *format* operand because there is no consistent way to detect the end of the constant. Octal character constants are limited to, at most, three octal digits, but hexadecimal character constants are only terminated by a non-hex-digit character. In the ISO C standard, the "##" concatenation operator can be used to terminate a constant and follow it with a hexadecimal character to be written. In the shell, concatenation occurs before the *printf* utility has a chance to parse the end of the hexadecimal constant.

The %b conversion specification is not part of the ISO C standard; it has been added here as a portable way to process <backslash>-escapes expanded in string operands as provided by the *echo* utility. See also the APPLICATION USAGE section of *echo* (on page 2663) for ways to use *printf* as a replacement for all of the traditional versions of the *echo* utility.

If an argument cannot be parsed correctly for the corresponding conversion specification, the *printf* utility is required to report an error. Thus, overflow and extraneous characters at the end of an argument being used for a numeric conversion shall be reported as errors.

## EXAMPLES

To alert the user and then print and read a series of prompts:

```
printf "\aPlease fill in the following: \nName: "
read name
printf "Phone number: "
read phone
```

To read out a list of right and wrong answers from a file, calculate the percentage correctly, and print them out. The numbers are right-justified and separated by a single <tab>. The percentage is written to one decimal place of accuracy:

```
while read right wrong ; do
    percent=$(echo "scale=1;($right*100)/($right+$wrong)" | bc)
    printf "%2d right\t%2d wrong\t(%s%%)\n" \
        $right $wrong $percent
done < database_file
```

The command:

```
printf "%5d%4d\n" 1 21 321 4321 54321
```

produces:

```
      1  21
     3214321
    54321   0
```

Note that the *format* operand is used three times to print all of the given strings and that a '0' was supplied by *printf* to satisfy the last %4d conversion specification.

The *printf* utility is required to notify the user when conversion errors are detected while producing numeric output; thus, the following results would be expected on an implementation with 32-bit twos-complement integers when %d is specified as the *format* operand:

| Argument    | Standard Output | Diagnostic Output                                |
|-------------|-----------------|--------------------------------------------------|
| 5a          | 5               | <b>printf: "5a" not completely converted</b>     |
| 9999999999  | 2147483647      | <b>printf: "9999999999" arithmetic overflow</b>  |
| -9999999999 | -2147483648     | <b>printf: "-9999999999" arithmetic overflow</b> |
| ABC         | 0               | <b>printf: "ABC" expected numeric value</b>      |

The diagnostic message format is not specified, but these examples convey the type of information that should be reported. Note that the value shown on standard output is what would be expected as the return value from the *strtol()* function as defined in the System Interfaces volume of POSIX.1-2008. A similar correspondence exists between %u and *strtoul()* and %e, %f, and %g (if the implementation supports floating-point conversions) and *strtod()*.

In a locale using the ISO/IEC 646: 1991 standard as the underlying codeset, the command:

```
printf "%d\n" 3 +3 -3 \'3 \"+3 "'-3"
```

produces:

```
3    Numeric value of constant 3
```

```
3    Numeric value of constant 3
```

```
-3   Numeric value of constant -3
```

```
51   Numeric value of the character '3' in the ISO/IEC 646: 1991 standard codeset
```

```
43   Numeric value of the character '+' in the ISO/IEC 646: 1991 standard codeset
```

```
45   Numeric value of the character '-' in the ISO/IEC 646: 1991 standard codeset
```

Note that in a locale with multi-byte characters, the value of a character is intended to be the value of the equivalent of the **wchar\_t** representation of the character as described in the System Interfaces volume of POSIX.1-2008.

## RATIONALE

The *printf* utility was added to provide functionality that has historically been provided by *echo*. However, due to irreconcilable differences in the various versions of *echo* extant, the version has few special features, leaving those to this new *printf* utility, which is based on one in the Ninth Edition system.

The EXTENDED DESCRIPTION section almost exactly matches the *printf()* function in the ISO C standard, although it is described in terms of the file format notation in XBD [Chapter 5](#) (on page 121).

Earlier versions of this standard specified that arguments for all conversions other than b, c, and s were evaluated in the same way (as C constants, but with stated exceptions). For implementations supporting the floating-point conversions it was not clear whether integer conversions need only accept integer constants and floating-point conversions need only accept floating-point constants, or whether both types of conversions should accept both types of constants. Also by not distinguishing between them, the requirement relating to a leading single-quote or double-quote applied to floating-point conversions even though this provided no useful functionality to applications that was not already available through the integer conversions. The current standard clarifies the situation by specifying that the arguments for floating-point conversions are evaluated as if by *strtod()*, and the arguments for integer conversions are evaluated as C integer constants, with the special treatment of leading single-quote and double-quote applying only to integer conversions.

103842 **FUTURE DIRECTIONS**

103843 None.

103844 **SEE ALSO**103845 *awk, bc, echo*103846 XBD [Chapter 5](#) (on page 121), [Chapter 8](#) (on page 173)103847 XSH *fprintf()*, *strtod()*103848 **CHANGE HISTORY**

103849 First released in Issue 4.

103850 **Issue 7**

103851 Austin Group Interpretations 1003.1-2001 #175 and #177 are applied.

103852 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

103853 **NAME**103854        **prs** — print an SCCS file (**DEVELOPMENT**)103855 **SYNOPSIS**

```
103856 XSI      prs [-a] [-d dataspec] [-r[SID]] file...
103857          prs [-e|-l] -c cutoff [-d dataspec] file...
103858          prs [-e|-l] -r[SID] [-d dataspec] file...
```

103859 **DESCRIPTION**

103860        The *prs* utility shall write to standard output parts or all of an SCCS file in a user-supplied  
 103861        format.

103862 **OPTIONS**

103863        The *prs* utility shall conform to XBD [Section 12.2](#) (on page 216), except that the **-r** option has an  
 103864        optional option-argument. This optional option-argument cannot be presented as a separate  
 103865        argument. The following options shall be supported:

103866        **-d *dataspec***    Specify the output data specification. The *dataspec* shall be a string consisting of  
 103867        SCCS file *data keywords* (see [Data Keywords](#), on page 3108) interspersed with  
 103868        optional user-supplied text.

103869        **-r[*SID*]**        Specify the SCCS identification string (SID) of a delta for which information is  
 103870        desired. If no *SID* option-argument is specified, the SID of the most recently  
 103871        created delta shall be assumed.

103872        **-e**                Request information for all deltas created earlier than and including the delta  
 103873        designated via the **-r** option or the date-time given by the **-c** option.

103874        **-l**                Request information for all deltas created later than and including the delta  
 103875        designated via the **-r** option or the date-time given by the **-c** option.

103876        **-c *cutoff***        Indicate the *cutoff* date-time, in the form:

103877        YY[MM[DD[HH[MM[SS]]]]]

103878        For the YY component, values in the range [69,99] shall refer to years 1969 to 1999  
 103879        inclusive, and values in the range [00,68] shall refer to years 2000 to 2068 inclusive.

103880        **Note:**        It is expected that in a future version of this standard the default century inferred  
 103881        from a 2-digit year will change. (This would apply to all commands accepting a  
 103882        2-digit year as input.)

103883        No changes (deltas) to the SCCS file that were created after the specified *cutoff*  
 103884        date-time shall be included in the output. Units omitted from the date-time default  
 103885        to their maximum possible values; for example, **-c 7502** is equivalent to  
 103886        **-c 750228235959**.

103887        **-a**                Request writing of information for both removed—that is, *delta type=R* (see  
 103888        [rmDEL](#))—and existing—that is, *delta type=D*,—deltas. If the **-a** option is not  
 103889        specified, information for existing deltas only shall be provided.

103890 **OPERANDS**

103891        The following operand shall be supported:

103892        *file*            A pathname of an existing SCCS file or a directory. If *file* is a directory, the *prs*  
 103893        utility shall behave as though each file in the directory were specified as a named  
 103894        file, except that non-SCCS files (last component of the pathname does not begin  
 103895        with **s**.) and unreadable files shall be silently ignored.



103896 If exactly one *file* operand appears, and it is *'-'*, the standard input shall be read;  
 103897 each line of the standard input shall be taken to be the name of an SCCS file to be  
 103898 processed. Non-SCCS files and unreadable files shall be silently ignored.

#### 103899 **STDIN**

103900 The standard input shall be a text file used only when the *file* operand is specified as *'-'*. Each  
 103901 line of the text file shall be interpreted as an SCCS pathname.

#### 103902 **INPUT FILES**

103903 Any SCCS files displayed are files of an unspecified format.

#### 103904 **ENVIRONMENT VARIABLES**

103905 The following environment variables shall affect the execution of *prs*:

103906 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 103907 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 103908 used to determine the values of locale categories.)

103909 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 103910 internationalization variables.

103911 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 103912 characters (for example, single-byte as opposed to multi-byte characters in  
 103913 arguments and input files).

103914 *LC\_MESSAGES*

103915 Determine the locale that should be used to affect the format and contents of  
 103916 diagnostic messages written to standard error.

103917 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

#### 103918 **ASYNCHRONOUS EVENTS**

103919 Default.

#### 103920 **STDOUT**

103921 The standard output shall be a text file whose format is dependent on the data keywords  
 103922 specified with the *-d* option.

#### 103923 **Data Keywords**

103924 Data keywords specify which parts of an SCCS file shall be retrieved and output. All parts of an  
 103925 SCCS file have an associated data keyword. A data keyword may appear in a *dataspec* multiple  
 103926 times.

103927 The information written by *prs* shall consist of:

- 103928 1. The user-supplied text
- 103929 2. Appropriate values (extracted from the SCCS file) substituted for the recognized data  
 103930 keywords in the order of appearance in the *dataspec*

103931 The format of a data keyword value shall either be simple (*'S'*), in which keyword substitution  
 103932 is direct, or multi-line (*'M'*).

103933 User-supplied text shall be any text other than recognized data keywords. A <tab> shall be  
 103934 specified by *'\t'* and <newline> by *'\n'*. When the *-r* option is not specified, the default  
 103935 *dataspec* shall be:

103936 *:PN::\n\n*

103937 and the following *dataspec* shall be used for each selected delta:



103938 :Dt:\t:DL:\nMRs:\n:MR:COMMENTS:\n:C:

103939

103940

103941

103942

103943

103944

103945

103946

103947

103948

103949

103950

103951

103952

103953

103954

103955

103956

103957

103958

103959

103960

103961

103962

103963

103964

103965

103966

103967

103968

103969

103970

103971

103972

103973

103974

103975

103976

103977

103978

103979

103980

103981

103982

103983

103984

103985

103986

| SCCS File Data Keywords |                                                          |              |                |        |
|-------------------------|----------------------------------------------------------|--------------|----------------|--------|
| Keyword                 | Data Item                                                | File Section | Value          | Format |
| :Dt:                    | Delta information                                        | Delta Table  | See below*     | S      |
| :DL:                    | Delta line statistics                                    | "            | :Li:/:Ld:/:Lu: | S      |
| :Li:                    | Lines inserted by Delta                                  | "            | nnnnn***       | S      |
| :Ld:                    | Lines deleted by Delta                                   | "            | nnnnn***       | S      |
| :Lu:                    | Lines unchanged by Delta                                 | "            | nnnnn***       | S      |
| :DT:                    | Delta type                                               | "            | D or R         | S      |
| :I:                     | SCCS ID string (SID)                                     | "            | See below**    | S      |
| :R:                     | Release number                                           | "            | nnnn           | S      |
| :L:                     | Level number                                             | "            | nnnn           | S      |
| :B:                     | Branch number                                            | "            | nnnn           | S      |
| :S:                     | Sequence number                                          | "            | nnnn           | S      |
| :D:                     | Date delta created                                       | "            | :Dy:/:Dm:/:Dd: | S      |
| :Dy:                    | Year delta created                                       | "            | nn             | S      |
| :Dm:                    | Month delta created                                      | "            | nn             | S      |
| :Dd:                    | Day delta created                                        | "            | nn             | S      |
| :T:                     | Time delta created                                       | "            | :Th:::Tm:::Ts: | S      |
| :Th:                    | Hour delta created                                       | "            | nn             | S      |
| :Tm:                    | Minutes delta created                                    | "            | nn             | S      |
| :Ts:                    | Seconds delta created                                    | "            | nn             | S      |
| :P:                     | Programmer who created Delta                             | "            | logname        | S      |
| :DS:                    | Delta sequence number                                    | "            | nnnn           | S      |
| :DP:                    | Predecessor Delta sequence number                        | "            | nnnn           | S      |
| :DI:                    | Sequence number of deltas included, excluded, or ignored | "            | :Dn:/:Dx:/:Dg: | S      |
| :Dn:                    | Deltas included (sequence #)                             | "            | :DS: :DS: ...  | S      |
| :Dx:                    | Deltas excluded (sequence #)                             | "            | :DS: :DS: ...  | S      |
| :Dg:                    | Deltas ignored (sequence #)                              | "            | :DS: :DS: ...  | S      |
| :MR:                    | MR numbers for delta                                     | "            | text           | M      |
| :C:                     | Comments for delta                                       | "            | text           | M      |
| :UN:                    | User names                                               | User Names   | text           | M      |
| :FL:                    | Flag list                                                | Flags        | text           | M      |
| :Y:                     | Module type flag                                         | "            | text           | S      |
| :MF:                    | MR validation flag                                       | "            | yes or no      | S      |
| :MP:                    | MR validation program name                               | "            | text           | S      |
| :KF:                    | Keyword error, warning flag                              | "            | yes or no      | S      |
| :KV:                    | Keyword validation string                                | "            | text           | S      |
| :BF:                    | Branch flag                                              | "            | yes or no      | S      |
| :J:                     | Joint edit flag                                          | "            | yes or no      | S      |
| :LK:                    | Locked releases                                          | "            | :R: ...        | S      |
| :Q:                     | User-defined keyword                                     | "            | text           | S      |
| :M:                     | Module name                                              | "            | text           | S      |
| :FB:                    | Floor boundary                                           | "            | :R:            | S      |
| :CB:                    | Ceiling boundary                                         | "            | :R:            | S      |
| :Ds:                    | Default SID                                              | "            | :I:            | S      |
| :ND:                    | Null delta flag                                          | "            | yes or no      | S      |

103987

103988

103989

103990

103991

103992

103993

103994

103995

103996

| SCCS File Data Keywords |                              |              |                   |        |
|-------------------------|------------------------------|--------------|-------------------|--------|
| Keyword                 | Data Item                    | File Section | Value             | Format |
| :FD:                    | File descriptive text        | Comments     | <i>text</i>       | M      |
| :BD:                    | Body                         | Body         | <i>text</i>       | M      |
| :GB:                    | Gotten body                  | "            | <i>text</i>       | M      |
| :W:                     | A form of <i>what</i> string | N/A          | :Z::M:\t:I:       | S      |
| :A:                     | A form of <i>what</i> string | N/A          | :Z::Y: :M: :I::Z: | S      |
| :Z:                     | <i>what</i> string delimiter | N/A          | @ ( # )           | S      |
| :F:                     | SCCS filename                | N/A          | <i>text</i>       | S      |
| :PN:                    | SCCS file pathname           | N/A          | <i>text</i>       | S      |

103997

\* :Dt::DT: :I: :D: :T: :P: :DS: :DP:

103998

\*\* :R::L::B::S: if the delta is a branch delta (:BF:= =yes)

103999

:R::L: if the delta is not a branch delta (:BF:= =no)

104000

\*\*\* The line statistics are capped at 99 999. For example, if 100 000 lines were unchanged in a certain revision, :Lu: shall produce the value 99 999.

104001

104002 **STDERR**

104003

The standard error shall be used only for diagnostic messages.

104004 **OUTPUT FILES**

104005

None.

104006 **EXTENDED DESCRIPTION**

104007

None.

104008 **EXIT STATUS**

104009

The following exit values shall be returned:

104010

0 Successful completion.

104011

&gt;0 An error occurred.

104012 **CONSEQUENCES OF ERRORS**

104013

Default.

104014 **APPLICATION USAGE**

104015

None.

104016 **EXAMPLES**

104017

1. The following example:

104018

prs -d "User Names for :F: are:\n:UN:" s.file

104019

might write to standard output:

104020

User Names for s.file are:

104021

xyz

104022

131

104023

abc

104024

2. The following example:

104025

prs -d "Delta for pgm :M:: :I: - :D: By :P:" -r s.file

104026

might write to standard output:

104027

Delta for pgm main.c: 3.7 - 77/12/01 By cas

104028 3. As a special case:  
104029 prs s.file  
104030 might write to standard output:  
104031 s.file:  
104032 <blank line>  
104033 D 1.1 77/12/01 00:00:00 cas 1 000000/00000/00000  
104034 MRS:  
104035 b178-12345  
104036 b179-54321  
104037 COMMENTS:  
104038 this is the comment line for s.file initial delta  
104039 <blank line>  
  
104040 for each delta table entry of the **D** type. The only option allowed to be used with this  
104041 special case is the **-a** option.  
  
104042 **RATIONALE**  
104043 None.  
  
104044 **FUTURE DIRECTIONS**  
104045 None.  
  
104046 **SEE ALSO**  
104047 *admin, delta, get, what*  
  
104048 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)  
  
104049 **CHANGE HISTORY**  
104050 First released in Issue 2.  
  
104051 **Issue 5**  
104052 The phrase “in which keyword substitution is followed by a <newline>” is deleted from the end  
104053 of the second paragraph of [Data Keywords](#) (on page 3108).  
  
104054 The interpretation of the YY component of the **-c cutoff** argument is noted.  
  
104055 **Issue 6**  
104056 The normative text is reworded to emphasize the term “shall” for implementation requirements.  
104057 The Open Group Base Resolution bwg2001-007 is applied, updating the table in STDOUT with a  
104058 note that line statistics are capped at 99 999 for the **:Li:**, **:Ld:**, **:Lu:**, and **:DL:** keywords.  
  
104059 The Open Group Interpretation PIN4C.00009 is applied.  
  
104060 **Issue 7**  
104061 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

104062 **NAME**

104063       ps — report process status

104064 **SYNOPSIS**

```

104065 XSI      ps [-aA] [-defl] [-g grouplist] [-G grouplist]
104066          [-n namelist] [-o format]... [-p proclist] [-t termlist]
104067          [-u userlist] [-U userlist]

```

104068 **DESCRIPTION**

104069       The *ps* utility shall write information about processes, subject to having appropriate privileges to  
 104070       obtain information about those processes.

104071       By default, *ps* shall select all processes with the same effective user ID as the current user and the  
 104072       same controlling terminal as the invoker.

104073 **OPTIONS**

104074       The *ps* utility shall conform to XBD [Section 12.2](#) (on page 216).

104075       The following options shall be supported:

104076       **-a**       Write information for all processes associated with terminals. Implementations  
 104077       may omit session leaders from this list.

104078       **-A**       Write information for all processes.

104079 XSI       **-d**       Write information for all processes, except session leaders.

104080 XSI       **-e**       Write information for all processes. (Equivalent to **-A**.)

104081 XSI       **-f**       Generate a **full** listing. (See the STDOUT section for the contents of a **full** listing.)

104082 XSI       **-g grouplist** Write information for processes whose session leaders are given in *grouplist*. The  
 104083       application shall ensure that the *grouplist* is a single argument in the form of a  
 104084       <blank> or <comma>-separated list.

104085       **-G grouplist** Write information for processes whose real group ID numbers are given in  
 104086       *grouplist*. The application shall ensure that the *grouplist* is a single argument in the  
 104087       form of a <blank> or <comma>-separated list.

104088 XSI       **-l**       Generate a **long** listing. (See STDOUT for the contents of a **long** listing.)

104089 XSI       **-n namelist** Specify the name of an alternative system *namelist* file in place of the default. The  
 104090       name of the default file and the format of a *namelist* file are unspecified.

104091       **-o format**   Write information according to the format specification given in *format*. This is  
 104092       fully described in the STDOUT section. Multiple **-o** options can be specified; the  
 104093       format specification shall be interpreted as the <space>-separated concatenation of  
 104094       all the *format* option-arguments.

104095       **-p proclist** Write information for processes whose process ID numbers are given in *proclist*.  
 104096       The application shall ensure that the *proclist* is a single argument in the form of a  
 104097       <blank> or <comma>-separated list.

104098       **-t termlist** Write information for processes associated with terminals given in *termlist*. The  
 104099       application shall ensure that the *termlist* is a single argument in the form of a  
 104100       <blank> or <comma>-separated list. Terminal identifiers shall be given in an  
 104101 XSI       implementation-defined format. On XSI-conformant systems, they shall be given  
 104102       in one of two forms: the device's filename (for example, **tty04**) or, if the device's  
 104103       filename starts with **tty**, just the identifier following the characters **tty** (for example,  
 104104       "04").

|        |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
|--------|-----|------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 104105 | XSI | <b>-u <i>userlist</i></b>    | Write information for processes whose user ID numbers or login names are given in <i>userlist</i> . The application shall ensure that the <i>userlist</i> is a single argument in the form of a <blank> or <comma>-separated list. In the listing, the numerical user ID shall be written unless the <b>-f</b> option is used, in which case the login name shall be written. |
| 104106 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104107 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104108 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104109 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104110 |     | <b>-U <i>userlist</i></b>    | Write information for processes whose real user ID numbers or login names are given in <i>userlist</i> . The application shall ensure that the <i>userlist</i> is a single argument in the form of a <blank> or <comma>-separated list.                                                                                                                                       |
| 104111 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104112 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104113 | XSI |                              | With the exception of <b>-f</b> , <b>-l</b> , <b>-n <i>namelist</i></b> , and <b>-o <i>format</i></b> , all of the options shown are used to select processes. If any are specified, the default list shall be ignored and <i>ps</i> shall select the processes represented by the inclusive OR of all the selection-criteria options.                                        |
| 104114 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104115 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104116 |     | <b>OPERANDS</b>              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104117 |     |                              | None.                                                                                                                                                                                                                                                                                                                                                                         |
| 104118 |     | <b>STDIN</b>                 |                                                                                                                                                                                                                                                                                                                                                                               |
| 104119 |     |                              | Not used.                                                                                                                                                                                                                                                                                                                                                                     |
| 104120 |     | <b>INPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                                               |
| 104121 |     |                              | None.                                                                                                                                                                                                                                                                                                                                                                         |
| 104122 |     | <b>ENVIRONMENT VARIABLES</b> |                                                                                                                                                                                                                                                                                                                                                                               |
| 104123 |     |                              | The following environment variables shall affect the execution of <i>ps</i> :                                                                                                                                                                                                                                                                                                 |
| 104124 |     | <b>COLUMNS</b>               | Override the system-selected horizontal display line size, used to determine the number of text columns to display. See XBD <a href="#">Chapter 8</a> (on page 173) for valid values and results when it is unset or null.                                                                                                                                                    |
| 104125 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104126 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104127 |     | <b>LANG</b>                  | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) the precedence of internationalization variables used to determine the values of locale categories.)                                                                                                                                |
| 104128 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104129 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104130 |     | <b>LC_ALL</b>                | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                                      |
| 104131 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104132 |     | <b>LC_CTYPE</b>              | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).                                                                                                                                                                                                     |
| 104133 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104134 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104135 |     | <b>LC_MESSAGES</b>           |                                                                                                                                                                                                                                                                                                                                                                               |
| 104136 |     |                              | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error and informative messages written to standard output.                                                                                                                                                                                              |
| 104137 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104138 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104139 |     | <b>LC_TIME</b>               | Determine the format and contents of the date and time strings displayed.                                                                                                                                                                                                                                                                                                     |
| 104140 | XSI | <b>NLSPATH</b>               | Determine the location of message catalogs for the processing of <b>LC_MESSAGES</b> .                                                                                                                                                                                                                                                                                         |
| 104141 |     | <b>TZ</b>                    | Determine the timezone used to calculate date and time strings displayed. If <b>TZ</b> is unset or null, an unspecified default timezone shall be used.                                                                                                                                                                                                                       |
| 104142 |     |                              |                                                                                                                                                                                                                                                                                                                                                                               |
| 104143 |     | <b>ASYNCHRONOUS EVENTS</b>   |                                                                                                                                                                                                                                                                                                                                                                               |
| 104144 |     |                              | Default.                                                                                                                                                                                                                                                                                                                                                                      |

104145 **STDOUT**

104146 When the **-o** option is not specified, the standard output format is unspecified.

104147 XSI On XSI-conformant systems, the output format shall be as follows. The column headings and  
 104148 descriptions of the columns in a *ps* listing are given below. The precise meanings of these fields  
 104149 are implementation-defined. The letters 'f' and 'l' (below) indicate the option (**full** or **long**)  
 104150 that shall cause the corresponding heading to appear; **all** means that the heading always  
 104151 appears. Note that these two options determine only what information is provided for a process;  
 104152 they do not determine which processes are listed.

|        |              |       |                                                                                                      |
|--------|--------------|-------|------------------------------------------------------------------------------------------------------|
| 104153 | <b>F</b>     | (l)   | Flags (octal and additive) associated with the process.                                              |
| 104154 | <b>S</b>     | (l)   | The state of the process.                                                                            |
| 104155 | <b>UID</b>   | (f,l) | The user ID number of the process owner; the login name is printed<br>under the <b>-f</b> option.    |
| 104157 | <b>PID</b>   | (all) | The process ID of the process; it is possible to kill a process if this<br>datum is known.           |
| 104159 | <b>PPID</b>  | (f,l) | The process ID of the parent process.                                                                |
| 104160 | <b>C</b>     | (f,l) | Processor utilization for scheduling.                                                                |
| 104161 | <b>PRI</b>   | (l)   | The priority of the process; higher numbers mean lower priority.                                     |
| 104162 | <b>NI</b>    | (l)   | Nice value; used in priority computation.                                                            |
| 104163 | <b>ADDR</b>  | (l)   | The address of the process.                                                                          |
| 104164 | <b>SZ</b>    | (l)   | The size in blocks of the core image of the process.                                                 |
| 104165 | <b>WCHAN</b> | (l)   | The event for which the process is waiting or sleeping; if blank, the<br>process is running.         |
| 104167 | <b>STIME</b> | (f)   | Starting time of the process.                                                                        |
| 104168 | <b>TTY</b>   | (all) | The controlling terminal for the process.                                                            |
| 104169 | <b>TIME</b>  | (all) | The cumulative execution time for the process.                                                       |
| 104170 | <b>CMD</b>   | (all) | The command name; the full command name and its arguments are<br>written under the <b>-f</b> option. |

104172 A process that has exited and has a parent, but has not yet been waited for by the parent, shall be  
 104173 marked **defunct**.

104174 Under the option **-f**, *ps* tries to determine the command name and arguments given when the  
 104175 process was created by examining memory or the swap area. Failing this, the command name, as  
 104176 it would appear without the option **-f**, is written in square brackets.

104177 The **-o** option allows the output format to be specified under user control.

104178 The application shall ensure that the format specification is a list of names presented as a single  
 104179 argument, <blank> or <comma>-separated. Each variable has a default header. The default  
 104180 header can be overridden by appending an <equals-sign> and the new text of the header. The  
 104181 rest of the characters in the argument shall be used as the header text. The fields specified shall  
 104182 be written in the order specified on the command line, and should be arranged in columns in the  
 104183 output. The field widths shall be selected by the system to be at least as wide as the header text  
 104184 (default or overridden value). If the header text is null, such as **-o user=**, the field width shall be  
 104185 at least as wide as the default header text. If all header text fields are null, no header line shall  
 104186 be written.

104187 The following names are recognized in the POSIX locale:

104188 **ruser** The real user ID of the process. This shall be the textual user ID, if it can be obtained  
 104189 and the field width permits, or a decimal representation otherwise.

|        |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------|---------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 104190 | <b>user</b>   | The effective user ID of the process. This shall be the textual user ID, if it can be obtained and the field width permits, or a decimal representation otherwise.                                                                                                                                                                                                                                                                                                                                                                                       |
| 104191 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104192 | <b>rgroup</b> | The real group ID of the process. This shall be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise.                                                                                                                                                                                                                                                                                                                                                                                          |
| 104193 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104194 | <b>group</b>  | The effective group ID of the process. This shall be the textual group ID, if it can be obtained and the field width permits, or a decimal representation otherwise.                                                                                                                                                                                                                                                                                                                                                                                     |
| 104195 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104196 | <b>pid</b>    | The decimal value of the process ID.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 104197 | <b>ppid</b>   | The decimal value of the parent process ID.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| 104198 | <b>pgid</b>   | The decimal value of the process group ID.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
| 104199 | <b>pcpu</b>   | The ratio of CPU time used recently to CPU time available in the same period, expressed as a percentage. The meaning of “recently” in this context is unspecified. The CPU time available is determined in an unspecified manner.                                                                                                                                                                                                                                                                                                                        |
| 104200 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104201 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104202 | <b>vsz</b>    | The size of the process in (virtual) memory in 1 024 byte units as a decimal integer.                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 104203 | <b>nice</b>   | The decimal value of the nice value of the process; see <i>nice</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| 104204 | <b>etime</b>  | In the POSIX locale, the elapsed time since the process was started, in the form:                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 104205 |               | [ <i>dd</i> - ] <i>hh</i> : <i>mm</i> : <i>ss</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 104206 |               | where <i>dd</i> shall represent the number of days, <i>hh</i> the number of hours, <i>mm</i> the number of minutes, and <i>ss</i> the number of seconds. The <i>dd</i> field shall be a decimal integer. The <i>hh</i> , <i>mm</i> , and <i>ss</i> fields shall be two-digit decimal integers padded on the left with zeros.                                                                                                                                                                                                                             |
| 104207 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104208 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104209 | <b>time</b>   | In the POSIX locale, the cumulative CPU time of the process in the form:                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 104210 |               | [ <i>dd</i> - ] <i>hh</i> : <i>mm</i> : <i>ss</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 104211 |               | The <i>dd</i> , <i>hh</i> , <i>mm</i> , and <i>ss</i> fields shall be as described in the <b>etime</b> specifier.                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 104212 | <b>tty</b>    | The name of the controlling terminal of the process (if any) in the same format used by the <i>who</i> utility.                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104213 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104214 | <b>comm</b>   | The name of the command being executed ( <i>argv</i> [0] value) as a string.                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 104215 | <b>args</b>   | The command with all its arguments as a string. The implementation may truncate this value to the field width; it is implementation-defined whether any further truncation occurs. It is unspecified whether the string represented is a version of the argument list as it was passed to the command when it started, or is a version of the arguments as they may have been modified by the application. Applications cannot depend on being able to modify their argument list and having that modification be reflected in the output of <i>ps</i> . |
| 104216 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104217 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104218 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104219 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104220 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104221 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104222 |               | Any field need not be meaningful in all implementations. In such a case a <hyphen-minus> (‘-’) should be output in place of the field value.                                                                                                                                                                                                                                                                                                                                                                                                             |
| 104223 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104224 |               | Only <b>comm</b> and <b>args</b> shall be allowed to contain <blank> characters; all others shall not. Any implementation-defined variables shall be specified in the system documentation along with the default header and indicating whether the field may contain <blank> characters.                                                                                                                                                                                                                                                                |
| 104225 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104226 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 104227 |               | The following table specifies the default header to be used in the POSIX locale corresponding to each format specifier.                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 104228 |               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |



Table 4-18 Variable Names and Default Headers in *ps*

| Format Specifier | Default Header | Format Specifier | Default Header |
|------------------|----------------|------------------|----------------|
| <b>args</b>      | <b>COMMAND</b> | <b>ppid</b>      | <b>PPID</b>    |
| <b>comm</b>      | <b>COMMAND</b> | <b>rgroup</b>    | <b>RGROUP</b>  |
| <b>etime</b>     | <b>ELAPSED</b> | <b>ruser</b>     | <b>RUSER</b>   |
| <b>group</b>     | <b>GROUP</b>   | <b>time</b>      | <b>TIME</b>    |
| <b>nice</b>      | <b>NI</b>      | <b>tty</b>       | <b>TT</b>      |
| <b>pcpu</b>      | <b>%CPU</b>    | <b>user</b>      | <b>USER</b>    |
| <b>pgid</b>      | <b>PGID</b>    | <b>vsz</b>       | <b>VSZ</b>     |
| <b>pid</b>       | <b>PID</b>     |                  |                |

**STDERR**

The standard error shall be used only for diagnostic messages.

**OUTPUT FILES**

None.

**EXTENDED DESCRIPTION**

None.

**EXIT STATUS**

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

**CONSEQUENCES OF ERRORS**

Default.

**APPLICATION USAGE**

Things can change while *ps* is running; the snapshot it gives is only true for an instant, and might not be accurate by the time it is displayed.

The **args** format specifier is allowed to produce a truncated version of the command arguments. In some implementations, this information is no longer available when the *ps* utility is executed.

If the field width is too narrow to display a textual ID, the system may use a numeric version. Normally, the system would be expected to choose large enough field widths, but if a large number of fields were selected to write, it might squeeze fields to their minimum sizes to fit on one line. One way to ensure adequate width for the textual IDs is to override the default header for a field to make it larger than most or all user or group names.

There is no special quoting mechanism for header text. The header text is the rest of the argument. If multiple header changes are needed, multiple **-o** options can be used, such as:

```
ps -o "user=User Name" -o pid=Process\ ID
```

On some implementations, especially multi-level secure systems, *ps* may be severely restricted and produce information only about child processes owned by the user.

**EXAMPLES**

The command:

```
ps -o user,pid,ppid=MOM -o args
```

writes at least the following in the POSIX locale:

```
USER    PID    MOM    COMMAND
```



104271 helene 34 12 ps -o uid,pid,ppid=MOM -o args

104272 The contents of the **COMMAND** field need not be the same in all implementations, due to  
104273 possible truncation.

#### 104274 RATIONALE

104275 There is very little commonality between BSD and System V implementations of *ps*. Many  
104276 options conflict or have subtly different usages. The standard developers attempted to select a  
104277 set of options for the base standard that were useful on a wide range of systems and selected  
104278 options that either can be implemented on both BSD and System V-based systems without  
104279 breaking the current implementations or where the options are sufficiently similar that any  
104280 changes would not be unduly problematic for users or implementors.

104281 It is recognized that on some implementations, especially multi-level secure systems, *ps* may be  
104282 nearly useless. The default output has therefore been chosen such that it does not break  
104283 historical implementations and also is likely to provide at least some useful information on most  
104284 systems.

104285 The major change is the addition of the format specification capability. The motivation for this  
104286 invention is to provide a mechanism for users to access a wider range of system information, if  
104287 the system permits it, in a portable manner. The fields chosen to appear in this volume of  
104288 POSIX.1-2008 were arrived at after considering what concepts were likely to be both reasonably  
104289 useful to the “average” user and had a reasonable chance of being implemented on a wide range  
104290 of systems. Again it is recognized that not all systems are able to provide all the information  
104291 and, conversely, some may wish to provide more. It is hoped that the approach adopted will be  
104292 sufficiently flexible and extensible to accommodate most systems. Implementations may be  
104293 expected to introduce new format specifiers.

104294 The default output should consist of a short listing containing the process ID, terminal name,  
104295 cumulative execution time, and command name of each process.

104296 The preference of the standard developers would have been to make the format specification an  
104297 operand of the *ps* command. Unfortunately, BSD usage precluded this.

104298 At one time a format was included to display the environment array of the process. This was  
104299 deleted because there is no portable way to display it.

104300 The **-A** option is equivalent to the BSD **-g** and the SVID **-e**. Because the two systems differed, a  
104301 mnemonic compromise was selected.

104302 The **-a** option is described with some optional behavior because the SVID omits session leaders,  
104303 but BSD does not.

104304 In an early proposal, format specifiers appeared for priority and start time. The former was not  
104305 defined adequately in this volume of POSIX.1-2008 and was removed in deference to the defined  
104306 nice value; the latter because elapsed time was considered to be more useful.

104307 In a new BSD version of *ps*, a **-O** option can be used to write all of the default information,  
104308 followed by additional format specifiers. This was not adopted because the default output is  
104309 implementation-defined. Nevertheless, this is a useful option that should be reserved for that  
104310 purpose. In the **-o** option for the POSIX Shell and Utilities *ps*, the format is the concatenation of  
104311 each **-o**. Therefore, the user can have an alias or function that defines the beginning of their  
104312 desired format and add more fields to the end of the output in certain cases where that would be  
104313 useful.

104314 The format of the terminal name is unspecified, but the descriptions of *ps*, *talk*, *who*, and *write*  
104315 require that they all use the same format.

104316 The **pcpu** field indicates that the CPU time available is determined in an unspecified manner.  
104317 This is because it is difficult to express an algorithm that is useful across all possible machine  
104318 architectures. Historical counterparts to this value have attempted to show percentage of use in  
104319 the recent past, such as the preceding minute. Frequently, these values for all processes did not  
104320 add up to 100%. Implementations are encouraged to provide data in this field to users that will  
104321 help them identify processes currently affecting the performance of the system.

104322 **FUTURE DIRECTIONS**

104323 None.

104324 **SEE ALSO**

104325 *kill, nice, renice*

104326 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

104327 **CHANGE HISTORY**

104328 First released in Issue 2.

104329 **Issue 6**

104330 This utility is marked as part of the User Portability Utilities option.

104331 The normative text is reworded to avoid use of the term “must” for application requirements.

104332 The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

104333 **Issue 7**

104334 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

104335 SD5-XCU-ERN-148 is applied, updating the OPTIONS section.

104336 **NAME**104337        `pwd` — return working directory name104338 **SYNOPSIS**104339        `pwd [-L|-P]`104340 **DESCRIPTION**

104341        The *pwd* utility shall write to standard output an absolute pathname of the current working  
 104342        directory, which does not contain the filenames dot or dot-dot.

104343 **OPTIONS**104344        The *pwd* utility shall conform to XBD [Section 12.2](#) (on page 216).

104345        The following options shall be supported by the implementation:

104346        **-L**        If the *PWD* environment variable contains an absolute pathname of the current |  
 104347        directory and the pathname does not contain any components that are dot or dot- |  
 104348        dot, *pwd* shall write this pathname to standard output, except that if the *PWD* |  
 104349        environment variable is longer than {PATH\_MAX} bytes including the terminating |  
 104350        null, it is unspecified whether *pwd* writes this pathname to standard output or |  
 104351        behaves as if the **-P** option had been specified. Otherwise, the **-L** option shall |  
 104352        behave as the **-P** option.

104353        **-P**        The pathname written to standard output shall not contain any components that  
 104354        refer to files of type symbolic link. If there are multiple pathnames that the *pwd*  
 104355        utility could write to standard output, one beginning with a single <slash>  
 104356        character and one or more beginning with two <slash> characters, then it shall  
 104357        write the pathname beginning with a single <slash> character. The pathname shall  
 104358        not contain any unnecessary <slash> characters after the leading one or two  
 104359        <slash> characters.

104360        If both **-L** and **-P** are specified, the last one shall apply. If neither **-L** nor **-P** is specified, the *pwd*  
 104361        utility shall behave as if **-L** had been specified.

104362 **OPERANDS**

104363        None.

104364 **STDIN**

104365        Not used.

104366 **INPUT FILES**

104367        None.

104368 **ENVIRONMENT VARIABLES**104369        The following environment variables shall affect the execution of *pwd*:

104370        *LANG*        Provide a default value for the internationalization variables that are unset or null.  
 104371        (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 104372        used to determine the values of locale categories.)

104373        *LC\_ALL*        If set to a non-empty string value, override the values of all the other  
 104374        internationalization variables.

104375        *LC\_MESSAGES*

104376        Determine the locale that should be used to affect the format and contents of  
 104377        diagnostic messages written to standard error.

104378 *XSI*        *NLSPATH*        Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

104379 *PWD* An absolute pathname of the current working directory. If an application sets or  
104380 unsets the value of *PWD*, the behavior of *pwd* is unspecified.

#### 104381 **ASYNCHRONOUS EVENTS**

104382 Default.

#### 104383 **STDOUT**

104384 The *pwd* utility output is an absolute pathname of the current working directory:

104385 "%s\n", <directory pathname>

#### 104386 **STDERR**

104387 The standard error shall be used only for diagnostic messages.

#### 104388 **OUTPUT FILES**

104389 None.

#### 104390 **EXTENDED DESCRIPTION**

104391 None.

#### 104392 **EXIT STATUS**

104393 The following exit values shall be returned:

104394 0 Successful completion.

104395 >0 An error occurred.

#### 104396 **CONSEQUENCES OF ERRORS**

104397 If an error is detected, output shall not be written to standard output, a diagnostic message shall  
104398 be written to standard error, and the exit status is not zero.

#### 104399 **APPLICATION USAGE**

104400 If the pathname obtained from *pwd* is longer than {PATH\_MAX} bytes, it could produce an error  
104401 if passed to *cd*. Therefore, in order to return to that directory it may be necessary to break the  
104402 pathname into sections shorter than {PATH\_MAX} and call *cd* on each section in turn (the first  
104403 section being an absolute pathname and subsequent sections being relative pathnames).

#### 104404 **EXAMPLES**

104405 None.

#### 104406 **RATIONALE**

104407 Some implementations have historically provided *pwd* as a shell special built-in command.

104408 In most utilities, if an error occurs, partial output may be written to standard output. This does  
104409 not happen in historical implementations of *pwd*. Because *pwd* is frequently used in historical  
104410 shell scripts without checking the exit status, it is important that the historical behavior is  
104411 required here; therefore, the CONSEQUENCES OF ERRORS section specifically disallows any  
104412 partial output being written to standard output.

104413 An earlier version of this standard stated that the *PWD* environment variable was affected when  
104414 the *-P* option was in effect. This was incorrect; conforming implementations do not do this.

#### 104415 **FUTURE DIRECTIONS**

104416 None.

#### 104417 **SEE ALSO**

104418 *cd*

104419 XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

104420 XSH *getcwd()*

104421 **CHANGE HISTORY**

104422 First released in Issue 2.

104423 **Issue 6**104424 The **-P** and **-L** options are added to describe actions relating to symbolic links as specified in the  
104425 IEEE P1003.2b draft standard.104426 **Issue 7**

104427 Austin Group Interpretation 1003.1-2001 #097 is applied.

104428 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

104429 Changes to the *pwd* utility and *PWD* environment variable have been made to match the  
104430 changes to the *getcwd()* function made for Austin Group Interpretation 1003.1-2001 #140.

## 104431 NAME

104432 qalter — alter batch job

## 104433 SYNOPSIS

```
104434 OB BE qalter [-a date_time] [-A account_string] [-c interval] [-e path_name]
104435          [-h hold_list] [-j join_list] [-k keep_list] [-l resource_list]
104436          [-m mail_options] [-M mail_list] [-N name] [-o path_name]
104437          [-p priority] [-r y|n] [-S path_name_list] [-u user_list]
104438          job_identifier...
```

## 104439 DESCRIPTION

104440 The attributes of a batch job are altered by a request to the batch server that manages the batch  
 104441 job. The *qalter* utility is a user-accessible batch client that requests the alteration of the attributes  
 104442 of one or more batch jobs.

104443 The *qalter* utility shall alter the attributes of those batch jobs, and only those batch jobs, for which  
 104444 a batch *job\_identifier* is presented to the utility.

104445 The *qalter* utility shall alter the attributes of batch jobs in the order in which the batch  
 104446 *job\_identifiers* are presented to the utility.

104447 If the *qalter* utility fails to process a batch *job\_identifier* successfully, the utility shall proceed to  
 104448 process the remaining batch *job\_identifiers*, if any.

104449 For each batch *job\_identifier* for which the *qalter* utility succeeds, each attribute of the identified  
 104450 batch job shall be altered as indicated by all the options presented to the utility.

104451 For each identified batch job for which the *qalter* utility fails, the utility shall not alter any  
 104452 attribute of the batch job.

104453 For each batch job that the *qalter* utility processes, the utility shall not modify any attribute other  
 104454 than those required by the options and option-arguments presented to the utility.

104455 The *qalter* utility shall alter batch jobs by sending a *Modify Job Request* to the batch server that  
 104456 manages each batch job. At the time the *qalter* utility exits, it shall have modified the batch job  
 104457 corresponding to each successfully processed batch *job\_identifier*. An attempt to alter the  
 104458 attributes of a batch job in the RUNNING state is implementation-defined.

## 104459 OPTIONS

104460 The *qalter* utility shall conform to XBD [Section 12.2](#) (on page 216).

104461 The following options shall be supported by the implementation:

104462 **-a date\_time** Redefine the time at which the batch job becomes eligible for execution.

104463 The *date\_time* argument shall be in the same form and represent the same time as  
 104464 for the *touch* utility. The time so represented shall be set into the *Execution\_Time*  
 104465 attribute of the batch job. If the time specified is earlier than the current time, the  
 104466 **-a** option shall have no effect.

104467 **-A account\_string**

104468 Redefine the account to which the resource consumption of the batch job should be  
 104469 charged.

104470 The syntax of the *account\_string* option-argument is unspecified.

104471 The *qalter* utility shall set the *Account\_Name* attribute of the batch job to the value  
 104472 of the *account\_string* option-argument.

|        |                     |                                                                                                          |
|--------|---------------------|----------------------------------------------------------------------------------------------------------|
| 104473 | <b>-c interval</b>  | Redefine whether the batch job should be checkpointed, and if so, how often.                             |
| 104474 |                     | The <i>qalter</i> utility shall accept a value for the interval option-argument that is one of           |
| 104475 |                     | the following:                                                                                           |
| 104476 | n                   | No checkpointing is to be performed on the batch job                                                     |
| 104477 |                     | (NO_CHECKPOINT).                                                                                         |
| 104478 | s                   | Checkpointing is to be performed only when the batch server is shut                                      |
| 104479 |                     | down (CHECKPOINT_AT_SHUTDOWN).                                                                           |
| 104480 | c                   | Automatic periodic checkpointing is to be performed at the                                               |
| 104481 |                     | <i>Minimum_Cpu_Interval</i> attribute of the batch queue, in units of CPU                                |
| 104482 |                     | minutes (CHECKPOINT_AT_MIN_CPU_INTERVAL).                                                                |
| 104483 | c=minutes           | Automatic periodic checkpointing is to be performed every <i>minutes</i>                                 |
| 104484 |                     | of CPU time, or every <i>Minimum_Cpu_Interval</i> minutes, whichever is                                  |
| 104485 |                     | greater. The <i>minutes</i> argument shall conform to the syntax for                                     |
| 104486 |                     | unsigned integers and shall be greater than zero.                                                        |
| 104487 |                     | An implementation may define other checkpoint intervals. The conformance                                 |
| 104488 |                     | document for an implementation shall describe any alternative checkpoint                                 |
| 104489 |                     | intervals, how they are specified, their internal behavior, and how they affect the                      |
| 104490 |                     | behavior of the utility.                                                                                 |
| 104491 |                     | The <i>qalter</i> utility shall set the <i>Checkpoint</i> attribute of the batch job to the value of the |
| 104492 |                     | <i>interval</i> option-argument.                                                                         |
| 104493 | <b>-e path_name</b> |                                                                                                          |
| 104494 |                     | Redefine the path to be used for the standard error stream of the batch job.                             |
| 104495 |                     | The <i>qalter</i> utility shall accept a <i>path_name</i> option-argument that conforms to the           |
| 104496 |                     | syntax of the <i>path_name</i> element defined in the System Interfaces volume of                        |
| 104497 |                     | POSIX.1-2008, which can be preceded by a host name element of the form                                   |
| 104498 |                     | <i>hostname</i> ..                                                                                       |
| 104499 |                     | If the <i>path_name</i> option-argument constitutes an absolute pathname, the <i>qalter</i>              |
| 104500 |                     | utility shall set the <i>Error_Path</i> attribute of the batch job to the value of the                   |
| 104501 |                     | <i>path_name</i> option-argument, including the host name element, if present.                           |
| 104502 |                     | If the <i>path_name</i> option-argument constitutes a relative pathname and no host                      |
| 104503 |                     | name element is specified, the <i>qalter</i> utility shall set the <i>Error_Path</i> attribute of the    |
| 104504 |                     | batch job to the value of the absolute pathname derived by expanding the                                 |
| 104505 |                     | <i>path_name</i> option-argument relative to the current directory of the process that                   |
| 104506 |                     | executes the <i>qalter</i> utility.                                                                      |
| 104507 |                     | If the <i>path_name</i> option-argument constitutes a relative pathname and a host name                  |
| 104508 |                     | element is specified, the <i>qalter</i> utility shall set the <i>Error_Path</i> attribute of the batch   |
| 104509 |                     | job to the value of the option-argument without expansion.                                               |
| 104510 |                     | If the <i>path_name</i> option-argument does not include a host name element, the <i>qalter</i>          |
| 104511 |                     | utility shall prefix the pathname in the <i>Error_Path</i> attribute with <i>hostname</i> ., where       |
| 104512 |                     | <i>hostname</i> is the name of the host upon which the <i>qalter</i> utility is being executed.          |
| 104513 | <b>-h hold_list</b> | Redefine the types of holds, if any, on the batch job. The <i>qalter</i> <b>-h</b> option shall          |
| 104514 |                     | accept a value for the <i>hold_list</i> option-argument that is a string of alphanumeric                 |
| 104515 |                     | characters in the portable character set.                                                                |
| 104516 |                     | The <i>qalter</i> utility shall accept a value for the <i>hold_list</i> option-argument that is a        |



104517 string of one or more of the characters 'u', 's', or 'o', or the single character  
 104518 'n'. For each unique character in the *hold\_list* option-argument, the *qalter* utility  
 104519 shall add a value to the *Hold\_Types* attribute of the batch job as follows, each  
 104520 representing a different hold type:

- 104521 u USER
- 104522 s SYSTEM
- 104523 o OPERATOR

104524 If any of these characters are duplicated in the *hold\_list* option-argument, the  
 104525 duplicates shall be ignored. An existing *Hold\_Types* attribute can be cleared by the  
 104526 hold type:

- 104527 n NO\_HOLD

104528 The *qalter* utility shall consider it an error if any hold type other than 'n' is  
 104529 combined with hold type 'n'. Strictly conforming applications shall not repeat  
 104530 any of the characters 'u', 's', 'o', or 'n' within the *hold\_list* option-argument.  
 104531 The *qalter* utility shall permit the repetition of characters, but shall not assign  
 104532 additional meaning to the repeated characters. An implementation may define  
 104533 other hold types. The conformance document for an implementation shall describe  
 104534 any additional hold types, how they are specified, their internal behavior, and how  
 104535 they affect the behavior of the utility.

104536 *-j join\_list* Redefine which streams of the batch job are to be merged. The *qalter -j* option shall  
 104537 accept a value for the *join\_list* option-argument that is a string of alphanumeric  
 104538 characters in the portable character set.

104539 The *qalter* utility shall accept a *join\_list* option-argument that consists of one or  
 104540 more of the characters 'e' and 'o', or the single character 'n'.

104541 All of the other batch job output streams specified shall be merged into the output  
 104542 stream represented by the character listed first in the *join\_list* option-argument.

104543 For each unique character in the *join\_list* option-argument, the *qalter* utility shall  
 104544 add a value to the *Join\_Path* attribute of the batch job as follows, each representing  
 104545 a different batch job stream to join:

- 104546 e The standard error of the batch job (JOIN\_STD\_ERROR).
- 104547 o The standard output of the batch job (JOIN\_STD\_OUTPUT).

104548 An existing *Join\_Path* attribute can be cleared by the join type:

- 104549 n NO\_JOIN

104550 If 'n' is specified, then no files are joined. The *qalter* utility shall consider it an  
 104551 error if any join type other than 'n' is combined with join type 'n'.

104552 Strictly conforming applications shall not repeat any of the characters 'e', 'o', or  
 104553 'n' within the *join\_list* option-argument. The *qalter* utility shall permit the  
 104554 repetition of characters, but shall not assign additional meaning to the repeated  
 104555 characters.

104556 An implementation may define other join types. The conformance document for an  
 104557 implementation shall describe any additional batch job streams, how they are  
 104558 specified, their internal behavior, and how they affect the behavior of the utility.



104559       **-k keep\_list**   Redefine which output of the batch job to retain on the execution host.

104560               The *qalter* **-k** option shall accept a value for the *keep\_list* option-argument that is a

104561               string of alphanumeric characters in the portable character set.

104562               The *qalter* utility shall accept a *keep\_list* option-argument that consists of one or

104563               more of the characters 'e' and 'o', or the single character 'n'.

104564               For each unique character in the *keep\_list* option-argument, the *qalter* utility shall

104565               add a value to the *Keep\_Files* attribute of the batch job as follows, each representing

104566               a different batch job stream to keep:

104567               e    The standard error of the batch job (KEEP\_STD\_ERROR).

104568               o    The standard output of the batch job (KEEP\_STD\_OUTPUT).

104569               If both 'e' and 'o' are specified, then both files are retained. An existing

104570               *Keep\_Files* attribute can be cleared by the keep type:

104571               n    NO\_KEEP

104572               If 'n' is specified, then no files are retained. The *qalter* utility shall consider it an

104573               error if any keep type other than 'n' is combined with keep type 'n'.

104574               Strictly conforming applications shall not repeat any of the characters 'e', 'o', or

104575               'n' within the *keep\_list* option-argument. The *qalter* utility shall permit the

104576               repetition of characters, but shall not assign additional meaning to the repeated

104577               characters. An implementation may define other keep types. The conformance

104578               document for an implementation shall describe any additional keep types, how

104579               they are specified, their internal behavior, and how they affect the behavior of the

104580               utility.

104581       **-l resource\_list**   Redefine the resources that are allowed or required by the batch job.

104582               The *qalter* utility shall accept a *resource\_list* option-argument that conforms to the

104583               following syntax:

104584               resource=value[, resource=value, ...]

104585               The *qalter* utility shall set one entry in the value of the *Resource\_List* attribute of the

104586               batch job for each resource listed in the *resource\_list* option-argument.

104587               Because the list of supported resource names might vary by batch server, the *qalter*

104588               utility shall rely on the batch server to validate the resource names and associated

104589               values. See [Section 3.3.3](#) (on page 2441) for a means of removing *keyword=value*

104590               (and *value@keyword*) pairs and other general rules for list-oriented batch job

104591               attributes.

104592               

104593       **-m mail\_options**   Redefine the points in the execution of the batch job at which the batch server is to

104594               send mail about a change in the state of the batch job.

104595               

104596               The *qalter* **-m** option shall accept a value for the *mail\_options* option-argument that

104597               is a string of alphanumeric characters in the portable character set.

104598               The *qalter* utility shall accept a value for the *mail\_options* option-argument that is a

104599               string of one or more of the characters 'e', 'b', and 'a', or the single character

104600               'n'. For each unique character in the *mail\_options* option-argument, the *qalter*

104601               utility shall add a value to the *Mail\_Users* attribute of the batch job as follows, each

104602 representing a different time during the life of a batch job at which to send mail:

104603 e MAIL\_AT\_EXIT

104604 b MAIL\_AT\_BEGINNING

104605 a MAIL\_AT\_ABORT

104606 If any of these characters are duplicated in the *mail\_options* option-argument, the

104607 duplicates shall be ignored.

104608 An existing *Mail\_Points* attribute can be cleared by the mail type:

104609 n NO\_MAIL

104610 If 'n' is specified, then mail is not sent. The *qalter* utility shall consider it an error

104611 if any mail type other than 'n' is combined with mail type 'n'. Strictly

104612 conforming applications shall not repeat any of the characters 'e', 'b', 'a', or

104613 'n' within the *mail\_options* option-argument. The *qalter* utility shall permit the

104614 repetition of characters but shall not assign additional meaning to the repeated

104615 characters.

104616 An implementation may define other mail types. The conformance document for

104617 an implementation shall describe any additional mail types, how they are

104618 specified, their internal behavior, and how they affect the behavior of the utility.

104619 **-M *mail\_list*** Redefine the list of users to which the batch server that executes the batch job is to

104620 send mail, if the batch server sends mail about the batch job.

104621 The syntax of the *mail\_list* option-argument is unspecified. If the implementation

104622 of the *qalter* utility uses a name service to locate users, the utility shall accept the

104623 syntax used by the name service.

104624 If the implementation of the *qalter* utility does not use a name service to locate

104625 users, the implementation shall accept the following syntax for user names:

104626 *mail\_address*[ , *mail\_address* , ... ]

104627 The interpretation of *mail\_address* is implementation-defined.

104628 The *qalter* utility shall set the *Mail\_Users* attribute of the batch job to the value of

104629 the *mail\_list* option-argument.

104630 **-N *name*** Redefine the name of the batch job.

104631 The *qalter* -N option shall accept a value for the *name* option-argument that is a

104632 string of up to 15 alphanumeric characters in the portable character set where the

104633 first character is alphabetic.

104634 The syntax of the *name* option-argument is unspecified.

104635 The *qalter* utility shall set the *Job\_Name* attribute of the batch job to the value of the

104636 *name* option-argument.

104637 **-o *path\_name***

104638 Redefine the path for the standard output of the batch job.

104639 The *qalter* utility shall accept a *path\_name* option-argument that conforms to the

104640 syntax of the *path\_name* element defined in the System Interfaces volume of

104641 POSIX.1-2008, which can be preceded by a host name element of the form

104642 *hostname*::.

|        |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
|--------|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 104643 |                          | If the <i>path_name</i> option-argument constitutes an absolute pathname, the <i>qalter</i> utility shall set the <i>Output_Path</i> attribute of the batch job to the value of the <i>path_name</i> option-argument.                                                                                                                                                             |
| 104644 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104645 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104646 |                          | If the <i>path_name</i> option-argument constitutes a relative pathname and no host name element is specified, the <i>qalter</i> utility shall set the <i>Output_Path</i> attribute of the batch job to the absolute pathname derived by expanding the <i>path_name</i> option-argument relative to the current directory of the process that executes the <i>qalter</i> utility. |
| 104647 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104648 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104649 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104650 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104651 |                          | If the <i>path_name</i> option-argument constitutes a relative pathname and a host name element is specified, the <i>qalter</i> utility shall set the <i>Output_Path</i> attribute of the batch job to the value of the <i>path_name</i> option-argument without any expansion of the pathname.                                                                                   |
| 104652 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104653 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104654 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104655 |                          | If the <i>path_name</i> option-argument does not include a host name element, the <i>qalter</i> utility shall prefix the pathname in the <i>Output_Path</i> attribute with <i>hostname:</i> , where <i>hostname</i> is the name of the host upon which the <i>qalter</i> utility is being executed.                                                                               |
| 104656 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104657 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104658 | <b>-p priority</b>       | Redefine the priority of the batch job.                                                                                                                                                                                                                                                                                                                                           |
| 104659 |                          | The <i>qalter</i> utility shall accept a value for the priority option-argument that conforms to the syntax for signed decimal integers, and which is not less than -1 024 and not greater than 1 023.                                                                                                                                                                            |
| 104660 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104661 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104662 |                          | The <i>qalter</i> utility shall set the <i>Priority</i> attribute of the batch job to the value of the <i>priority</i> option-argument.                                                                                                                                                                                                                                           |
| 104663 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104664 | <b>-r y   n</b>          | Redefine whether the batch job is rerunnable.                                                                                                                                                                                                                                                                                                                                     |
| 104665 |                          | If the value of the option-argument is 'y', the <i>qalter</i> utility shall set the <i>Rerunable</i> attribute of the batch job to TRUE.                                                                                                                                                                                                                                          |
| 104666 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104667 |                          | If the value of the option-argument is 'n', the <i>qalter</i> utility shall set the <i>Rerunable</i> attribute of the batch job to FALSE.                                                                                                                                                                                                                                         |
| 104668 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104669 |                          | The <i>qalter</i> utility shall consider it an error if any character other than 'y' or 'n' is specified in the option-argument.                                                                                                                                                                                                                                                  |
| 104670 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104671 | <b>-S path_name_list</b> |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104672 |                          | Redefine the shell that interprets the script at the destination system.                                                                                                                                                                                                                                                                                                          |
| 104673 |                          | The <i>qalter</i> utility shall accept a <i>path_name_list</i> option-argument that conforms to the following syntax:                                                                                                                                                                                                                                                             |
| 104674 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104675 |                          | pathname[@host] [ , pathname[@host] , . . . ]                                                                                                                                                                                                                                                                                                                                     |
| 104676 |                          | The <i>qalter</i> utility shall accept only one pathname that is missing a corresponding host name. The <i>qalter</i> utility shall allow only one pathname per named host.                                                                                                                                                                                                       |
| 104677 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104678 |                          | The <i>qalter</i> utility shall add a value to the <i>Shell_Path_List</i> attribute of the batch job for each entry in the <i>path_name_list</i> option-argument. See <a href="#">Section 3.3.3</a> (on page 2441) for a means of removing <i>keyword=value</i> (and <i>value@keyword</i> ) pairs and other general rules for list-oriented batch job attributes.                 |
| 104679 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104680 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104681 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104682 | <b>-u user_list</b>      | Redefine the user name under which the batch job is to run at the destination system.                                                                                                                                                                                                                                                                                             |
| 104683 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |
| 104684 |                          | The <i>qalter</i> utility shall accept a <i>user_list</i> option-argument that conforms to the following syntax:                                                                                                                                                                                                                                                                  |
| 104685 |                          |                                                                                                                                                                                                                                                                                                                                                                                   |

104686 username[@host][, , username[@host], , . . .]

104687 The *qalter* utility shall accept only one user name that is missing a corresponding  
104688 host name. The *qalter* utility shall accept only one user name per named host.

104689 The *qalter* utility shall add a value to the *User\_List* attribute of the batch job for each  
104690 entry in the *user\_list* option-argument. See [Section 3.3.3](#) (on page 2441) for a means  
104691 of removing *keyword=value* (and *value@keyword*) pairs and other general rules for  
104692 list-oriented batch job attributes.

#### 104693 OPERANDS

104694 The *qalter* utility shall accept one or more operands that conform to the syntax for a batch  
104695 *job\_identifier* (see [Section 3.3.1](#), on page 2439).

#### 104696 STDIN

104697 Not used.

#### 104698 INPUT FILES

104699 None.

#### 104700 ENVIRONMENT VARIABLES

104701 The following environment variables shall affect the execution of *qalter*:

104702 *LANG* Provide a default value for the internationalization variables that are unset or null.  
104703 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
104704 used to determine the values of locale categories.)

104705 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
104706 internationalization variables.

104707 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
104708 characters (for example, single-byte as opposed to multi-byte characters in  
104709 arguments).

104710 *LC\_MESSAGES*

104711 Determine the locale that should be used to affect the format and contents of  
104712 diagnostic messages written to standard error.

104713 *LOGNAME* Determine the login name of the user.

104714 *TZ* Determine the timezone used to interpret the *date-time* option-argument. If *TZ* is  
104715 unset or null, an unspecified default timezone shall be used.

#### 104716 ASYNCHRONOUS EVENTS

104717 Default.

#### 104718 STDOUT

104719 None.

#### 104720 STDERR

104721 The standard error shall be used only for diagnostic messages.

#### 104722 OUTPUT FILES

104723 None.

#### 104724 EXTENDED DESCRIPTION

104725 None.

**104726 EXIT STATUS**

104727 The following exit values shall be returned:

104728 0 Successful completion.

104729 >0 An error occurred.

**104730 CONSEQUENCES OF ERRORS**

104731 In addition to the default behavior, the *qalter* utility shall not be required to write a diagnostic  
104732 message to standard error when the error reply received from a batch server indicates that the  
104733 batch *job\_identifier* does not exist on the server. Whether or not the *qalter* utility attempts to locate  
104734 the batch job on other batch servers is implementation-defined.

**104735 APPLICATION USAGE**

104736 None.

**104737 EXAMPLES**

104738 None.

**104739 RATIONALE**

104740 The *qalter* utility allows users to change the attributes of a batch job.

104741 As a means of altering a queued job, the *qalter* utility is superior to deleting and requeuing the  
104742 batch job insofar as an altered job retains its place in the queue with some traditional selection  
104743 algorithms. In addition, the *qalter* utility is both shorter and simpler than a sequence of *qdel* and  
104744 *qsub* utilities.

104745 The result of an attempt on the part of a user to alter a batch job in a RUNNING state is  
104746 implementation-defined because a batch job in the RUNNING state will already have opened its  
104747 output files and otherwise performed any actions indicated by the options in effect at the time  
104748 the batch job began execution.

104749 The options processed by the *qalter* utility are identical to those of the *qsub* utility, with a few  
104750 exceptions: *-V*, *-v*, and *-q*. The *-V* and *-v* are inappropriate for the *qalter* utility, since they  
104751 capture potentially transient environment information from the submitting process. The *-q*  
104752 option would specify a new queue, which would largely negate the previously stated advantage  
104753 of using *qalter*; furthermore, the *qmove* utility provides a superior means of moving jobs.

104754 Each of the following paragraphs provides the rationale for a *qalter* option.

104755 Additional rationale concerning these options can be found in the rationale for the *qsub* utility.

104756 The *-a* option allows users to alter the date and time at which a batch job becomes eligible to  
104757 run.

104758 The *-A* option allows users to change the account that will be charged for the resources  
104759 consumed by the batch job. Support for the *-A* option is mandatory for conforming  
104760 implementations of *qalter*, even though support of accounting is optional for servers. Whether or  
104761 not to support accounting is left to the implementor of the server, but mandatory support of the  
104762 *-A* option assures users of a consistent interface and allows them to control accounting on  
104763 servers that support accounting.

104764 The *-c* option allows users to alter the checkpointing interval of a batch job. A checkpointing  
104765 system, which is not defined by POSIX.1-2008, allows recovery of a batch job at the most recent  
104766 checkpoint in the event of a crash. Checkpointing is typically used for jobs that consume  
104767 expensive computing time or must meet a critical schedule. Users should be allowed to make  
104768 the tradeoff between the overhead of checkpointing and the risk to the timely completion of the  
104769 batch job; therefore, this volume of POSIX.1-2008 provides the checkpointing interval option.  
104770 Support for checkpointing is optional for servers.

The **-e** option allows users to alter the name and location of the standard error stream written by a batch job. However, the path of the standard error stream is meaningless if the value of the *Join\_Path* attribute of the batch job is TRUE.

The **-h** option allows users to set the hold type in the *Hold\_Types* attribute of a batch job. The *qhold* and *qrls* utilities add or remove hold types to the *Hold\_Types* attribute, respectively. The **-h** option has been modified to allow for implementation-defined hold types.

The **-j** option allows users to alter the decision to join (merge) the standard error stream of the batch job with the standard output stream of the batch job.

The **-l** option allows users to change the resource limits imposed on a batch job.

The **-m** option allows users to modify the list of points in the life of a batch job at which the designated users will receive mail notification.

The **-M** option allows users to alter the list of users who will receive notification about events in the life of a batch job.

The **-N** option allows users to change the name of a batch job.

The **-o** option allows users to alter the name and path to which the standard output stream of the batch job will be written.

The **-P** option allows users to modify the priority of a batch job. Support for priority is optional for batch servers.

The **-r** option allows users to alter the rerunability status of a batch job.

The **-S** option allows users to change the name and location of the shell image that will be invoked to interpret the script of the batch job. This option has been modified to allow a list of shell name and locations associated with different hosts.

The **-u** option allows users to change the user identifier under which the batch job will execute.

The *job\_identifier* operand syntax is provided so that the user can differentiate between the originating and destination (or executing) batch server. These may or may not be the same. The *.server\_name* portion identifies the originating batch server, while the *@server* portion identifies the destination batch server.

Historically, the *qalter* utility has been a component of the Network Queuing System (NQS), the existing practice from which this utility has been derived.

#### FUTURE DIRECTIONS

The *qalter* utility may be removed in a future version.

#### SEE ALSO

Chapter 3 (on page 2417), *qdel*, *qhold*, *qmove*, *qrls*, *qsub*, *touch*

XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

#### CHANGE HISTORY

Derived from IEEE Std 1003.2d-1994.

#### Issue 6

The *TZ* entry is added to the ENVIRONMENT VARIABLES section.

IEEE PASC Interpretation 1003.2 #182 is applied, clarifying the description of the **-a** option.

104810 **Issue 7**104811 The *qalter* utility is marked obsolescent.

104812 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



104813 **NAME**

104814 qdel — delete batch jobs

104815 **SYNOPSIS**104816 OB BE `qdel job_identifier...`104817 **DESCRIPTION**

104818 A batch job is deleted by sending a request to the batch server that manages the batch job. A  
 104819 batch job that has been deleted is no longer subject to management by batch services.

104820 The *qdel* utility is a user-accessible client of batch services that requests the deletion of one or  
 104821 more batch jobs.

104822 The *qdel* utility shall request a batch server to delete those batch jobs for which a batch  
 104823 *job\_identifier* is presented to the utility.

104824 The *qdel* utility shall delete batch jobs in the order in which their batch *job\_identifiers* are  
 104825 presented to the utility.

104826 If the *qdel* utility fails to process any batch *job\_identifier* successfully, the utility shall proceed to  
 104827 process the remaining batch *job\_identifiers*, if any.

104828 The *qdel* utility shall delete each batch job by sending a *Delete Job Request* to the batch server that  
 104829 manages the batch job.

104830 The *qdel* utility shall not exit until the batch job corresponding to each successfully processed  
 104831 batch *job\_identifier* has been deleted.

104832 **OPTIONS**

104833 None.

104834 **OPERANDS**

104835 The *qdel* utility shall accept one or more operands that conform to the syntax for a batch  
 104836 *job\_identifier* (see [Section 3.3.1](#), on page 2439).

104837 **STDIN**

104838 Not used.

104839 **INPUT FILES**

104840 None.

104841 **ENVIRONMENT VARIABLES**104842 The following environment variables shall affect the execution of *qdel*:

104843 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 104844 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 104845 used to determine the values of locale categories.)

104846 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 104847 internationalization variables.

104848 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 104849 characters (for example, single-byte as opposed to multi-byte characters in  
 104850 arguments).

104851 *LC\_MESSAGES*

104852 Determine the locale that should be used to affect the format and contents of  
 104853 diagnostic messages written to standard error.



104854 *LOGNAME* Determine the login name of the user.

104855 **ASYNCHRONOUS EVENTS**

104856 Default.

104857 **STDOUT**

104858 An implementation of the *qdel* utility may write informative messages to standard output.

104859 **STDERR**

104860 The standard error shall be used only for diagnostic messages.

104861 **OUTPUT FILES**

104862 None.

104863 **EXTENDED DESCRIPTION**

104864 None.

104865 **EXIT STATUS**

104866 The following exit values shall be returned:

104867 0 Successful completion.

104868 >0 An error occurred.

104869 **CONSEQUENCES OF ERRORS**

104870 In addition to the default behavior, the *qdel* utility shall not be required to write a diagnostic  
104871 message to standard error when the error reply received from a batch server indicates that the  
104872 batch *job\_identifier* does not exist on the server. Whether or not the *qdel* utility waits to output the  
104873 diagnostic message while attempting to locate the job on other servers is implementation-  
104874 defined.

104875 **APPLICATION USAGE**

104876 None.

104877 **EXAMPLES**

104878 None.

104879 **RATIONALE**

104880 The *qdel* utility allows users and administrators to delete jobs.

104881 The *qdel* utility provides functionality that is not otherwise available. For example, the *kill* utility  
104882 of the operating system does not suffice. First, to use the *kill* utility, the user might have to log in  
104883 on a remote node, because the *kill* utility does not operate across the network. Second, unlike  
104884 *qdel*, *kill* cannot remove jobs from queues. Lastly, the arguments of the *qdel* utility are job  
104885 identifiers rather than process identifiers, and so this utility can be passed the output of the  
104886 *qselect* utility, thus providing users with a means of deleting a list of jobs.

104887 Because a set of jobs can be selected using the *qselect* utility, the *qdel* utility has not been  
104888 complicated with options that provide for selection of jobs. Instead, the batch jobs to be deleted  
104889 are identified individually by their job identifiers.

104890 Historically, the *qdel* utility has been a component of NQS, the existing practice on which it is  
104891 based. However, the *qdel* utility defined in this volume of POSIX.1-2008 does not provide an  
104892 option for specifying a signal number to send to the batch job prior to the killing of the process;  
104893 that capability has been subsumed by the *qsig* utility.

104894 A discussion was held about the delays of networking and the possibility that the batch server  
104895 may never respond, due to a down router, down batch server, or other network mishap. The  
104896 DESCRIPTION records this under the words “fails to process any job identifier”. In the broad  
104897 sense, the network problem is also an error, which causes the failure to process the batch job

104898 identifier.

104899 **FUTURE DIRECTIONS**

104900 The *qdel* utility may be removed in a future version.

104901 **SEE ALSO**

104902 [Chapter 3](#) (on page 2417), *kill*, *qselect*, *qsig*

104903 XBD [Chapter 8](#) (on page 173)

104904 **CHANGE HISTORY**

104905 Derived from IEEE Std 1003.2d-1994.

104906 **Issue 6**

104907 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

104908 **Issue 7**

104909 The *qdel* utility is marked obsolescent.

104910 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

104911 **NAME**

104912 qhold — hold batch jobs

104913 **SYNOPSIS**104914 OB BE `qhold [-h hold_list] job_identifier...`104915 **DESCRIPTION**

104916 A hold is placed on a batch job by a request to the batch server that manages the batch job. A  
 104917 batch job that has one or more holds is not eligible for execution. The *qhold* utility is a user-  
 104918 accessible client of batch services that requests one or more types of hold to be placed on one or  
 104919 more batch jobs.

104920 The *qhold* utility shall place holds on those batch jobs for which a batch *job\_identifier* is presented  
 104921 to the utility.

104922 The *qhold* utility shall place holds on batch jobs in the order in which their batch *job\_identifiers*  
 104923 are presented to the utility. If the *qhold* utility fails to process any batch *job\_identifier* successfully,  
 104924 the utility shall proceed to process the remaining batch *job\_identifiers*, if any.

104925 The *qhold* utility shall place holds on each batch job by sending a *Hold Job Request* to the batch  
 104926 server that manages the batch job.

104927 The *qhold* utility shall not exit until holds have been placed on the batch job corresponding to  
 104928 each successfully processed batch *job\_identifier*.

104929 **OPTIONS**

104930 The *qhold* utility shall conform to XBD [Section 12.2](#) (on page 216).

104931 The following option shall be supported by the implementation:

104932 **-h *hold\_list*** Define the types of holds to be placed on the batch job.

104933 The *qhold* **-h** option shall accept a value for the *hold\_list* option-argument that is a  
 104934 string of alphanumeric characters in the portable character set (see XBD [Section](#)  
 104935 [6.1](#), on page 125).

104936 The *qhold* utility shall accept a value for the *hold\_list* option-argument that is a  
 104937 string of one or more of the characters 'u', 's', or 'o', or the single character  
 104938 'n'.

104939 For each unique character in the *hold\_list* option-argument, the *qhold* utility shall  
 104940 add a value to the *Hold\_Types* attribute of the batch job as follows, each  
 104941 representing a different hold type:

104942 u USER

104943 s SYSTEM

104944 o OPERATOR

104945 If any of these characters are duplicated in the *hold\_list* option-argument, the  
 104946 duplicates shall be ignored.

104947 An existing *Hold\_Types* attribute can be cleared by the following hold type:

104948 n NO\_HOLD

104949 The *qhold* utility shall consider it an error if any hold type other than 'n' is  
 104950 combined with hold type 'n'.

104951 Strictly conforming applications shall not repeat any of the characters 'u', 's',

104952 'o', or 'n' within the *hold\_list* option-argument. The *qhold* utility shall permit the  
 104953 repetition of characters, but shall not assign additional meaning to the repeated  
 104954 characters.

104955 An implementation may define other hold types. The conformance document for  
 104956 an implementation shall describe any additional hold types, how they are  
 104957 specified, their internal behavior, and how they affect the behavior of the utility.

104958 If the **-h** option is not presented to the *qhold* utility, the implementation shall set  
 104959 the *Hold\_Types* attribute to USER.

#### 104960 OPERANDS

104961 The *qhold* utility shall accept one or more operands that conform to the syntax for a batch  
 104962 *job\_identifier* (see [Section 3.3.1](#), on page 2439).

#### 104963 STDIN

104964 Not used.

#### 104965 INPUT FILES

104966 None.

#### 104967 ENVIRONMENT VARIABLES

104968 The following environment variables shall affect the execution of *qhold*:

104969 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 104970 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 104971 used to determine the values of locale categories.)

104972 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 104973 internationalization variables.

104974 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 104975 characters (for example, single-byte as opposed to multi-byte characters in  
 104976 arguments).

104977 *LC\_MESSAGES*  
 104978 Determine the locale that should be used to affect the format and contents of  
 104979 diagnostic messages written to standard error.

104980 *LOGNAME* Determine the login name of the user.

#### 104981 ASYNCHRONOUS EVENTS

104982 Default.

#### 104983 STDOUT

104984 None.

#### 104985 STDERR

104986 The standard error shall be used only for diagnostic messages.

#### 104987 OUTPUT FILES

104988 None.

#### 104989 EXTENDED DESCRIPTION

104990 None.

#### 104991 EXIT STATUS

104992 The following exit values shall be returned:

104993 0 Successful completion.

104994 >0 An error occurred.

#### 104995 CONSEQUENCES OF ERRORS

104996 In addition to the default behavior, the *qhold* utility shall not be required to write a diagnostic  
104997 message to standard error when the error reply received from a batch server indicates that the  
104998 batch *job\_identifier* does not exist on the server. Whether or not the *qhold* utility waits to output  
104999 the diagnostic message while attempting to locate the job on other servers is implementation-  
105000 defined.

#### 105001 APPLICATION USAGE

105002 None.

#### 105003 EXAMPLES

105004 None.

#### 105005 RATIONALE

105006 The *qhold* utility allows users to place a hold on one or more jobs. A hold makes a batch job  
105007 ineligible for execution.

105008 The *qhold* utility has options that allow the user to specify the type of hold. Should the user wish  
105009 to place a hold on a set of jobs that meet a selection criteria, such a list of jobs can be acquired  
105010 using the *qselect* utility.

105011 The *-h* option allows the user to specify the type of hold that is to be placed on the job. This  
105012 option allows for USER, SYSTEM, OPERATOR, and implementation-defined hold types. The  
105013 USER and OPERATOR holds are distinct. The batch server that manages the batch job will verify  
105014 that the user is authorized to set the specified hold for the batch job.

105015 Mail is not required on hold because the administrator has the tools and libraries to build this  
105016 option if he or she wishes.

105017 Historically, the *qhold* utility has been a part of some existing batch systems, although it has not  
105018 traditionally been a part of the NQS.

#### 105019 FUTURE DIRECTIONS

105020 The *qhold* utility may be removed in a future version.

#### 105021 SEE ALSO

105022 [Chapter 3](#) (on page 2417), *qselect*

105023 [XBD Section 6.1](#) (on page 125), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

#### 105024 CHANGE HISTORY

105025 Derived from IEEE Std 1003.2d-1994.

#### 105026 Issue 6

105027 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

#### 105028 Issue 7

105029 The *qhold* utility is marked obsolescent.

105030 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

105031 **NAME**

105032 qmove — move batch jobs

105033 **SYNOPSIS**105034 OB BE `qmove destination job_identifier...`105035 **DESCRIPTION**

105036 To move a batch job is to remove the batch job from the batch queue in which it resides and  
 105037 instantiate the batch job in another batch queue. A batch job is moved by a request to the batch  
 105038 server that manages the batch job. The *qmove* utility is a user-accessible batch client that requests  
 105039 the movement of one or more batch jobs.

105040 The *qmove* utility shall move those batch jobs, and only those batch jobs, for which a batch  
 105041 *job\_identifier* is presented to the utility.

105042 The *qmove* utility shall move batch jobs in the order in which the corresponding batch  
 105043 *job\_identifiers* are presented to the utility.

105044 If the *qmove* utility fails to process a batch *job\_identifier* successfully, the utility shall proceed to  
 105045 process the remaining batch *job\_identifiers*, if any.

105046 The *qmove* utility shall move batch jobs by sending a *Move Job Request* to the batch server that  
 105047 manages each batch job. The *qmove* utility shall not exit before the batch jobs corresponding to all  
 105048 successfully processed batch *job\_identifiers* have been moved.

105049 **OPTIONS**

105050 None.

105051 **OPERANDS**

105052 The *qmove* utility shall accept one operand that conforms to the syntax for a destination (see  
 105053 [Section 3.3.2](#), on page 2440).

105054 The *qmove* utility shall accept one or more operands that conform to the syntax for a batch  
 105055 *job\_identifier* (see [Section 3.3.1](#), on page 2439).

105056 **STDIN**

105057 Not used.

105058 **INPUT FILES**

105059 None.

105060 **ENVIRONMENT VARIABLES**105061 The following environment variables shall affect the execution of *qmove*:

105062 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 105063 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 105064 used to determine the values of locale categories.)

105065 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 105066 internationalization variables.

105067 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 105068 characters (for example, single-byte as opposed to multi-byte characters in  
 105069 arguments).

105070 *LC\_MESSAGES*

105071 Determine the locale that should be used to affect the format and contents of  
 105072 diagnostic messages written to standard error.

105073 *LOGNAME* Determine the login name of the user.

105074 **ASYNCHRONOUS EVENTS**

105075 Default.

105076 **STDOUT**

105077 None.

105078 **STDERR**

105079 The standard error shall be used only for diagnostic messages.

105080 **OUTPUT FILES**

105081 None.

105082 **EXTENDED DESCRIPTION**

105083 None.

105084 **EXIT STATUS**

105085 The following exit values shall be returned:

105086 0 Successful completion.

105087 >0 An error occurred.

105088 **CONSEQUENCES OF ERRORS**

105089 In addition to the default behavior, the *qmove* utility shall not be required to write a diagnostic  
105090 message to standard error when the error reply received from a batch server indicates that the  
105091 batch *job\_identifier* does not exist on the server. Whether or not the *qmove* utility waits to output  
105092 the diagnostic message while attempting to locate the job on other servers is implementation-  
105093 defined.

105094 **APPLICATION USAGE**

105095 None.

105096 **EXAMPLES**

105097 None.

105098 **RATIONALE**

105099 The *qmove* utility allows users to move jobs between queues.

105100 The alternative to using the *qmove* utility—deleting the batch job and requeuing it—entails  
105101 considerably more typing.

105102 Since the means of selecting jobs based on attributes has been encapsulated in the *qselect* utility,  
105103 the only option of the *qmove* utility concerns authorization. The **-u** option provides the user with  
105104 the convenience of changing the user identifier under which the batch job will execute.  
105105 Minimalism and consistency have taken precedence over convenience; the **-u** option has been  
105106 deleted because the equivalent capability exists with the **-u** option of the *qalter* utility.

105107 **FUTURE DIRECTIONS**

105108 The *qmove* utility may be removed in a future version.

105109 **SEE ALSO**

105110 Chapter 3 (on page 2417), *qalter*, *qselect*

105111 XBD Chapter 8 (on page 173)

105112 **CHANGE HISTORY**

105113 Derived from IEEE Std 1003.2d-1994.

105114 **Issue 6**

105115 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

105116 **Issue 7**

105117 The *qmove* utility is marked obsolescent.

105118 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



105119 **NAME**

105120 qmsg — send message to batch jobs

105121 **SYNOPSIS**105122 OB BE `qmsg [-EO] message_string job_identifier...`105123 **DESCRIPTION**

105124 To send a message to a batch job is to request that a server write a message string into one or  
 105125 more output files of the batch job. A message is sent to a batch job by a request to the batch  
 105126 server that manages the batch job. The *qmsg* utility is a user-accessible batch client that requests  
 105127 the sending of messages to one or more batch jobs.

105128 The *qmsg* utility shall write messages into the files of batch jobs by sending a *Job Message Request*  
 105129 to the batch server that manages the batch job. The *qmsg* utility shall not directly write the  
 105130 message into the files of the batch job.

105131 The *qmsg* utility shall send a *Job Message Request* for those batch jobs, and only those batch jobs,  
 105132 for which a batch *job\_identifier* is presented to the utility.

105133 The *qmsg* utility shall send *Job Message Requests* for batch jobs in the order in which their batch  
 105134 *job\_identifiers* are presented to the utility.

105135 If the *qmsg* utility fails to process any batch *job\_identifier* successfully, the utility shall proceed to  
 105136 process the remaining batch *job\_identifiers*, if any.

105137 The *qmsg* utility shall not exit before a *Job Message Request* has been sent to the server that  
 105138 manages the batch job that corresponds to each successfully processed batch *job\_identifier*.

105139 **OPTIONS**

105140 The *qmsg* utility shall conform to XBD [Section 12.2](#) (on page 216).

105141 The following options shall be supported by the implementation:

105142 **-E** Specify that the message is written to the standard error of each batch job.

105143 The *qmsg* utility shall write the message into the standard error of the batch job.

105144 **-O** Specify that the message is written to the standard output of each batch job.

105145 The *qmsg* utility shall write the message into the standard output of the batch job.

105146 If neither the **-O** nor the **-E** option is presented to the *qmsg* utility, the utility shall write the  
 105147 message into an implementation-defined file. The conformance document for the  
 105148 implementation shall describe the name and location of the implementation-defined file. If both  
 105149 the **-O** and the **-E** options are presented to the *qmsg* utility, then the utility shall write the  
 105150 messages to both standard output and standard error.

105151 **OPERANDS**

105152 The *qmsg* utility shall accept a minimum of two operands, *message\_string* and one or more batch  
 105153 *job\_identifiers*.

105154 The *message\_string* operand shall be the string to be written to one or more output files of the  
 105155 batch job followed by a <newline>. If the string contains <blank> characters, then the  
 105156 application shall ensure that the string is quoted. The *message\_string* shall be encoded in the  
 105157 portable character set (see XBD [Section 6.1](#), on page 125).

105158 All remaining operands are batch *job\_identifiers* that conform to the syntax for a batch  
 105159 *job\_identifier* (see [Section 3.3.1](#), on page 2439).

105160 **STDIN**

105161 Not used.

105162 **INPUT FILES**

105163 None.

105164 **ENVIRONMENT VARIABLES**105165 The following environment variables shall affect the execution of *qmsg*:

105166 *LANG* Provide a default value for the internationalization variables that are unset or null.  
105167 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
105168 used to determine the values of locale categories.)

105169 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
105170 internationalization variables.

105171 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
105172 characters (for example, single-byte as opposed to multi-byte characters in  
105173 arguments).

105174 *LC\_MESSAGES*

105175 Determine the locale that should be used to affect the format and contents of  
105176 diagnostic messages written to standard error.

105177 *LOGNAME* Determine the login name of the user.105178 **ASYNCHRONOUS EVENTS**

105179 Default.

105180 **STDOUT**

105181 None.

105182 **STDERR**

105183 The standard error shall be used only for diagnostic messages.

105184 **OUTPUT FILES**

105185 None.

105186 **EXTENDED DESCRIPTION**

105187 None.

105188 **EXIT STATUS**

105189 The following exit values shall be returned:

105190 0 Successful completion.

105191 &gt;0 An error occurred.

105192 **CONSEQUENCES OF ERRORS**

105193 In addition to the default behavior, the *qmsg* utility shall not be required to write a diagnostic  
105194 message to standard error when the error reply received from a batch server indicates that the  
105195 batch *job\_identifier* does not exist on the server. Whether or not the *qmsg* utility waits to output  
105196 the diagnostic message while attempting to locate the job on other servers is implementation-  
105197 defined.

**105198 APPLICATION USAGE**

105199 None.

**105200 EXAMPLES**

105201 None.

**105202 RATIONALE**

105203 The *qmsg* utility allows users to write messages into the output files of running jobs. Users,  
105204 including operators and administrators, have a number of occasions when they want to place  
105205 messages in the output files of a batch job. For example, if a disk that is being used by a batch job  
105206 is showing errors, the operator might note this in the standard error stream of the batch job.

105207 The options of the *qmsg* utility provide users with the means of placing the message in the  
105208 output stream of their choice. The default output stream for the message—if the user does not  
105209 designate an output stream—is implementation-defined, since many implementations will  
105210 provide, as an extension to this volume of POSIX.1-2008, a log file that shows the history of  
105211 utility execution.

105212 If users wish to send a message to a set of jobs that meet a selection criteria, the *qselect* utility can  
105213 be used to acquire the appropriate list of job identifiers.

105214 The **-E** option allows users to place the message in the standard error stream of the batch job.

105215 The **-O** option allows users to place the message in the standard output stream of the batch job.

105216 Historically, the *qmsg* utility is an existing practice in the offerings of one or more implementors  
105217 of an NQS-derived batch system. The utility has been found to be useful enough that it deserves  
105218 to be included in this volume of POSIX.1-2008.

**105219 FUTURE DIRECTIONS**

105220 The *qmsg* utility may be removed in a future version.

**105221 SEE ALSO**

105222 [Chapter 3](#) (on page 2417), *qselect*

105223 XBD [Section 6.1](#) (on page 125), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

**105224 CHANGE HISTORY**

105225 Derived from IEEE Std 1003.2d-1994.

**105226 Issue 6**

105227 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

**105228 Issue 7**

105229 The *qmsg* utility is marked obsolescent.

105230 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

105231 **NAME**

105232 qrerun — rerun batch jobs

105233 **SYNOPSIS**105234 OB BE `qrerun job_identifier...`105235 **DESCRIPTION**

105236 To rerun a batch job is to terminate the session leader of the batch job, delete any associated  
 105237 checkpoint files, and return the batch job to the batch queued state. A batch job is rerun by a  
 105238 request to the batch server that manages the batch job. The *qrerun* utility is a user-accessible  
 105239 batch client that requests the rerunning of one or more batch jobs.

105240 The *qrerun* utility shall rerun those batch jobs for which a batch *job\_identifier* is presented to the  
 105241 utility.

105242 The *qrerun* utility shall rerun batch jobs in the order in which their batch *job\_identifiers* are  
 105243 presented to the utility.

105244 If the *qrerun* utility fails to process any batch *job\_identifier* successfully, the utility shall proceed to  
 105245 process the remaining batch *job\_identifiers*, if any.

105246 The *qrerun* utility shall rerun batch jobs by sending a *Rerun Job Request* to the batch server that  
 105247 manages each batch job.

105248 For each successfully processed batch *job\_identifier*, the *qrerun* utility shall have rerun the  
 105249 corresponding batch job at the time the utility exits.

105250 **OPTIONS**

105251 None.

105252 **OPERANDS**

105253 The *qrerun* utility shall accept one or more operands that conform to the syntax for a batch  
 105254 *job\_identifier* (see [Section 3.3.1](#), on page 2439).

105255 **STDIN**

105256 Not used.

105257 **INPUT FILES**

105258 None.

105259 **ENVIRONMENT VARIABLES**105260 The following environment variables shall affect the execution of *qrerun*:

105261 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 105262 (See [XBD Section 8.2](#) (on page 174) the precedence of internationalization variables  
 105263 used to determine the values of locale categories.)

105264 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 105265 internationalization variables.

105266 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 105267 characters (for example, single-byte as opposed to multi-byte characters in  
 105268 arguments).

105269 *LC\_MESSAGES*

105270 Determine the locale that should be used to affect the format and contents of  
 105271 diagnostic messages written to standard error.

105272 **LOGNAME** Determine the login name of the user.

105273 **ASYNCHRONOUS EVENTS**

105274 Default.

105275 **STDOUT**

105276 None.

105277 **STDERR**

105278 The standard error shall be used only for diagnostic messages.

105279 **OUTPUT FILES**

105280 None.

105281 **EXTENDED DESCRIPTION**

105282 None.

105283 **EXIT STATUS**

105284 The following exit values shall be returned:

105285 0 Successful completion.

105286 >0 An error occurred.

105287 **CONSEQUENCES OF ERRORS**

105288 In addition to the default behavior, the *qrerun* utility shall not be required to write a diagnostic  
105289 message to standard error when the error reply received from a batch server indicates that the  
105290 batch *job\_identifier* does not exist on the server. Whether or not the *qrerun* utility waits to output  
105291 the diagnostic message while attempting to locate the job on other servers is implementation-  
105292 defined.

105293 **APPLICATION USAGE**

105294 None.

105295 **EXAMPLES**

105296 None.

105297 **RATIONALE**

105298 The *qrerun* utility allows users to cause jobs in the running state to exit and rerun.

105299 The *qrerun* utility is a new utility, *vis-a-vis* existing practice, that has been defined in this volume  
105300 of POSIX.1-2008 to correct user-perceived deficiencies in the existing practice.

105301 **FUTURE DIRECTIONS**

105302 The *qrerun* utility may be removed in a future version.

105303 **SEE ALSO**

105304 [Chapter 3](#) (on page 2417)

105305 XBD [Chapter 8](#) (on page 173)

105306 **CHANGE HISTORY**

105307 Derived from IEEE Std 1003.2d-1994.

105308 **Issue 6**

105309 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

105310 **Issue 7**

105311 The *qrerun* utility is marked obsolescent.

105312 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

105313 **NAME**

105314 qrln — release batch jobs

105315 **SYNOPSIS**105316 OB BE `qrln [-h hold_list] job_identifier...`105317 **DESCRIPTION**

105318 A batch job might have one or more holds, which prevent the batch job from executing. A batch  
 105319 job from which all the holds have been removed becomes eligible for execution and is said to  
 105320 have been released. A batch job hold is removed by sending a request to the batch server that  
 105321 manages the batch job. The *qrln* utility is a user-accessible client of batch services that requests  
 105322 holds be removed from one or more batch jobs.

105323 The *qrln* utility shall remove one or more holds from those batch jobs for which a batch  
 105324 *job\_identifier* is presented to the utility.

105325 The *qrln* utility shall remove holds from batch jobs in the order in which their batch *job\_identifiers*  
 105326 are presented to the utility.

105327 If the *qrln* utility fails to process a batch *job\_identifier* successfully, the utility shall proceed to  
 105328 process the remaining batch *job\_identifiers*, if any.

105329 The *qrln* utility shall remove holds on each batch job by sending a *Release Job Request* to the batch  
 105330 server that manages the batch job.

105331 The *qrln* utility shall not exit until the holds have been removed from the batch job  
 105332 corresponding to each successfully processed batch *job\_identifier*.

105333 **OPTIONS**

105334 The *qrln* utility shall conform to XBD [Section 12.2](#) (on page 216).

105335 The following option shall be supported by the implementation:

105336 **-h *hold\_list*** Define the types of holds to be removed from the batch job.

105337 The *qrln* **-h** option shall accept a value for the *hold\_list* option-argument that is a  
 105338 string of alphanumeric characters in the portable character set (see XBD [Section](#)  
 105339 [6.1](#), on page 125).

105340 The *qrln* utility shall accept a value for the *hold\_list* option-argument that is a string  
 105341 of one or more of the characters 'u', 's', or 'o', or the single character 'n'.

105342 For each unique character in the *hold\_list* option-argument, the *qrln* utility shall add  
 105343 a value to the *Hold\_Types* attribute of the batch job as follows, each representing a  
 105344 different hold type:

105345 u USER

105346 s SYSTEM

105347 o OPERATOR

105348 If any of these characters are duplicated in the *hold\_list* option-argument, the  
 105349 duplicates shall be ignored.

105350 An existing *Hold\_Types* attribute can be cleared by the following hold type:

105351 n NO\_HOLD

105352 The *qrln* utility shall consider it an error if any hold type other than 'n' is  
 105353 combined with hold type 'n'.

105354 Strictly conforming applications shall not repeat any of the characters 'u', 's',  
 105355 'o', or 'n' within the *hold\_list* option-argument. The *qrIs* utility shall permit the  
 105356 repetition of characters, but shall not assign additional meaning to the repeated  
 105357 characters.

105358 An implementation may define other hold types. The conformance document for  
 105359 an implementation shall describe any additional hold types, how they are  
 105360 specified, their internal behavior, and how they affect the behavior of the utility.

105361 If the **-h** option is not presented to the *qrIs* utility, the implementation shall remove  
 105362 the USER hold in the *Hold\_Types* attribute.

#### 105363 OPERANDS

105364 The *qrIs* utility shall accept one or more operands that conform to the syntax for a batch  
 105365 *job\_identifier* (see [Section 3.3.1](#), on page 2439).

#### 105366 STDIN

105367 Not used.

#### 105368 INPUT FILES

105369 None.

#### 105370 ENVIRONMENT VARIABLES

105371 The following environment variables shall affect the execution of *qrIs*:

105372 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 105373 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 105374 used to determine the values of locale categories.)

105375 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 105376 internationalization variables.

105377 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 105378 characters (for example, single-byte as opposed to multi-byte characters in  
 105379 arguments).

105380 *LC\_MESSAGES*

105381 Determine the locale that should be used to affect the format and contents of  
 105382 diagnostic messages written to standard error.

105383 *LOGNAME* Determine the login name of the user.

#### 105384 ASYNCHRONOUS EVENTS

105385 Default.

#### 105386 STDOUT

105387 None.

#### 105388 STDERR

105389 The standard error shall be used only for diagnostic messages.

#### 105390 OUTPUT FILES

105391 None.

#### 105392 EXTENDED DESCRIPTION

105393 None.



**105394 EXIT STATUS**

105395 The following exit values shall be returned:

105396 0 Successful completion.

105397 >0 An error occurred.

**105398 CONSEQUENCES OF ERRORS**

105399 In addition to the default behavior, the *qrsl* utility shall not be required to write a diagnostic  
105400 message to standard error when the error reply received from a batch server indicates that the  
105401 batch *job\_identifier* does not exist on the server. Whether or not the *qrsl* utility waits to output the  
105402 diagnostic message while attempting to locate the job on other servers is implementation-  
105403 defined.

**105404 APPLICATION USAGE**

105405 None.

**105406 EXAMPLES**

105407 None.

**105408 RATIONALE**

105409 The *qrsl* utility allows users, operators, and administrators to remove holds from jobs.

105410 The *qrsl* utility does not support any job selection options or wildcard arguments. Users may  
105411 acquire a list of jobs selected by attributes using the *qselect* utility. For example, a user could  
105412 select all of their held jobs.

105413 The **-h** option allows the user to specify the type of hold that is to be removed. This option  
105414 allows for USER, SYSTEM, OPERATOR, and implementation-defined hold types. The batch  
105415 server that manages the batch job will verify whether the user is authorized to remove the  
105416 specified hold for the batch job. If more than one type of hold has been placed on the batch job, a  
105417 user may wish to remove only some of them.

105418 Mail is not required on release because the administrator has the tools and libraries to build this  
105419 option if required.

105420 The *qrsl* utility is a new utility *vis-a-vis* existing practice; it has been defined in this volume of  
105421 POSIX.1-2008 as the natural complement to the *qhold* utility.

**105422 FUTURE DIRECTIONS**

105423 The *qrsl* utility may be removed in a future version.

**105424 SEE ALSO**

105425 [Chapter 3](#) (on page 2417), *qhold*, *qselect*

105426 [XBD Section 6.1](#) (on page 125), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

**105427 CHANGE HISTORY**

105428 Derived from IEEE Std 1003.2d-1994.

**105429 Issue 6**

105430 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

**105431 Issue 7**

105432 The *qrsl* utility is marked obsolescent.

105433 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

105434 **NAME**

105435 qselect — select batch jobs

105436 **SYNOPSIS**

```
105437 OB BE qselect [-a [op]date_time] [-A account_string] [-c [op]interval]
105438           [-h hold_list] [-l resource_list] [-N name] [-p [op]priority]
105439           [-q destination] [-r y|n] [-s states] [-u user_list]
```

105440 **DESCRIPTION**

105441 To select a set of batch jobs is to return the batch *job\_identifiers* for each batch job that meets a list  
 105442 of selection criteria. A set of batch jobs is selected by a request to a batch server. The *qselect* utility  
 105443 is a user-accessible batch client that requests the selection of batch jobs.

105444 Upon successful completion, the *qselect* utility shall have returned a list of zero or more batch  
 105445 *job\_identifiers* that meet the criteria specified by the options and option-arguments presented to  
 105446 the utility.

105447 The *qselect* utility shall select batch jobs by sending a *Select Jobs Request* to a batch server. The  
 105448 *qselect* utility shall not exit until the server replies to each request generated.

105449 For each option presented to the *qselect* utility, the utility shall restrict the set of selected batch  
 105450 jobs as described in the OPTIONS section.

105451 The *qselect* utility shall not restrict selection of batch jobs except by authorization and as required  
 105452 by the options presented to the utility.

105453 When an option is specified with a mandatory or optional *op* component to the option-  
 105454 argument, then *op* shall specify a relation between the value of a certain batch job attribute and  
 105455 the *value* component of the option-argument. If an *op* is allowable on an option, then the  
 105456 description of the option letter indicates the *op* as either mandatory or optional. Acceptable  
 105457 strings for the *op* component, and the relation the string indicates, are shown in the following  
 105458 list:

105459 .eq. The value represented by the attribute of the batch job is equal to the value represented  
 105460 by the option-argument.

105461 .ge. The value represented by the attribute of the batch job is greater than or equal to the  
 105462 value represented by the option-argument.

105463 .gt. The value represented by the attribute of the batch job is greater than the value  
 105464 represented by the option-argument.

105465 .lt. The value represented by the attribute of the batch job is less than the value represented  
 105466 by the option-argument.

105467 .le. The value represented by the attribute of the batch job is less than or equal to the value  
 105468 represented by the option-argument.

105469 .ne. The value represented by the attribute of the batch job is not equal to the value  
 105470 represented by the option-argument.

105471 **OPTIONS**

105472 The *qselect* utility shall conform to XBD [Section 12.2](#) (on page 216).

105473 The following options shall be supported by the implementation:

105474 **-a [op]date\_time**

105475 Restrict selection to a specific time, or a range of times.

105476 The *qselect* utility shall select only batch jobs for which the value of the

105477 *Execution\_Time* attribute is related to the Epoch equivalent of the local time  
 105478 expressed by the value of the *date\_time* component of the option-argument in the  
 105479 manner indicated by the value of the *op* component of the option-argument.

105480 The *qselect* utility shall accept a *date\_time* component of the option-argument that  
 105481 conforms to the syntax of the *time* operand of the *touch* utility.

105482 If the *op* component of the option-argument is not presented to the *qselect* utility,  
 105483 the utility shall select batch jobs for which the *Execution\_Time* attribute is equal to  
 105484 the *date\_time* component of the option-argument.

105485 When comparing times, the *qselect* utility shall use the following definitions for the  
 105486 *op* component of the option-argument:

105487 .eq. The time represented by value of the *Execution\_Time* attribute of the batch  
 105488 job is equal to the time represented by the *date\_time* component of the  
 105489 option-argument.

105490 .ge. The time represented by value of the *Execution\_Time* attribute of the batch  
 105491 job is after or equal to the time represented by the *date\_time* component of  
 105492 the option-argument.

105493 .gt. The time represented by value of the *Execution\_Time* attribute of the batch  
 105494 job is after the time represented by the *date\_time* component of the option-  
 105495 argument.

105496 .lt. The time represented by value of the *Execution\_Time* attribute of the batch  
 105497 job is before the time represented by the *date\_time* component of the  
 105498 option-argument.

105499 .le. The time represented by value of the *Execution\_Time* attribute of the batch  
 105500 job is before or equal to the time represented by the *date\_time* component  
 105501 of the option-argument.

105502 .ne. The time represented by value of the *Execution\_Time* attribute of the batch  
 105503 job is not equal to the time represented by the *date\_time* component of the  
 105504 option-argument.

105505 The *qselect* utility shall accept the defined character strings for the *op* component of  
 105506 the option-argument.

105507 **-A** *account\_string*

105508 Restrict selection to the batch jobs charging a specified account.

105509 The *qselect* utility shall select only batch jobs for which the value of the  
 105510 *Account\_Name* attribute of the batch job matches the value of the *account\_string*  
 105511 option-argument.

105512 The syntax of the *account\_string* option-argument is unspecified.

105513 **-c** [*op*]*interval*

105514 Restrict selection to batch jobs within a range of checkpoint intervals.

105515 The *qselect* utility shall select only batch jobs for which the value of the *Checkpoint*  
 105516 attribute relates to the value of the *interval* component of the option-argument in  
 105517 the manner indicated by the value of the *op* component of the option-argument.

105518 If the *op* component of the option-argument is omitted, the *qselect* utility shall select  
 105519 batch jobs for which the value of the *Checkpoint* attribute is equal to the value of the  
 105520 *interval* component of the option-argument.

105521 When comparing checkpoint intervals, the *qselect* utility shall use the following  
 105522 definitions for the *op* component of the option-argument:

105523 .eq. The value of the *Checkpoint* attribute of the batch job equals the value of  
 105524 the *interval* component of the option-argument.

105525 .ge. The value of the *Checkpoint* attribute of the batch job is greater than or  
 105526 equal to the value of the *interval* component option-argument.

105527 .gt. The value of the *Checkpoint* attribute of the batch job is greater than the  
 105528 value of the *interval* component option-argument.

105529 .lt. The value of the *Checkpoint* attribute of the batch job is less than the value  
 105530 of the *interval* component option-argument.

105531 .le. The value of the *Checkpoint* attribute of the batch job is less than or equal  
 105532 to the value of the *interval* component option-argument.

105533 .ne. The value of the *Checkpoint* attribute of the batch job does not equal the  
 105534 value of the *interval* component option-argument.

105535 The *qselect* utility shall accept the defined character strings for the *op* component of  
 105536 the option-argument.

105537 The ordering relationship for the values of the interval option-argument is defined  
 105538 to be:

105539 'n' .gt. 's' .gt. 'c=minutes' .ge. 'c'

105540 When comparing *Checkpoint* attributes with an interval having the value of the  
 105541 single character 'u', only equality or inequality are valid comparisons.

105542 **-h hold\_list** Restrict selection to batch jobs that have a specific type of hold.

105543 The *qselect* utility shall select only batch jobs for which the value of the *Hold\_Types*  
 105544 attribute matches the value of the *hold\_list* option-argument.

105545 The *qselect* **-h** option shall accept a value for the *hold\_list* option-argument that is a  
 105546 string of alphanumeric characters in the portable character set (see XBD [Section 6.1](#), on page 125).  
 105547

105548 The *qselect* utility shall accept a value for the *hold\_list* option-argument that is a  
 105549 string of one or more of the characters 'u', 's', or 'o', or the single character  
 105550 'n'.

105551 Each unique character in the *hold\_list* option-argument of the *qselect* utility is  
 105552 defined as follows, each representing a different hold type:

105553 u USER  
 105554 s SYSTEM  
 105555 o OPERATOR

105556 If any of these characters are duplicated in the *hold\_list* option-argument, the  
 105557 duplicates shall be ignored.

105558 The *qselect* utility shall consider it an error if any hold type other than 'n' is  
 105559 combined with hold type 'n'.

105560 Strictly conforming applications shall not repeat any of the characters 'u', 's',  
 105561 'o', or 'n' within the *hold\_list* option-argument. The *qselect* utility shall permit

105562 the repetition of characters, but shall not assign additional meaning to the repeated  
105563 characters.

105564 An implementation may define other hold types. The conformance document for  
105565 an implementation shall describe any additional hold types, how they are  
105566 specified, their internal behavior, and how they affect the behavior of the utility.

105567 **-I resource\_list**

105568 Restrict selection to batch jobs with specified resource limits and attributes.

105569 The *qselect* utility shall accept a *resource\_list* option-argument with the following  
105570 syntax:

105571 *resource\_name op value [, , resource\_name op value, , ...]*

105572 When comparing resource values, the *qselect* utility shall use the following  
105573 definitions for the *op* component of the option-argument:

105574 .eq. The value of the resource of the same name in the *Resource\_List* attribute  
105575 of the batch job equals the value of the value component of the option-  
105576 argument.

105577 .ge. The value of the resource of the same name in the *Resource\_List* attribute  
105578 of the batch job is greater than or equal to the value of the *value*  
105579 component of the option-argument.

105580 .gt. The value of the resource of the same name in the *Resource\_List* attribute  
105581 of the batch job is greater than the value of the value component of the  
105582 option-argument.

105583 .lt. The value of the resource of the same name in the *Resource\_List* attribute  
105584 of the batch job is less than the value of the value component of the  
105585 option-argument.

105586 .ne. The value of the resource of the same name in the *Resource\_List* attribute  
105587 of the batch job does not equal the value of the value component of the  
105588 option-argument.

105589 .le. The value of the resource of the same name in the *Resource\_List* attribute  
105590 of the batch job is less than or equal to the value of the *value* component of  
105591 the option-argument.

105592 When comparing the limit of a *Resource\_List* attribute with the *value* component of  
105593 the option-argument, if the limit, the value, or both are non-numeric, only equality  
105594 or inequality are valid comparisons.

105595 The *qselect* utility shall select only batch jobs for which the values of the  
105596 *resource\_names* listed in the *resource\_list* option-argument match the corresponding  
105597 limits of the *Resource\_List* attribute of the batch job.

105598 Limits of *resource\_names* present in the *Resource\_List* attribute of the batch job that  
105599 have no corresponding values in the *resource\_list* option-argument shall not be  
105600 considered when selecting batch jobs.

105601 **-N name** Restrict selection to batch jobs with a specified name.

105602 The *qselect* utility shall select only batch jobs for which the value of the *Job\_Name*  
105603 attribute matches the value of the *name* option-argument. The string specified in  
105604 the *name* option-argument shall be passed, uninterpreted, to the server. This allows  
105605 an implementation to match “wildcard” patterns against batch job names.

105606 An implementation shall describe in the conformance document the format it  
105607 supports for matching against the *Job\_Name* attribute.

105608 **-p** [*op*]*priority*

105609 Restrict selection to batch jobs of the specified priority or range of priorities.

105610 The *qselect* utility shall select only batch jobs for which the value of the *Priority*  
105611 attribute of the batch job relates to the value of the *priority* component of the  
105612 option-argument in the manner indicated by the value of the *op* component of the  
105613 option-argument.

105614 If the *op* component of the option-argument is omitted, the *qselect* utility shall select  
105615 batch jobs for which the value of the *Priority* attribute of the batch job is equal to  
105616 the value of the *priority* component of the option-argument.

105617 When comparing priority values, the *qselect* utility shall use the following  
105618 definitions for the *op* component of the option-argument:

105619 .eq. The value of the *Priority* attribute of the batch job equals the value of the  
105620 *priority* component of the option-argument.

105621 .ge. The value of the *Priority* attribute of the batch job is greater than or equal  
105622 to the value of the *priority* component option-argument.

105623 .gt. The value of the *Priority* attribute of the batch job is greater than the value  
105624 of the *priority* component option-argument.

105625 .lt. The value of the *Priority* attribute of the batch job is less than the value of  
105626 the *priority* component option-argument.

105627 .lte. The value of the *Priority* attribute of the batch job is less than or equal to  
105628 the value of the *priority* component option-argument.

105629 .ne. The value of the *Priority* attribute of the batch job does not equal the value  
105630 of the *priority* component option-argument.

105631 **-q** *destination*

105632 Restrict selection to the specified batch queue or server, or both.

105633 The *qselect* utility shall select only batch jobs that are located at the destination  
105634 indicated by the value of the *destination* option-argument.

105635 The destination defines a batch queue, a server, or a batch queue at a server.

105636 The *qselect* utility shall accept an option-argument for the **-q** option that conforms  
105637 to the syntax for a destination. If the **-q** option is not presented to the *qselect* utility,  
105638 the utility shall select batch jobs from all batch queues at the default batch server.

105639 If the option-argument describes only a batch queue, the *qselect* utility shall select  
105640 only batch jobs from the batch queue of the specified name at the default batch  
105641 server. The means by which *qselect* determines the default server is  
105642 implementation-defined.

105643 If the option-argument describes only a batch server, the *qselect* utility shall select  
105644 batch jobs from all the batch queues at that batch server.

105645 If the option-argument describes both a batch queue and a batch server, the *qselect*  
105646 utility shall select only batch jobs from the specified batch queue at the specified  
105647 server.



|        |                              |                                                                                                                                                                                                                                                      |
|--------|------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 105648 | <b>-r y   n</b>              | Restrict selection to batch jobs with the specified rerunability status.                                                                                                                                                                             |
| 105649 |                              | The <i>qselect</i> utility shall select only batch jobs for which the value of the <i>Rerunable</i> attribute of the batch job matches the value of the option-argument.                                                                             |
| 105650 |                              |                                                                                                                                                                                                                                                      |
| 105651 |                              | The <i>qselect</i> utility shall accept a value for the option-argument that consists of either the single character 'y' or the single character 'n'. The character 'y' represents the value TRUE, and the character 'n' represents the value FALSE. |
| 105652 |                              |                                                                                                                                                                                                                                                      |
| 105653 |                              |                                                                                                                                                                                                                                                      |
| 105654 | <b>-s states</b>             | Restrict selection to batch jobs in the specified states.                                                                                                                                                                                            |
| 105655 |                              | The <i>qselect</i> utility shall accept an option-argument that consists of any combination of the characters 'e', 'q', 'r', 'w', 'h', and 't'.                                                                                                      |
| 105656 |                              |                                                                                                                                                                                                                                                      |
| 105657 |                              | Conforming applications shall not repeat any character in the option-argument.                                                                                                                                                                       |
| 105658 |                              | The <i>qselect</i> utility shall permit the repetition of characters in the option-argument, but shall not assign additional meaning to repeated characters.                                                                                         |
| 105659 |                              |                                                                                                                                                                                                                                                      |
| 105660 |                              | The <i>qselect</i> utility shall interpret the characters in the <i>states</i> option-argument as follows:                                                                                                                                           |
| 105661 |                              |                                                                                                                                                                                                                                                      |
| 105662 | e                            | Represents the EXITING state.                                                                                                                                                                                                                        |
| 105663 | q                            | Represents the QUEUED state.                                                                                                                                                                                                                         |
| 105664 | r                            | Represents the RUNNING state.                                                                                                                                                                                                                        |
| 105665 | t                            | Represents the TRANSITING state.                                                                                                                                                                                                                     |
| 105666 | h                            | Represents the HELD state.                                                                                                                                                                                                                           |
| 105667 | w                            | Represents the WAITING state.                                                                                                                                                                                                                        |
| 105668 |                              | For each character in the <i>states</i> option-argument, the <i>qselect</i> utility shall select batch jobs in the corresponding state.                                                                                                              |
| 105669 |                              |                                                                                                                                                                                                                                                      |
| 105670 | <b>-u user_list</b>          | Restrict selection to batch jobs owned by the specified user names.                                                                                                                                                                                  |
| 105671 |                              | The <i>qselect</i> utility shall select only the batch jobs of those users specified in the <i>user_list</i> option-argument.                                                                                                                        |
| 105672 |                              |                                                                                                                                                                                                                                                      |
| 105673 |                              | The <i>qselect</i> utility shall accept a <i>user_list</i> option-argument that conforms to the following syntax:                                                                                                                                    |
| 105674 |                              |                                                                                                                                                                                                                                                      |
| 105675 |                              | <code>username[@host][, , username[@host], , ...]</code>                                                                                                                                                                                             |
| 105676 |                              | The <i>qselect</i> utility shall accept only one user name that is missing a corresponding host name. The <i>qselect</i> utility shall accept only one user name per named host.                                                                     |
| 105677 |                              |                                                                                                                                                                                                                                                      |
| 105678 | <b>OPERANDS</b>              |                                                                                                                                                                                                                                                      |
| 105679 |                              | None.                                                                                                                                                                                                                                                |
| 105680 | <b>STDIN</b>                 |                                                                                                                                                                                                                                                      |
| 105681 |                              | Not used.                                                                                                                                                                                                                                            |
| 105682 | <b>INPUT FILES</b>           |                                                                                                                                                                                                                                                      |
| 105683 |                              | None.                                                                                                                                                                                                                                                |
| 105684 | <b>ENVIRONMENT VARIABLES</b> |                                                                                                                                                                                                                                                      |
| 105685 |                              | The following environment variables shall affect the execution of <i>qselect</i> :                                                                                                                                                                   |

|        |                               |                                                                                                                                                                                                                                                |
|--------|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 105686 | <i>LANG</i>                   | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) the precedence of internationalization variables used to determine the values of locale categories.) |
| 105687 |                               |                                                                                                                                                                                                                                                |
| 105688 |                               |                                                                                                                                                                                                                                                |
| 105689 | <i>LC_ALL</i>                 | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                       |
| 105690 |                               |                                                                                                                                                                                                                                                |
| 105691 | <i>LC_CTYPE</i>               | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).                                                                      |
| 105692 |                               |                                                                                                                                                                                                                                                |
| 105693 |                               |                                                                                                                                                                                                                                                |
| 105694 | <i>LC_MESSAGES</i>            |                                                                                                                                                                                                                                                |
| 105695 |                               | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                   |
| 105696 |                               |                                                                                                                                                                                                                                                |
| 105697 | <i>LOGNAME</i>                | Determine the login name of the user.                                                                                                                                                                                                          |
| 105698 | <i>TZ</i>                     | Determine the timezone used to interpret the <i>date-time</i> option-argument. If <i>TZ</i> is unset or null, an unspecified default timezone shall be used.                                                                                   |
| 105699 |                               |                                                                                                                                                                                                                                                |
| 105700 | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                                                                                                                |
| 105701 |                               | Default.                                                                                                                                                                                                                                       |
| 105702 | <b>STDOUT</b>                 |                                                                                                                                                                                                                                                |
| 105703 |                               | The <i>qselect</i> utility shall write zero or more batch <i>job_identifiers</i> to standard output.                                                                                                                                           |
| 105704 |                               | The <i>qselect</i> utility shall separate the batch <i>job_identifiers</i> written to standard output by white space.                                                                                                                          |
| 105705 |                               |                                                                                                                                                                                                                                                |
| 105706 |                               | The <i>qselect</i> utility shall write batch <i>job_identifiers</i> in the following format:                                                                                                                                                   |
| 105707 |                               | <i>sequence_number.server_name@server</i>                                                                                                                                                                                                      |
| 105708 | <b>STDERR</b>                 |                                                                                                                                                                                                                                                |
| 105709 |                               | The standard error shall be used only for diagnostic messages.                                                                                                                                                                                 |
| 105710 | <b>OUTPUT FILES</b>           |                                                                                                                                                                                                                                                |
| 105711 |                               | None.                                                                                                                                                                                                                                          |
| 105712 | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                                                                                                                |
| 105713 |                               | None.                                                                                                                                                                                                                                          |
| 105714 | <b>EXIT STATUS</b>            |                                                                                                                                                                                                                                                |
| 105715 |                               | The following exit values shall be returned:                                                                                                                                                                                                   |
| 105716 | 0                             | Successful completion.                                                                                                                                                                                                                         |
| 105717 | >0                            | An error occurred.                                                                                                                                                                                                                             |
| 105718 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                                                                                                                |
| 105719 |                               | Default.                                                                                                                                                                                                                                       |



105720 **APPLICATION USAGE**

105721 None.

105722 **EXAMPLES**

105723 The following example shows how a user might use the *qselect* utility in conjunction with the  
105724 *qdel* utility to delete all of his or her jobs in the queued state without affecting any jobs that are  
105725 already running:

105726 `qdel $(qselect -s q)`

105727 or:

105728 `qselect -s q || xargs qdel`105729 **RATIONALE**

105730 The *qselect* utility allows users to acquire a list of job identifiers that match user-specified  
105731 selection criteria. The list of identifiers returned by the *qselect* utility conforms to the syntax of  
105732 the batch job identifier list processed by a utility such as *qmove*, *qdel*, and *qrls*. The *qselect* utility  
105733 is thus a powerful tool for causing another batch system utility to act upon a set of jobs that  
105734 match a list of selection criteria.

105735 The options of the *qselect* utility let the user apply a number of useful filters for selecting jobs.  
105736 Each option further restricts the selection of jobs. Many of the selection options allow the  
105737 specification of a relational operator. The FORTRAN-like syntax of the operator—that is,  
105738 ".lt."—was chosen rather than the C-like "<=" meta-characters.

105739 The **-a** option allows users to restrict the selected jobs to those that have been submitted (or  
105740 altered) to wait until a particular time. The time period is determined by the argument of this  
105741 option, which includes both a time and an operator—it is thus possible to select jobs waiting  
105742 until a specific time, jobs waiting until after a certain time, or those waiting for a time before the  
105743 specified time.

105744 The **-A** option allows users to restrict the selected jobs to those that have been submitted (or  
105745 altered) to charge a particular account.

105746 The **-c** option allows users to restrict the selected jobs to those whose checkpointing interval  
105747 falls within the specified range.

105748 The **-l** option allows users to select those jobs whose resource limits fall within the range  
105749 indicated by the value of the option. For example, a user could select those jobs for which the  
105750 CPU time limit is greater than two hours.

105751 The **-N** option allows users to select jobs by job name. For instance, all the parts of a task that  
105752 have been divided in parallel jobs might be given the same name, and thus manipulated as a  
105753 group by means of this option.

105754 The **-q** option allows users to select jobs in a specified queue.

105755 The **-r** option allows users to select only those jobs with a specified rerun criteria. For instance, a  
105756 user might select only those jobs that can be rerun for use with the *qrerun* utility.

105757 The **-s** option allows users to select only those jobs that are in a certain state.

105758 The **-u** option allows users to select jobs that have been submitted to execute under a particular  
105759 account.

105760 The selection criteria provided by the options of the *qselect* utility allow users to select jobs based  
105761 on all the appropriate attributes that can be assigned to jobs by the *qsub* utility.

105762 Historically, the *qselect* utility has not been a part of existing practice; it is an improvement that

105763 has been introduced in this volume of POSIX.1-2008.

105764 **FUTURE DIRECTIONS**

105765 The *qselect* utility may be removed in a future version.

105766 **SEE ALSO**

105767 Chapter 3 (on page 2417), *qdel*, *qrerun*, *qrls*, *qselect*, *qsub*, *touch*

105768 XBD Section 6.1 (on page 125), Chapter 8 (on page 173), Section 12.2 (on page 216)

105769 **CHANGE HISTORY**

105770 Derived from IEEE Std 1003.2d-1994.

105771 **Issue 7**

105772 The *qselect* utility is marked obsolescent.

105773 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

105774 **NAME**

105775           qsig — signal batch jobs

105776 **SYNOPSIS**105777 OB BE   qsig [-s *signal*] *job\_identifier*...105778 **DESCRIPTION**

105779       To signal a batch job is to send a signal to the session leader of the batch job. A batch job is  
 105780       signaled by sending a request to the batch server that manages the batch job. The *qsig* utility is a  
 105781       user-accessible batch client that requests the signaling of a batch job.

105782       The *qsig* utility shall signal those batch jobs for which a batch *job\_identifier* is presented to the  
 105783       utility. The *qsig* utility shall not signal any batch jobs whose batch *job\_identifiers* are not  
 105784       presented to the utility.

105785       The *qsig* utility shall signal batch jobs in the order in which the corresponding batch  
 105786       *job\_identifiers* are presented to the utility. If the *qsig* utility fails to process a batch *job\_identifier*  
 105787       successfully, the utility shall proceed to process the remaining batch *job\_identifiers*, if any.

105788       The *qsig* utility shall signal batch jobs by sending a *Signal Job Request* to the batch server that  
 105789       manages the batch job.

105790       For each successfully processed batch *job\_identifier*, the *qsig* utility shall have received a  
 105791       completion reply to each *Signal Job Request* sent to a batch server at the time the utility exits.

105792 **OPTIONS**105793       The *qsig* utility shall conform to XBD [Section 12.2](#) (on page 216).

105794       The following option shall be supported by the implementation:

105795       -s *signal*     Define the signal to be sent to the batch job.

105796               The *qsig* utility shall accept a *signal* option-argument that is either a symbolic signal  
 105797               name or an unsigned integer signal number (see the POSIX.1-1990 standard,  
 105798               Section 3.3.1.1). The *qsig* utility shall accept signal names for which the SIG prefix  
 105799               has been omitted.

105800               If the *signal* option-argument is a signal name, the *qsig* utility shall send that name.

105801               If the *signal* option-argument is a number, the *qsig* utility shall send the signal  
 105802               value represented by the number.

105803               If the -s option is not presented to the *qsig* utility, the utility shall send the signal  
 105804               SIGTERM to each signaled batch job.

105805 **OPERANDS**

105806       The *qsig* utility shall accept one or more operands that conform to the syntax for a batch  
 105807       *job\_identifier* (see [Section 3.3.1](#), on page 2439).

105808 **STDIN**

105809       Not used.

105810 **INPUT FILES**

105811       None.

105812 **ENVIRONMENT VARIABLES**105813       The following environment variables shall affect the execution of *qsig*:

|        |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|--------|-------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 105814 | <i>LANG</i>                   | Provide a default value for the internationalization variables that are unset or null. (See XBD <a href="#">Section 8.2</a> (on page 174) the precedence of internationalization variables used to determine the values of locale categories.)                                                                                                                                                                                       |
| 105815 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105816 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105817 | <i>LC_ALL</i>                 | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                                                                                                                                                                                                                                                                             |
| 105818 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105819 | <i>LC_CTYPE</i>               | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).                                                                                                                                                                                                                                                            |
| 105820 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105821 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105822 | <i>LC_MESSAGES</i>            |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105823 |                               | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                                                                                                                                                                                                         |
| 105824 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105825 | <i>LOGNAME</i>                | Determine the login name of the user.                                                                                                                                                                                                                                                                                                                                                                                                |
| 105826 | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105827 |                               | Default.                                                                                                                                                                                                                                                                                                                                                                                                                             |
| 105828 | <b>STDOUT</b>                 |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105829 |                               | An implementation of the <i>qsig</i> utility may write informative messages to standard output.                                                                                                                                                                                                                                                                                                                                      |
| 105830 | <b>STDERR</b>                 |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105831 |                               | The standard error shall be used only for diagnostic messages.                                                                                                                                                                                                                                                                                                                                                                       |
| 105832 | <b>OUTPUT FILES</b>           |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105833 |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 105834 | <b>EXTENDED DESCRIPTION</b>   |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105835 |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 105836 | <b>EXIT STATUS</b>            |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105837 |                               | The following exit values shall be returned:                                                                                                                                                                                                                                                                                                                                                                                         |
| 105838 | 0                             | Successful completion.                                                                                                                                                                                                                                                                                                                                                                                                               |
| 105839 | >0                            | An error occurred.                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 105840 | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105841 |                               | In addition to the default behavior, the <i>qsig</i> utility shall not be required to write a diagnostic message to standard error when the error reply received from a batch server indicates that the batch <i>job_identifier</i> does not exist on the server. Whether or not the <i>qsig</i> utility waits to output the diagnostic message while attempting to locate the batch job on other servers is implementation-defined. |
| 105842 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105843 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105844 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105845 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105846 | <b>APPLICATION USAGE</b>      |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105847 |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 105848 | <b>EXAMPLES</b>               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105849 |                               | None.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 105850 | <b>RATIONALE</b>              |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105851 |                               | The <i>qsig</i> utility allows users to signal batch jobs.                                                                                                                                                                                                                                                                                                                                                                           |
| 105852 |                               | A user may be unable to signal a batch job with the <i>kill</i> utility of the operating system for a number of reasons. First, the process ID of the batch job may be unknown to the user. Second, the processes of the batch job may be on a remote node. However, by virtue of communication between batch nodes, the <i>qsig</i> utility can arrange for the signaling of a process.                                             |
| 105853 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105854 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105855 |                               |                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| 105856 |                               | Because a batch job that is not running cannot be signaled, and because the signal may not                                                                                                                                                                                                                                                                                                                                           |

- 105857 terminate the batch job, the *qsig* utility is not a substitute for the *qdel* utility.
- 105858 The options of the *qsig* utility allow the user to specify the signal that is to be sent to the batch  
105859 job.
- 105860 The *-s* option allows users to specify a signal by name or by number, and thus override the  
105861 default signal. The POSIX.1-1990 standard defines signals by both name and number.
- 105862 The *qsig* utility is a new utility, *vis-a-vis* existing practice; it has been defined in this volume of  
105863 POSIX.1-2008 in response to user-perceived shortcomings in existing practice.
- 105864 **FUTURE DIRECTIONS**
- 105865 The *qsig* utility may be removed in a future version.
- 105866 **SEE ALSO**
- 105867 [Chapter 3](#) (on page 2417), *kill*, *qdel*
- 105868 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)
- 105869 **CHANGE HISTORY**
- 105870 Derived from IEEE Std 1003.2d-1994.
- 105871 **Issue 6**
- 105872 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.
- 105873 **Issue 7**
- 105874 The *qsig* utility is marked obsolescent.
- 105875 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

105876 **NAME**

105877       qstat — show status of batch jobs

105878 **SYNOPSIS**105879 OB BE     qstat [-f] *job\_identifier...*105880             qstat -Q [-f] *destination...*105881             qstat -B [-f] *server\_name...*105882 **DESCRIPTION**

105883       The status of a batch job, batch queue, or batch server is obtained by a request to the server. The  
 105884       *qstat* utility is a user-accessible batch client that requests the status of one or more batch jobs,  
 105885       batch queues, or servers, and writes the status information to standard output.

105886       For each successfully processed batch *job\_identifier*, the *qstat* utility shall display information  
 105887       about the corresponding batch job.

105888       For each successfully processed destination, the *qstat* utility shall display information about the  
 105889       corresponding batch queue.

105890       For each successfully processed server name, the *qstat* utility shall display information about the  
 105891       corresponding server.

105892       The *qstat* utility shall acquire batch job status information by sending a *Job Status Request* to a  
 105893       batch server. The *qstat* utility shall acquire batch queue status information by sending a *Queue*  
 105894       *Status Request* to a batch server. The *qstat* utility shall acquire server status information by  
 105895       sending a *Server Status Request* to a batch server.

105896 **OPTIONS**105897       The *qstat* utility shall conform to XBD [Section 12.2](#) (on page 216).

105898       The following options shall be supported by the implementation:

105899       **-f**           Specify that a full display is produced.

105900                    The minimum contents of a full display are specified in the STDOUT section.

105901                    Additional contents and format of a full display are implementation-defined.

105902       **-Q**           Specify that the operand is a destination.

105903                    The *qstat* utility shall display information about each batch queue at each  
 105904                    destination identified as an operand.

105905       **-B**           Specify that the operand is a server name.

105906                    The *qstat* utility shall display information about each server identified as an  
 105907                    operand.

105908 **OPERANDS**

105909       If the **-Q** option is presented to the *qstat* utility, the utility shall accept one or more operands that  
 105910       conform to the syntax for a destination (see [Section 3.3.2](#), on page 2440).

105911       If the **-B** option is presented to the *qstat* utility, the utility shall accept one or more *server\_name*  
 105912       operands.

105913       If neither the **-B** nor the **-Q** option is presented to the *qstat* utility, the utility shall accept one or  
 105914       more operands that conform to the syntax for a batch *job\_identifier* (see [Section 3.3.1](#), on page  
 105915       2439).

105916 **STDIN**

105917 Not used.

105918 **INPUT FILES**

105919 None.

105920 **ENVIRONMENT VARIABLES**105921 The following environment variables shall affect the execution of *qstat*:105922 *HOME* Determine the pathname of the user's home directory.

105923 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 105924 (See XBD [Section 8.2](#) (on page 174) the precedence of internationalization variables  
 105925 used to determine the values of locale categories.)

105926 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 105927 internationalization variables.

105928 *LC\_COLLATE*

105929 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 105930 character collating elements within regular expressions.

105931 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 105932 characters (for example, single-byte as opposed to multi-byte characters in  
 105933 arguments).

105934 *LC\_MESSAGES*

105935 Determine the locale that should be used to affect the format and contents of  
 105936 diagnostic messages written to standard error.

105937 *LC\_NUMERIC*

105938 Determine the locale for selecting the radix character used when writing floating-  
 105939 point formatted output.

105940 **ASYNCHRONOUS EVENTS**

105941 Default.

105942 **STDOUT**

105943 If an operand presented to the *qstat* utility is a batch *job\_identifier* and the *-f* option is not  
 105944 specified, the *qstat* utility shall display the following items on a single line, in the stated order,  
 105945 with white space between each item, for each successfully processed operand:

- 105946 • The batch *job\_identifier*
- 105947 • The batch job name
- 105948 • The *Job\_Owner* attribute
- 105949 • The CPU time used by the batch job
- 105950 • The batch job state
- 105951 • The batch job location

105952 If an operand presented to the *qstat* utility is a batch *job\_identifier* and the *-f* option is specified,  
 105953 the *qstat* utility shall display the following items for each success fully processed operand:

- 105954 • The batch *job\_identifier*
- 105955 • The batch job name

- 105956 • The *Job\_Owner* attribute
  - 105957 • The execution user ID
  - 105958 • The CPU time used by the batch job
  - 105959 • The batch job state
  - 105960 • The batch job location
  - 105961 • Additional implementation-defined information, if any, about the batch job or batch queue
- 105962 If an operand presented to the *qstat* utility is a destination, the **-Q** option is specified, and the **-f**  
 105963 option is not specified, the *qstat* utility shall display the following items on a single line, in the  
 105964 stated order, with white space between each item, for each successfully processed operand:
- 105965 • The batch queue name
  - 105966 • The maximum number of batch jobs that shall be run in the batch queue concurrently
  - 105967 • The total number of batch jobs in the batch queue
  - 105968 • The status of the batch queue
  - 105969 • For each state, the number of batch jobs in that state in the batch queue and the name of  
 105970 the state
  - 105971 • The type of batch queue (execution or routing)
- 105972 If the operands presented to the *qstat* utility are destinations, the **-Q** option is specified, and the  
 105973 **-f** option is specified, the *qstat* utility shall display the following items for each successfully  
 105974 processed operand:
- 105975 • The batch queue name
  - 105976 • The maximum number of batch jobs that shall be run in the batch queue concurrently
  - 105977 • The total number of batch jobs in the batch queue
  - 105978 • The status of the batch queue
  - 105979 • For each state, the number of batch jobs in that state in the batch queue and the name of  
 105980 the state
  - 105981 • The type of batch queue (execution or routing)
  - 105982 • Additional implementation-defined information, if any, about the batch queue
- 105983 If the operands presented to the *qstat* utility are batch server names, the **-B** option is specified,  
 105984 and the **-f** option is not specified, the *qstat* utility shall display the following items on a single  
 105985 line, in the stated order, with white space between each item, for each successfully processed  
 105986 operand:
- 105987 • The batch server name
  - 105988 • The maximum number of batch jobs that shall be run in the batch queue concurrently
  - 105989 • The total number of batch jobs managed by the batch server
  - 105990 • The status of the batch server
  - 105991 • For each state, the number of batch jobs in that state and the name of the state
- 105992 If the operands presented to the *qstat* utility are server names, the **-B** option is specified, and the  
 105993 **-f** option is specified, the *qstat* utility shall display the following items for each successfully



105994 processed operand:

- 105995 • The server name
- 105996 • The maximum number of batch jobs that shall be run in the batch queue concurrently
- 105997 • The total number of batch jobs managed by the server
- 105998 • The status of the server
- 105999 • For each state, the number of batch jobs in that state and the name of the state
- 106000 • Additional implementation-defined information, if any, about the server

106001 **STDERR**

106002 The standard error shall be used only for diagnostic messages.

106003 **OUTPUT FILES**

106004 None.

106005 **EXTENDED DESCRIPTION**

106006 None.

106007 **EXIT STATUS**

106008 The following exit values shall be returned:

106009 0 Successful completion.

106010 >0 An error occurred.

106011 **CONSEQUENCES OF ERRORS**

106012 In addition to the default behavior, the *qstat* utility shall not be required to write a diagnostic

106013 message to standard error when the error reply received from a batch server indicates that the

106014 batch *job\_identifier* does not exist on the server. Whether or not the *qstat* utility waits to output

106015 the diagnostic message while attempting to locate the batch job on other servers is

106016 implementation-defined.

106017 **APPLICATION USAGE**

106018 None.

106019 **EXAMPLES**

106020 None.

106021 **RATIONALE**

106022 The *qstat* utility allows users to display the status of jobs and list the batch jobs in queues.

106023 The operands of the *qstat* utility may be either job identifiers, queues (specified as destination

106024 identifiers), or batch server names. The **-Q** and **-B** options, or absence thereof, indicate the

106025 nature of the operands.

106026 The other options of the *qstat* utility allow the user to control the amount of information

106027 displayed and the format in which it is displayed. Should a user wish to display the status of a

106028 set of jobs that match a selection criteria, the *qselect* utility may be used to acquire such a list.

106029 The **-f** option allows users to request a “full” display in an implementation-defined format.

106030 Historically, the *qstat* utility has been a part of the NQS and its derivatives, the existing practice

106031 on which it is based.

106032 **FUTURE DIRECTIONS**

106033       The *qstat* utility may be removed in a future version.

106034 **SEE ALSO**

106035       [Chapter 3](#) (on page 2417), [qselect](#)

106036       XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

106037 **CHANGE HISTORY**

106038       Derived from IEEE Std 1003.2d-1994.

106039 **Issue 6**

106040       IEEE PASC Interpretation 1003.2 #191 is applied, removing the following ENVIRONMENT  
106041       VARIABLES listed as affecting *qstat*: *COLUMNS*, *LINES*, *LOGNAME*, *TERM*, and *TZ*.

106042       The *LC\_TIME* entry is also removed from the ENVIRONMENT VARIABLES section.

106043 **Issue 7**

106044       The *qstat* utility is marked obsolescent.

106045       SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

106046 **NAME**

106047       qsub — submit a script

106048 **SYNOPSIS**

```

106049 OB BE  qsub [-a date_time] [-A account_string] [-c interval]
106050          [-C directive_prefix] [-e path_name] [-h] [-j join_list]
106051          [-k keep_list] [-m mail_options] [-M mail_list] [-N name]
106052          [-o path_name] [-p priority] [-q destination] [-r y|n]
106053          [-S path_name_list] [-u user_list] [-v variable_list] [-V]
106054          [-z] [script]

```

106055 **DESCRIPTION**

106056       To submit a script is to create a batch job that executes the script. A script is submitted by a  
 106057       request to a batch server. The *qsub* utility is a user-accessible batch client that submits a script.

106058       Upon successful completion, the *qsub* utility shall have created a batch job that will execute the  
 106059       submitted script.

106060       The *qsub* utility shall submit a script by sending a *Queue Job Request* to a batch server.

106061       The *qsub* utility shall place the value of the following environment variables in the *Variable\_List*  
 106062       attribute of the batch job: *HOME*, *LANG*, *LOGNAME*, *PATH*, *MAIL*, *SHELL*, and *TZ*. The name  
 106063       of the environment variable shall be the current name prefixed with the string *PBS\_O\_*.

106064       **Note:**     If the current value of the *HOME* variable in the environment space of the *qsub* utility is  
 106065       */aa/bb/cc*, then *qsub* shall place *PBS\_O\_HOME=/aa/bb/cc* in the *Variable\_List* attribute of the  
 106066       batch job.

106067       In addition to the variables described above, the *qsub* utility shall add the following variables  
 106068       with the indicated values to the variable list:

106069       *PBS\_O\_WORKDIR*   The absolute path of the current working directory of the *qsub* utility  
 106070       process.

106071       *PBS\_O\_HOST*     The name of the host on which the *qsub* utility is running.

106072 **OPTIONS**

106073       The *qsub* utility shall conform to XBD [Section 12.2](#) (on page 216).

106074       The following options shall be supported by the implementation:

106075       **-a date\_time**   Define the time at which a batch job becomes eligible for execution.

106076                       The *qsub* utility shall accept an option-argument that conforms to the syntax of the  
 106077       *time* operand of the *touch* utility.

106078

**Table 4-19** Environment Variable Values (Utilities)

106079

106080

106081

106082

106083

106084

106085

106086

106087

106088

| Variable Name        | Value at qsub Time        |
|----------------------|---------------------------|
| <i>PBS_O_HOME</i>    | <i>HOME</i>               |
| <i>PBS_O_HOST</i>    | Client host name          |
| <i>PBS_O_LANG</i>    | <i>LANG</i>               |
| <i>PBS_O_LOGNAME</i> | <i>LOGNAME</i>            |
| <i>PBS_O_PATH</i>    | <i>PATH</i>               |
| <i>PBS_O_MAIL</i>    | <i>MAIL</i>               |
| <i>PBS_O_SHELL</i>   | <i>SHELL</i>              |
| <i>PBS_O_TZ</i>      | <i>TZ</i>                 |
| <i>PBS_O_WORKDIR</i> | Current working directory |

106089

106090

**Note:** The server that initiates execution of the batch job will add other variables to the batch job's environment; see [Section 3.2.2.1](#) (on page 2423).

106091

106092

106093

106094

The *qsub* utility shall set the *Execution\_Time* attribute of the batch job to the number of seconds since the Epoch that is equivalent to the local time expressed by the value of the *date\_time* option-argument. The Epoch is defined in XBD [Section 3.150](#) (on page 57).

106095

106096

106097

If the *-a* option is not presented to the *qsub* utility, the utility shall set the *Execution\_Time* attribute of the batch job to a time (number of seconds since the Epoch) that is earlier than the time at which the utility exits.

106098

*-A account\_string*

106099

106100

Define the account to which the resource consumption of the batch job should be charged.

106101

The syntax of the *account\_string* option-argument is unspecified.

106102

106103

The *qsub* utility shall set the *Account\_Name* attribute of the batch job to the value of the *account\_string* option-argument.

106104

106105

If the *-A* option is not presented to the *qsub* utility, the utility shall omit the *Account\_Name* attribute from the attributes of the batch job.

106106

*-c interval*

Define whether the batch job should be checkpointed, and if so, how often.

106107

106108

The *qsub* utility shall accept a value for the interval option-argument that is one of the following:

106109

106110

*n* No checkpointing shall be performed on the batch job (NO\_CHECKPOINT).

106111

106112

*s* Checkpointing shall be performed only when the batch server is shut down (CHECKPOINT\_AT\_SHUTDOWN).

106113

106114

106115

*c* Automatic periodic checkpointing shall be performed at the *Minimum\_Cpu\_Interval* attribute of the batch queue, in units of CPU minutes (CHECKPOINT\_AT\_MIN\_CPU\_INTERVAL).

106116

106117

106118

106119

*c=minutes* Automatic periodic checkpointing shall be performed every *minutes* of CPU time, or every *Minimum\_Cpu\_Interval* minutes, whichever is greater. The *minutes* argument shall conform to the syntax for unsigned integers and shall be greater than zero.

106120

The *qsub* utility shall set the *Checkpoint* attribute of the batch job to the value of the

106121 *interval* option-argument.

106122 If the `-c` option is not presented to the *qsub* utility, the utility shall set the *Checkpoint*  
 106123 attribute of the batch job to the single character 'u'  
 106124 (CHECKPOINT\_UNSPECIFIED).

106125 **-C** *directive\_prefix*  
 106126 Define the prefix that declares a directive to the *qsub* utility within the script.

106127 The *directive\_prefix* is not a batch job attribute; it affects the behavior of the *qsub*  
 106128 utility.

106129 If the `-C` option is presented to the *qsub* utility, and the value of the *directive\_prefix*  
 106130 option-argument is the null string, the utility shall not scan the script file for  
 106131 directives. If the `-C` option is not presented to the *qsub* utility, then the value of the  
 106132 *PBS\_DPREFIX* environment variable is used. If the environment variable is not  
 106133 defined, then #PBS encoded in the portable character set is the default.

106134 **-e** *path\_name*  
 106135 Define the path to be used for the standard error stream of the batch job.

106136 The *qsub* utility shall accept a *path\_name* option-argument which can be preceded  
 106137 by a host name element of the form *hostname:*.

106138 If the *path\_name* option-argument constitutes an absolute pathname, the *qsub* utility  
 106139 shall set the *Error\_Path* attribute of the batch job to the value of the *path\_name*  
 106140 option-argument.

106141 If the *path\_name* option-argument constitutes a relative pathname and no host  
 106142 name element is specified, the *qsub* utility shall set the *Error\_Path* attribute of the  
 106143 batch job to the value of the absolute pathname derived by expanding the  
 106144 *path\_name* option-argument relative to the current directory of the process  
 106145 executing *qsub*.

106146 If the *path\_name* option-argument constitutes a relative pathname and a host name  
 106147 element is specified, the *qsub* utility shall set the *Error\_Path* attribute of the batch  
 106148 job to the value of the *path\_name* option-argument without expansion. The host  
 106149 name element shall be included.

106150 If the *path\_name* option-argument does not include a host name element, the *qsub*  
 106151 utility shall prefix the pathname with *hostname:*, where *hostname* is the name of the  
 106152 host upon which the *qsub* utility is being executed.

106153 If the `-e` option is not presented to the *qsub* utility, the utility shall set the  
 106154 *Error\_Path* attribute of the batch job to the host name and path of the current  
 106155 directory of the submitting process and the default filename.

106156 The default filename for standard error has the following format:  
 106157 *job\_name.esquence\_number*

106158 **-h** Specify that a USER hold is applied to the batch job.

106159 The *qsub* utility shall set the value of the *Hold\_Types* attribute of the batch job to the  
 106160 value USER.

106161 If the `-h` option is not presented to the *qsub* utility, the utility shall set the  
 106162 *Hold\_Types* attribute of the batch job to the value NO\_HOLD.

106163        **-j** *join\_list*    Define which streams of the batch job are to be merged. The *qsub* **-j** option shall  
 106164                            accept a value for the *join\_list* option-argument that is a string of alphanumeric  
 106165                            characters in the portable character set (see XBD [Section 6.1](#), on page 125).

106166                            The *qsub* utility shall accept a *join\_list* option-argument that consists of one or more  
 106167                            of the characters 'e' and 'o', or the single character 'n'.

106168                            All of the other batch job output streams specified will be merged into the output  
 106169                            stream represented by the character listed first in the *join\_list* option-argument.

106170                            For each unique character in the *join\_list* option-argument, the *qsub* utility shall  
 106171                            add a value to the *Join\_Path* attribute of the batch job as follows, each representing  
 106172                            a different batch job stream to join:

106173                            e    The standard error of the batch job (JOIN\_STD\_ERROR).  
 106174                            o    The standard output of the batch job (JOIN\_STD\_OUTPUT).

106175                            An existing *Join\_Path* attribute can be cleared by the following join type:

106176                            n    NO\_JOIN

106177                            If 'n' is specified, then no files are joined. The *qsub* utility shall consider it an error  
 106178                            if any join type other than 'n' is combined with join type 'n'.

106179                            Strictly conforming applications shall not repeat any of the characters 'e', 'o', or  
 106180                            'n' within the *join\_list* option-argument. The *qsub* utility shall permit the  
 106181                            repetition of characters, but shall not assign additional meaning to the repeated  
 106182                            characters.

106183                            An implementation may define other join types. The conformance document for an  
 106184                            implementation shall describe any additional batch job streams, how they are  
 106185                            specified, their internal behavior, and how they affect the behavior of the utility.

106186                            If the **-j** option is not presented to the *qsub* utility, the utility shall set the value of  
 106187                            the *Join\_Path* attribute of the batch job to NO\_JOIN.

106188        **-k** *keep\_list*    Define which output of the batch job to retain on the execution host.

106189                            The *qsub* **-k** option shall accept a value for the *keep\_list* option-argument that is a  
 106190                            string of alphanumeric characters in the portable character set (see XBD [Section](#)  
 106191                            [6.1](#), on page 125).

106192                            The *qsub* utility shall accept a *keep\_list* option-argument that consists of one or  
 106193                            more of the characters 'e' and 'o', or the single character 'n'.

106194                            For each unique character in the *keep\_list* option-argument, the *qsub* utility shall  
 106195                            add a value to the *Keep\_Files* attribute of the batch job as follows, each representing  
 106196                            a different batch job stream to keep:

106197                            e    The standard error of the batch job (KEEP\_STD\_ERROR).  
 106198                            o    The standard output of the batch job (KEEP\_STD\_OUTPUT).

106199                            If both 'e' and 'o' are specified, then both files are retained. An existing  
 106200                            *Keep\_Files* attribute can be cleared by the following keep type:

106201                            n    NO\_KEEP

106202                            If 'n' is specified, then no files are retained. The *qsub* utility shall consider it an  
 106203                            error if any keep type other than 'n' is combined with keep type 'n'.

106204 Strictly conforming applications shall not repeat any of the characters 'e', 'o', or  
 106205 'n' within the *keep\_list* option-argument. The *qsub* utility shall permit the  
 106206 repetition of characters, but shall not assign additional meaning to the repeated  
 106207 characters.

106208 An implementation may define other keep types. The conformance document for  
 106209 an implementation shall describe any additional keep types, how they are  
 106210 specified, their internal behavior, and how they affect the behavior of the utility. If  
 106211 the **-k** option is not presented to the *qsub* utility, the utility shall set the *Keep\_Files*  
 106212 attribute of the batch job to the value NO\_KEEP.

106213 **-m** *mail\_options*

106214 Define the points in the execution of the batch job at which the batch server that  
 106215 manages the batch job shall send mail about a change in the state of the batch job.

106216 The *qsub* **-m** option shall accept a value for the *mail\_options* option-argument that  
 106217 is a string of alphanumeric characters in the portable character set (see XBD [Section](#)  
 106218 [6.1](#), on page 125).

106219 The *qsub* utility shall accept a value for the *mail\_options* option-argument that is a  
 106220 string of one or more of the characters 'e', 'b', and 'a', or the single character  
 106221 'n'.

106222 For each unique character in the *mail\_options* option-argument, the *qsub* utility shall  
 106223 add a value to the *Mail\_Users* attribute of the batch job as follows, each  
 106224 representing a different time during the life of a batch job at which to send mail:

106225 e MAIL\_AT\_EXIT

106226 b MAIL\_AT\_BEGINNING

106227 a MAIL\_AT\_ABORT

106228 If any of these characters are duplicated in the *mail\_options* option-argument, the  
 106229 duplicates shall be ignored.

106230 An existing *Mail\_Points* attribute can be cleared by the following mail type:

106231 n NO\_MAIL

106232 If 'n' is specified, then mail is not sent. The *qsub* utility shall consider it an error if  
 106233 any mail type other than 'n' is combined with mail type 'n'.

106234 Strictly conforming applications shall not repeat any of the characters 'e', 'b',  
 106235 'a', or 'n' within the *mail\_options* option-argument.

106236 The *qsub* utility shall permit the repetition of characters, but shall not assign  
 106237 additional meaning to the repeated characters. An implementation may define  
 106238 other mail types. The conformance document for an implementation shall describe  
 106239 any additional mail types, how they are specified, their internal behavior, and how  
 106240 they affect the behavior of the utility.

106241 If the **-m** option is not presented to the *qsub* utility, the utility shall set the  
 106242 *Mail\_Points* attribute to the value MAIL\_AT\_ABORT.

106243 **-M** *mail\_list* Define the list of users to which a batch server that executes the batch job shall  
 106244 send mail, if the server sends mail about the batch job.

106245 The syntax of the *mail\_list* option-argument is unspecified.

106246 If the implementation of the *qsub* utility uses a name service to locate users, the



utility should accept the syntax used by the name service.

If the implementation of the *qsub* utility does not use a name service to locate users, the implementation should accept the following syntax for user names:

*mail\_address*[ , , *mail\_address* , , . . . ]

The interpretation of *mail\_address* is implementation-defined.

The *qsub* utility shall set the *Mail\_Users* attribute of the batch job to the value of the *mail\_list* option-argument.

If the **-M** option is not presented to the *qsub* utility, the utility shall place only the user name and host name for the current process in the *Mail\_Users* attribute of the batch job.

**-N name** Define the name of the batch job.

The *qsub* **-N** option shall accept a value for the *name* option-argument that is a string of up to 15 alphanumeric characters in the portable character set (see XBD Section 6.1, on page 125) where the first character is alphabetic.

The *qsub* utility shall set the value of the *Job\_Name* attribute of the batch job to the value of the *name* option-argument.

If the **-N** option is not presented to the *qsub* utility, the utility shall set the *Job\_Name* attribute of the batch job to the name of the *script* argument from which the directory specification if any, has been removed.

If the **-N** option is not presented to the *qsub* utility, and the script is read from standard input, the utility shall set the *Job\_Name* attribute of the batch job to the value STDIN.

**-o path\_name**

Define the path for the standard output of the batch job.

The *qsub* utility shall accept a *path\_name* option-argument that conforms to the syntax of the *path\_name* element defined in the System Interfaces volume of POSIX.1-2008, which can be preceded by a host name element of the form *hostname*..

If the *path\_name* option-argument constitutes an absolute pathname, the *qsub* utility shall set the *Output\_Path* attribute of the batch job to the value of the *path\_name* option-argument without expansion.

If the *path\_name* option-argument constitutes a relative pathname and no host name element is specified, the *qsub* utility shall set the *Output\_Path* attribute of the batch job to the pathname derived by expanding the value of the *path\_name* option-argument relative to the current directory of the process executing the *qsub*.

If the *path\_name* option-argument constitutes a relative pathname and a host name element is specified, the *qsub* utility shall set the *Output\_Path* attribute of the batch job to the value of the *path\_name* option-argument without expansion.

If the *path\_name* option-argument does not specify a host name element, the *qsub* utility shall prefix the pathname with *hostname*., where *hostname* is the name of the host upon which the *qsub* utility is executing.

If the **-o** option is not presented to the *qsub* utility, the utility shall set the *Output\_Path* attribute of the batch job to the host name and path of the current



|        |                          |                                                                                                                                                                                                                                      |
|--------|--------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 106290 |                          | directory of the submitting process and the default filename.                                                                                                                                                                        |
| 106291 |                          | The default filename for standard output has the following format:                                                                                                                                                                   |
| 106292 |                          | <i>job_name.osequence_number</i>                                                                                                                                                                                                     |
| 106293 | <b>-p priority</b>       | Define the priority the batch job should have relative to other batch jobs owned by the batch server.                                                                                                                                |
| 106294 |                          |                                                                                                                                                                                                                                      |
| 106295 |                          | The <i>qsub</i> utility shall set the <i>Priority</i> attribute of the batch job to the value of the <i>priority</i> option-argument.                                                                                                |
| 106296 |                          |                                                                                                                                                                                                                                      |
| 106297 |                          | If the <b>-p</b> option is not presented to the <i>qsub</i> utility, the value of the <i>Priority</i> attribute is implementation-defined.                                                                                           |
| 106298 |                          |                                                                                                                                                                                                                                      |
| 106299 |                          | The <i>qsub</i> utility shall accept a value for the <i>priority</i> option-argument that conforms to the syntax for signed decimal integers, and which is not less than -1 024 and not greater than 1 023.                          |
| 106300 |                          |                                                                                                                                                                                                                                      |
| 106301 |                          |                                                                                                                                                                                                                                      |
| 106302 | <b>-q destination</b>    | Define the destination of the batch job.                                                                                                                                                                                             |
| 106303 |                          |                                                                                                                                                                                                                                      |
| 106304 |                          | The destination is not a batch job attribute; it determines the batch server, and possibly the batch queue, to which the <i>qsub</i> utility batch queues the batch job.                                                             |
| 106305 |                          |                                                                                                                                                                                                                                      |
| 106306 |                          | The <i>qsub</i> utility shall submit the script to the batch server named by the <i>destination</i> option-argument or the server that owns the batch queue named in the <i>destination</i> option-argument.                         |
| 106307 |                          |                                                                                                                                                                                                                                      |
| 106308 |                          |                                                                                                                                                                                                                                      |
| 106309 |                          | The <i>qsub</i> utility shall accept an option-argument for the <b>-q</b> option that conforms to the syntax for a destination (see <a href="#">Section 3.3.2</a> , on page 2440).                                                   |
| 106310 |                          |                                                                                                                                                                                                                                      |
| 106311 |                          | If the <b>-q</b> option is not presented to the <i>qsub</i> utility, the <i>qsub</i> utility shall submit the batch job to the default destination. The mechanism for determining the default destination is implementation-defined. |
| 106312 |                          |                                                                                                                                                                                                                                      |
| 106313 |                          |                                                                                                                                                                                                                                      |
| 106314 | <b>-r y   n</b>          | Define whether the batch job is rerunnable.                                                                                                                                                                                          |
| 106315 |                          | If the value of the option-argument is <i>y</i> , the <i>qsub</i> utility shall set the <i>Rerunable</i> attribute of the batch job to TRUE.                                                                                         |
| 106316 |                          |                                                                                                                                                                                                                                      |
| 106317 |                          | If the value of the option-argument is <i>n</i> , the <i>qsub</i> utility shall set the <i>Rerunable</i> attribute of the batch job to FALSE.                                                                                        |
| 106318 |                          |                                                                                                                                                                                                                                      |
| 106319 |                          | If the <b>-r</b> option is not presented to the <i>qsub</i> utility, the utility shall set the <i>Rerunable</i> attribute of the batch job to TRUE.                                                                                  |
| 106320 |                          |                                                                                                                                                                                                                                      |
| 106321 | <b>-S path_name_list</b> | Define the pathname to the shell under which the batch job is to execute.                                                                                                                                                            |
| 106322 |                          |                                                                                                                                                                                                                                      |
| 106323 |                          | The <i>qsub</i> utility shall accept a <i>path_name_list</i> option-argument that conforms to the following syntax:                                                                                                                  |
| 106324 |                          |                                                                                                                                                                                                                                      |
| 106325 |                          | <i>pathname</i> [ <i>@host</i> ][ <i>,</i> <i>,</i> <i>pathname</i> [ <i>@host</i> ] <i>,</i> <i>,</i> . . . ]                                                                                                                       |
| 106326 |                          | The <i>qsub</i> utility shall allow only one pathname for a given host name. The <i>qsub</i> utility shall allow only one pathname that is missing a corresponding host name.                                                        |
| 106327 |                          |                                                                                                                                                                                                                                      |
| 106328 |                          | The <i>qsub</i> utility shall add a value to the <i>Shell_Path_List</i> attribute of the batch job for each entry in the <i>path_name_list</i> option-argument.                                                                      |
| 106329 |                          |                                                                                                                                                                                                                                      |
| 106330 |                          | If the <b>-S</b> option is not presented to the <i>qsub</i> utility, the utility shall set the                                                                                                                                       |

106331 *Shell\_Path\_List* attribute of the batch job to the null string.

106332 The conformance document for an implementation shall describe the mechanism  
 106333 used to set the default shell and determine the current value of the default shell.  
 106334 An implementation shall provide a means for the installation to set the default  
 106335 shell to the login shell of the user under which the batch job is to execute. See  
 106336 [Section 3.3.3](#) (on page 2441) for a means of removing *keyword=value* (and  
 106337 *value@keyword*) pairs and other general rules for list-oriented batch job attributes.

106338 **-u *user\_list*** Define the user name under which the batch job is to execute.

106339 The *qsub* utility shall accept a *user\_list* option-argument that conforms to the  
 106340 following syntax:

106341 *username[@host][, ,username[@host], , ...]*

106342 The *qsub* utility shall accept only one user name that is missing a corresponding  
 106343 host name. The *qsub* utility shall accept only one user name per named host.

106344 The *qsub* utility shall add a value to the *User\_List* attribute of the batch job for each  
 106345 entry in the *user\_list* option-argument.

106346 If the **-u** option is not presented to the *qsub* utility, the utility shall set the *User\_List*  
 106347 attribute of the batch job to the user name from which the utility is executing. See  
 106348 [Section 3.3.3](#) (on page 2441) for a means of removing *keyword=value* (and  
 106349 *value@keyword*) pairs and other general rules for list-oriented batch job attributes.

106350 **-v *variable\_list***

106351 Add to the list of variables that are exported to the session leader of the batch job.

106352 A *variable\_list* is a set of strings of either the form *<variable>* or *<variable=value>*,  
 106353 delimited by *<comma>* characters.

106354 If the **-v** option is presented to the *qsub* utility, the utility shall also add, to the  
 106355 environment *Variable\_List* attribute of the batch job, every variable named in the  
 106356 environment *variable\_list* option-argument and, optionally, values of specified  
 106357 variables.

106358 If a value is not provided on the command line, the *qsub* utility shall set the value  
 106359 of each variable in the environment *Variable\_List* attribute of the batch job to the  
 106360 value of the corresponding environment variable for the process in which the  
 106361 utility is executing; see [Table 4-19](#) (on page 3168).

106362 A conforming application shall not repeat a variable in the environment  
 106363 *variable\_list* option-argument.

106364 The *qsub* utility shall not repeat a variable in the environment *Variable\_List* attribute  
 106365 of the batch job. See [Section 3.3.3](#) (on page 2441) for a means of removing  
 106366 *keyword=value* (and *value@keyword*) pairs and other general rules for list-oriented  
 106367 batch job attributes.

106368 **-V** Specify that all of the environment variables of the process are exported to the  
 106369 context of the batch job.

106370 The *qsub* utility shall place every environment variable in the process in which the  
 106371 utility is executing in the list and shall set the value of each variable in the attribute  
 106372 to the value of that variable in the process.

106373            **-z**            Specify that the utility does not write the batch *job\_identifier* of the created batch job  
106374            to standard output.

106375            If the **-z** option is presented to the *qsub* utility, the utility shall not write the batch  
106376            *job\_identifier* of the created batch job to standard output.

106377            If the **-z** option is not presented to the *qsub* utility, the utility shall write the  
106378            identifier of the created batch job to standard output.

#### 106379 OPERANDS

106380            The *qsub* utility shall accept a *script* operand that indicates the path to the script of the batch job.

106381            If the *script* operand is not presented to the *qsub* utility, or if the operand is the single-character  
106382            string *'-'*, the utility shall read the script from standard input.

106383            If the script represents a partial path, the *qsub* utility shall expand the path relative to the current  
106384            directory of the process executing the utility.

#### 106385 STDIN

106386            The *qsub* utility reads the script of the batch job from standard input if the script operand is  
106387            omitted or is the single character *'-'*.

#### 106388 INPUT FILES

106389            In addition to binding the file indicated by the *script* operand to the batch job, the *qsub* utility  
106390            reads the script file and acts on directives in the script.

#### 106391 ENVIRONMENT VARIABLES

106392            The following environment variables shall affect the execution of *qsub*:

106393            *LANG*            Provide a default value for the internationalization variables that are unset or null.  
106394            (See XBD Section 8.2 (on page 174) the precedence of internationalization variables  
106395            used to determine the values of locale categories.)

106396            *LC\_ALL*            If set to a non-empty string value, override the values of all the other  
106397            internationalization variables.

106398            *LC\_CTYPE*          Determine the locale for the interpretation of sequences of bytes of text data as  
106399            characters (for example, single-byte as opposed to multi-byte characters in  
106400            arguments).

106401            *LC\_MESSAGES*

106402            Determine the locale that should be used to affect the format and contents of  
106403            diagnostic messages written to standard error.

106404            *LOGNAME*          Determine the login name of the user.

106405            *PBS\_DPREFIX*

106406            Determine the default prefix for directives within the script.

106407            *SHELL*            Determine the pathname of the preferred command language interpreter of the  
106408            user.

106409            *TZ*                Determine the timezone used to interpret the *date-time* option-argument. If *TZ* is  
106410            unset or null, an unspecified default timezone shall be used.

#### 106411 ASYNCHRONOUS EVENTS

106412            Once created, a batch job exists until it exits, aborts, or is deleted.

106413            After a batch job is created by the *qsub* utility, batch servers might route, execute, modify, or  
106414            delete the batch job.

106415 **STDOUT**

106416 The *qsub* utility writes the batch *job\_identifier* assigned to the batch job to standard output, unless  
106417 the **-z** option is specified.

106418 **STDERR**

106419 The standard error shall be used only for diagnostic messages.

106420 **OUTPUT FILES**

106421 None.

106422 **EXTENDED DESCRIPTION**106423 **Script Preservation**

106424 The *qsub* utility shall make the script available to the server executing the batch job in such a  
106425 way that the server executes the script as it exists at the time of submission.

106426 The *qsub* utility can send a copy of the script to the server with the *Queue Job Request* or store a  
106427 temporary copy of the script in a location specified to the server.

106428 **Option Specification**

106429 A script can contain directives to the *qsub* utility.

106430 The *qsub* utility shall scan the lines of the script for directives, skipping blank lines, until the first  
106431 line that begins with a string other than the directive string; if directives occur on subsequent  
106432 lines, the utility shall ignore those directives.

106433 Lines are separated by a <newline>. If the first line of the script begins with "#!" or a <colon>  
106434 (' : '), then it is skipped. The *qsub* utility shall process a line in the script as a directive if and  
106435 only if the string of characters from the first non-white-space character on the line until the first  
106436 <space> or <tab> on the line match the directive prefix. If a line in the script contains a directive  
106437 and the final characters of the line are <backslash> and <newline>, then the next line shall be  
106438 interpreted as a continuation of that directive.

106439 The *qsub* utility shall process the options and option-arguments contained on the directive prefix  
106440 line using the same syntax as if the options were input on the *qsub* utility.

106441 The *qsub* utility shall continue to process a directive prefix line until after a <newline> is  
106442 encountered. An implementation may ignore lines which, according to the syntax of the shell  
106443 that will interpret the script, are comments. An implementation shall describe in the  
106444 conformance document the format of any shell comments that it will recognize.

106445 If an option is present in both a directive and the arguments to the *qsub* utility, the utility shall  
106446 ignore the option and the corresponding option-argument, if any, in the directive.

106447 If an option that is present in the directive is not present in the arguments to the *qsub* utility, the  
106448 utility shall process the option and the option-argument, if any.

106449 In order of preference, the *qsub* utility shall select the directive prefix from one of the following  
106450 sources:

- 106451 • If the **-C** option is presented to the utility, the value of the *directive\_prefix* option-argument
- 106452 • If the environment variable *PBS\_DPPREFIX* is defined, the value of that variable
- 106453 • The four-character string "#PBS" encoded in the portable character set

106454 If the **-C** option is present in the script file it shall be ignored.

**106455 EXIT STATUS**

106456 The following exit values shall be returned:

106457 0 Successful completion.

106458 >0 An error occurred.

**106459 CONSEQUENCES OF ERRORS**

106460 Default.

**106461 APPLICATION USAGE**

106462 None.

**106463 EXAMPLES**

106464 None.

**106465 RATIONALE**

106466 The *qsub* utility allows users to create a batch job that will process the script specified as the  
106467 operand of the utility.

106468 The options of the *qsub* utility allow users to control many aspects of the queuing and execution  
106469 of a batch job.

106470 The **-a** option allows users to designate the time after which the batch job will become eligible to  
106471 run. By specifying an execution time, users can take advantage of resources at off-peak hours,  
106472 synchronize jobs with chronologically predictable events, and perhaps take advantage of off-  
106473 peak pricing of computing time. For these reasons and others, a timing option is existing  
106474 practice on the part of almost every batch system, including NQS.

106475 The **-A** option allows users to specify the account that will be charged for the batch job. Support  
106476 for account is not mandatory for conforming batch servers.

106477 The **-C** option allows users to prescribe the prefix for directives within the script file. The default  
106478 prefix "**#PBS**" may be inappropriate if the script will be interpreted with an alternate shell, as  
106479 specified by the **-S** option.

106480 The **-c** option allows users to establish the checkpointing interval for their jobs. A checkpointing  
106481 system, which is not defined by this volume of POSIX.1-2008, allows recovery of a batch job at  
106482 the most recent checkpoint in the event of a crash. Checkpointing is typically used for jobs that  
106483 consume expensive computing time or must meet a critical schedule. Users should be allowed to  
106484 make the tradeoff between the overhead of checkpointing and the risk to the timely completion  
106485 of the batch job; therefore, this volume of POSIX.1-2008 provides the checkpointing interval  
106486 option. Support for checkpointing is optional for batch servers.

106487 The **-e** option allows users to redirect the standard error streams of their jobs to a non-default  
106488 path. For example, if the submitted script generally produces a great deal of useless error  
106489 output, a user might redirect the standard error output to the null device. Or, if the file system  
106490 holding the default location (the home directory of the user) has too little free space, the user  
106491 might redirect the standard error stream to a file in another file system.

106492 The **-h** option allows users to create a batch job that is held until explicitly released. The ability  
106493 to create a held job is useful when some external event must complete before the batch job can  
106494 execute. For example, the user might submit a held job and release it when the system load has  
106495 dropped.

106496 The **-j** option allows users to merge the standard error of a batch job into its standard output  
106497 stream, which has the advantage of showing the sequential relationship between output and  
106498 error messages.

The **-m** option allows users to designate those points in the execution of a batch job at which mail will be sent to the submitting user, or to the account(s) indicated by the **-M** option. By requesting mail notification at points of interest in the life of a job, the submitting user, or other designated users, can track the progress of a batch job.

The **-N** option allows users to associate a name with the batch job. The job name in no way affects the processing of the batch job, but rather serves as a mnemonic handle for users. For example, the batch job name can help the user distinguish between multiple jobs listed by the *qstat* utility.

The **-o** option allows users to redirect the standard output stream. A user might, for example, wish to redirect to the null device the standard output stream of a job that produces copious yet superfluous output.

The **-P** option allows users to designate the relative priority of a batch job for selection from a queue.

The **-q** option allows users to specify an initial queue for the batch job. If the user specifies a routing queue, the batch server routes the batch job to another queue for execution or further routing. If the user specifies a non-routing queue, the batch server of the queue eventually executes the batch job.

The **-r** option allows users to control whether the submitted job will be rerun if the controlling batch node fails during execution of the batch job. The **-r** option likewise allows users to indicate whether or not the batch job is eligible to be rerun by the *qrerun* utility. Some jobs cannot be correctly rerun because of changes they make in the state of databases or other aspects of their environment. This volume of POSIX.1-2008 specifies that the default, if the **-r** option is not presented to the utility, will be that the batch job cannot be rerun, since the result of rerunning a non-rerunnable job might be catastrophic.

The **-S** option allows users to specify the program (usually a shell) that will be invoked to process the script of the batch job. This option has been modified to allow a list of shell names and locations associated with different hosts.

The **-u** option is useful when the submitting user is authorized to use more than one account on a given host, in which case the **-u** option allows the user to select from among those accounts. The option-argument is a list of user-host pairs, so that the submitting user can provide different user identifiers for different nodes in the event the batch job is routed. The **-u** option provides a lot of flexibility to accommodate sites with complex account structures. Users that have the same user identifier on all the hosts they are authorized to use will not need to use the **-u** option.

The **-V** option allows users to export all their current environment variables, as of the time the batch job is submitted, to the context of the processes of the batch job.

The **-v** option allows users to export specific environment variables from their current process to the processes of the batch job.

The **-z** option allows users to suppress the writing of the batch job identifier to standard output. The **-z** option is an existing NQS practice that has been standardized.

Historically, the *qsub* utility has served the batch job-submission function in the NQS system, the existing practice on which it is based. Some changes and additions have been made to the *qsub* utility in this volume of POSIX.1-2008, *vis-a-vis* NQS, as a result of the growing pool of experience with distributed batch systems.

The set of features of the *qsub* utility as defined in this volume of POSIX.1-2008 appears to incorporate all the common existing practice on potentially conforming platforms.

106544 **FUTURE DIRECTIONS**

106545       The *qsub* utility may be removed in a future version.

106546 **SEE ALSO**

106547       Chapter 3 (on page 2417), *qrerun*, *qstat*, *touch*

106548       XBD Section 3.150 (on page 57), Section 6.1 (on page 125), Chapter 8 (on page 173), Section 12.2  
106549       (on page 216)

106550 **CHANGE HISTORY**

106551       Derived from IEEE Std 1003.2d-1994.

106552 **Issue 6**

106553       The *-I* option has been removed as there is no portable description of the resources that are  
106554       allowed or required by the batch job.

106555 **Issue 7**

106556       The *qsub* utility is marked obsolescent.

106557       SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



106558 **NAME**106559 `read` — read from standard input into shell variables |106560 **SYNOPSIS**106561 `read [-r] var...`106562 **DESCRIPTION**106563 The *read* utility shall read a single logical line from standard input into one or more shell |  
106564 variables.106565 By default, unless the `-r` option is specified, `<backslash>` shall act as an escape character. An  
106566 unescaped `<backslash>` shall preserve the literal value of the following character, with the  
106567 exception of a `<newline>`. If a `<newline>` follows the `<backslash>`, the *read* utility shall interpret  
106568 this as line continuation. The `<backslash>` and `<newline>` shall be removed before splitting the  
106569 input into fields. All other unescaped `<backslash>` characters shall be removed after splitting the  
106570 input into fields.106571 If standard input is a terminal device and the invoking shell is interactive, *read* shall prompt for a  
106572 continuation line when it reads an input line ending with a `<backslash>` `<newline>`, unless the  
106573 `-r` option is specified.106574 The terminating `<newline>` (if any) shall be removed from the input and the results shall be split  
106575 into fields as in the shell for the results of parameter expansion (see [Section 2.6.5](#), on page 2349);  
106576 the first field shall be assigned to the first variable *var*, the second field to the second variable  
106577 *var*, and so on. If there are fewer fields than there are *var* operands, the remaining *vars* shall be  
106578 set to empty strings. If there are fewer *vars* than fields, the last *var* shall be set to a value  
106579 comprising the following elements:

- 106580
- The field that corresponds to the last *var* in the normal assignment sequence described  
106581 above
  - The delimiter(s) that follow the field corresponding to the last *var*
  - The remaining fields and their delimiters, with trailing *IFS* white space ignored

106584 The setting of variables specified by the *var* operands shall affect the current shell execution  
106585 environment; see [Section 2.12](#) (on page 2371). If it is called in a subshell or separate utility  
106586 execution environment, such as one of the following:106587 

```
(read foo)
```

  
106588 

```
nohup read ...
```

  
106589 

```
find . -exec read ... \;
```

106590 it shall not affect the shell variables in the caller's environment.

106591 **OPTIONS**106592 The *read* utility shall conform to XBD [Section 12.2](#) (on page 216).

106593 The following option is supported:

106594 `-r` Do not treat a `<backslash>` character in any special way. Consider each  
106595 `<backslash>` to be part of the input line.106596 **OPERANDS**

106597 The following operand shall be supported:

106598 *var* The name of an existing or nonexisting shell variable.



106599 **STDIN**

106600 The standard input shall be a text file.

106601 **INPUT FILES**

106602 None.

106603 **ENVIRONMENT VARIABLES**106604 The following environment variables shall affect the execution of *read*:

106605 *IFS* Determine the internal field separators used to delimit fields; see [Section 2.5.3](#) (on  
106606 page 2341).

106607 *LANG* Provide a default value for the internationalization variables that are unset or null.  
106608 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
106609 variables used to determine the values of locale categories.)

106610 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
106611 internationalization variables.

106612 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
106613 characters (for example, single-byte as opposed to multi-byte characters in  
106614 arguments).

106615 *LC\_MESSAGES*

106616 Determine the locale that should be used to affect the format and contents of  
106617 diagnostic messages written to standard error.

106618 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

106619 *PS2* Provide the prompt string that an interactive shell shall write to standard error  
106620 when a line ending with a <backslash> <newline> is read and the *-r* option was  
106621 not specified.

106622 **ASYNCHRONOUS EVENTS**

106623 Default.

106624 **STDOUT**

106625 Not used.

106626 **STDERR**

106627 The standard error shall be used for diagnostic messages and prompts for continued input.

106628 **OUTPUT FILES**

106629 None.

106630 **EXTENDED DESCRIPTION**

106631 None.

106632 **EXIT STATUS**

106633 The following exit values shall be returned:

106634 0 Successful completion.

106635 &gt;0 End-of-file was detected or an error occurred.

106636 **CONSEQUENCES OF ERRORS**

106637 Default.

**APPLICATION USAGE**

The `-r` option is included to enable *read* to subsume the purpose of the *line* utility, which is not included in POSIX.1-2008.

**EXAMPLES**

The following command:

```
while read -r xx yy
do
    printf "%s %s\n" "$yy" "$xx"
done < input_file
```

prints a file with the first field of each line moved to the end of the line.

**RATIONALE**

The *read* utility historically has been a shell built-in. It was separated off into its own utility to take advantage of the richer description of functionality introduced by this volume of POSIX.1-2008.

Since *read* affects the current shell execution environment, it is generally provided as a shell regular built-in. If it is called in a subshell or separate utility execution environment, such as one of the following:

```
(read foo)
nohup read ...
find . -exec read ... \;
```

it does not affect the shell variables in the environment of the caller.

Although the standard input is required to be a text file, and therefore will always end with a `<newline>` (unless it is an empty file), the processing of continuation lines when the `-r` option is not used can result in the input not ending with a `<newline>`. This occurs if the last line of the input file ends with a `<backslash>` `<newline>`. It is for this reason that “if any” is used in “The terminating `<newline>` (if any) shall be removed from the input” in the description. It is not a relaxation of the requirement for standard input to be a text file.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

[Chapter 2](#) (on page 2335)

XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

**CHANGE HISTORY**

First released in Issue 2.

**Issue 7**

Austin Group Interpretation 1003.1-2001 #194 is applied, clarifying the handling of the `<backslash>` escape character.

SD5-XCU-ERN-126 is applied, clarifying that input lines end with a `<newline>`.

The description of here-documents is removed from the *read* reference page.

106677 **NAME**106678        *renice* — set nice values of running processes106679 **SYNOPSIS**106680        *renice* [-g|-p|-u] -n *increment ID...*106681 **DESCRIPTION**

106682        The *renice* utility shall request that the nice values (see XBD [Section 3.244](#), on page 72) of one or  
 106683        more running processes be changed. By default, the applicable processes are specified by their  
 106684        process IDs. When a process group is specified (see -g), the request shall apply to all processes  
 106685        in the process group.

106686        The nice value shall be bounded in an implementation-defined manner. If the requested  
 106687        *increment* would raise or lower the nice value of the executed utility beyond implementation-  
 106688        defined limits, then the limit whose value was exceeded shall be used.

106689        When a user is *reniced*, the request applies to all processes whose saved set-user-ID matches the  
 106690        user ID corresponding to the user.

106691        Regardless of which options are supplied or any other factor, *renice* shall not alter the nice values  
 106692        of any process unless the user requesting such a change has appropriate privileges to do so for  
 106693        the specified process. If the user lacks appropriate privileges to perform the requested action, the  
 106694        utility shall return an error status.

106695        The saved set-user-ID of the user's process shall be checked instead of its effective user ID when  
 106696        *renice* attempts to determine the user ID of the process in order to determine whether the user  
 106697        has appropriate privileges.

106698 **OPTIONS**106699        The *renice* utility shall conform to XBD [Section 12.2](#) (on page 216), except for Guideline 9.

106700        The following options shall be supported:

106701        -g        Interpret the following operands as unsigned decimal integer process group IDs.

106702        -n *increment* Specify how the nice value of the specified process or processes is to be adjusted.  
 106703        The *increment* option-argument is a positive or negative decimal integer that shall  
 106704        be used to modify the nice value of the specified process or processes.

106705        Positive *increment* values shall cause a lower nice value. Negative *increment* values  
 106706        may require appropriate privileges and shall cause a higher nice value.

106707        -p        Interpret the following operands as unsigned decimal integer process IDs. The -p  
 106708        option is the default if no options are specified.

106709        -u        Interpret the following operands as users. If a user exists with a user name equal to  
 106710        the operand, then the user ID of that user is used in further processing. Otherwise,  
 106711        if the operand represents an unsigned decimal integer, it shall be used as the  
 106712        numeric user ID of the user.

106713 **OPERANDS**

106714        The following operands shall be supported:

106715        ID        A process ID, process group ID, or user name/user ID, depending on the option  
 106716        selected.

106717 **STDIN**

106718        Not used.

106719 **INPUT FILES**

106720 None.

106721 **ENVIRONMENT VARIABLES**106722 The following environment variables shall affect the execution of *renice*:

106723 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 106724 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 106725 variables used to determine the values of locale categories.)

106726 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 106727 internationalization variables.

106728 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 106729 characters (for example, single-byte as opposed to multi-byte characters in  
 106730 arguments).

106731 *LC\_MESSAGES*

106732 Determine the locale that should be used to affect the format and contents of  
 106733 diagnostic messages written to standard error.

106734 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

106735 **ASYNCHRONOUS EVENTS**

106736 Default.

106737 **STDOUT**

106738 Not used.

106739 **STDERR**

106740 The standard error shall be used only for diagnostic messages.

106741 **OUTPUT FILES**

106742 None.

106743 **EXTENDED DESCRIPTION**

106744 None.

106745 **EXIT STATUS**

106746 The following exit values shall be returned:

106747 0 Successful completion.

106748 &gt;0 An error occurred.

106749 **CONSEQUENCES OF ERRORS**

106750 Default.

106751 **APPLICATION USAGE**

106752 None.

106753 **EXAMPLES**

106754 1. Adjust the nice value so that process IDs 987 and 32 would have a lower nice value:

106755 `renice -n 5 -p 987 32`

106756 2. Adjust the nice value so that group IDs 324 and 76 would have a higher nice value, if the  
 106757 user has appropriate privileges to do so:

106758 `renice -n -4 -g 324 76`

3. Adjust the nice value so that numeric user ID 8 and user **sas** would have a lower nice value:

```
renice -n 4 -u 8 sas
```

Useful nice value increments on historical systems include 19 or 20 (the affected processes run only when nothing else in the system attempts to run) and any negative number (to make processes run faster).

#### RATIONALE

The *gid*, *pid*, and *user* specifications do not fit either the definition of operand or option-argument. However, for clarity, they have been included in the OPTIONS section, rather than the OPERANDS section.

The definition of nice value is not intended to suggest that all processes in a system have priorities that are comparable. Scheduling policy extensions such as the realtime priorities in the System Interfaces volume of POSIX.1-2008 make the notion of a single underlying priority for all scheduling policies problematic. Some implementations may implement the *nice*-related features to affect all processes on the system, others to affect just the general time-sharing activities implied by this volume of POSIX.1-2008, and others may have no effect at all. Because of the use of “implementation-defined” in *nice* and *renice*, a wide range of implementation strategies are possible.

Originally, this utility was written in the historical manner, using the term “nice value”. This was always a point of concern with users because it was never intuitively obvious what this meant. With a newer version of *renice*, which used the term “system scheduling priority”, it was hoped that novice users could better understand what this utility was meant to do. Also, it would be easier to document what the utility was meant to do. Unfortunately, the addition of the POSIX realtime scheduling capabilities introduced the concepts of process and thread scheduling priorities that were totally unaffected by the *nice/renice* utilities or the *nice()/setpriority()* functions. Continuing to use the term “system scheduling priority” would have incorrectly suggested that these utilities and functions were indeed affecting these realtime priorities. It was decided to revert to the historical term “nice value” to reference this unrelated process attribute.

Although this utility has use by system administrators (and in fact appears in the system administration portion of the BSD documentation), the standard developers considered that it was very useful for individual end users to control their own processes.

Earlier versions of this standard allowed the following forms in the SYNOPSIS:

```
renice nice_value[-p] pid...[-g gid...][-p pid...][-u user...]  
renice nice_value -g gid...[-g gid...]-p pid...[-u user...]  
renice nice_value -u user...[-g gid...]-p pid...[-u user...]
```

These forms are no longer specified by POSIX.1-2008 but may be present in some implementations.

#### FUTURE DIRECTIONS

None.

#### SEE ALSO

*nice*

XBD [Section 3.244](#) (on page 72), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

106802 **CHANGE HISTORY**

106803 First released in Issue 4.

106804 **Issue 5**

106805 In the SYNOPSIS, an ellipsis is added to the **-u** option in all three obsolescent forms.

106806 **Issue 6**

106807 This utility is marked as part of the User Portability Utilities option.

106808 The APPLICATION USAGE section is added.

106809 The obsolescent forms of the SYNOPSIS are removed.

106810 Text previously conditional on POSIX\_SAVED\_IDS is mandatory in this version. This is a FIPS  
106811 requirement.

106812 **Issue 7**

106813 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying that Guideline 9 of the Utility  
106814 Syntax Guidelines does not apply.

106815 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

106816 The *renice* utility is moved from the User Portability Utilities option to the Base. User Portability  
106817 Utilities is now an option for interactive utilities.

106818 **NAME**106819 `rm` — remove directory entries106820 **SYNOPSIS**106821 `rm [-iRr] file...` |106822 `rm -f [-iRr] [file...]` |106823 **DESCRIPTION**106824 The *rm* utility shall remove the directory entry specified by each *file* argument.

106825 If either of the files dot or dot-dot are specified as the basename portion of an operand (that is,  
 106826 the final pathname component) or if an operand resolves to the root directory, *rm* shall write a  
 106827 diagnostic message to standard error and do nothing more with such operands.

106828 For each *file* the following steps shall be taken:106829 1. If the *file* does not exist:

106830 a. If the `-f` option is not specified, *rm* shall write a diagnostic message to standard  
 106831 error.

106832 b. Go on to any remaining *files*.106833 2. If *file* is of type directory, the following steps shall be taken:

106834 a. If neither the `-R` option nor the `-r` option is specified, *rm* shall write a diagnostic  
 106835 message to standard error, do nothing more with *file*, and go on to any remaining  
 106836 files.

106837 b. If *file* is an empty directory, *rm* may skip to step 2d. If the `-f` option is not specified, +  
 106838 and either the permissions of *file* do not permit writing and the standard input is a  
 106839 terminal or the `-i` option is specified, *rm* shall write a prompt to standard error and  
 106840 read a line from the standard input. If the response is not affirmative, *rm* shall do  
 106841 nothing more with the current file and go on to any remaining files.

106842 c. For each entry contained in *file*, other than dot or dot-dot, the four steps listed here  
 106843 (1 to 4) shall be taken with the entry as if it were a *file* operand. The *rm* utility shall  
 106844 not traverse directories by following symbolic links into other parts of the  
 106845 hierarchy, but shall remove the links themselves.

106846 d. If the `-i` option is specified, *rm* shall write a prompt to standard error and read a  
 106847 line from the standard input. If the response is not affirmative, *rm* shall do nothing  
 106848 more with the current file, and go on to any remaining files.

106849 3. If *file* is not of type directory, the `-f` option is not specified, and either the permissions of  
 106850 *file* do not permit writing and the standard input is a terminal or the `-i` option is specified,  
 106851 *rm* shall write a prompt to the standard error and read a line from the standard input. If  
 106852 the response is not affirmative, *rm* shall do nothing more with the current file and go on  
 106853 to any remaining files.

106854 4. If the current file is a directory, *rm* shall perform actions equivalent to the *rmdir()* function  
 106855 defined in the System Interfaces volume of POSIX.1-2008 called with a pathname of the  
 106856 current file used as the *path* argument. If the current file is not a directory, *rm* shall  
 106857 perform actions equivalent to the *unlink()* function defined in the System Interfaces  
 106858 volume of POSIX.1-2008 called with a pathname of the current file used as the *path*  
 106859 argument.

106860 If this fails for any reason, *rm* shall write a diagnostic message to standard error, do  
 106861 nothing more with the current file, and go on to any remaining files.

106862 The *rm* utility shall be able to descend to arbitrary depths in a file hierarchy, and shall not fail  
 106863 due to path length limitations (unless an operand specified by the user exceeds system  
 106864 limitations).

#### 106865 OPTIONS

106866 The *rm* utility shall conform to XBD [Section 12.2](#) (on page 216).

106867 The following options shall be supported:

- 106868 **-f** Do not prompt for confirmation. Do not write diagnostic messages or modify the |  
 106869 exit status in the case of no file operands, or in the case of operands that do not |  
 106870 exist. Any previous occurrences of the **-i** option shall be ignored.
- 106871 **-i** Prompt for confirmation as described previously. Any previous occurrences of the  
 106872 **-f** option shall be ignored.
- 106873 **-R** Remove file hierarchies. See the DESCRIPTION.
- 106874 **-r** Equivalent to **-R**.

#### 106875 OPERANDS

106876 The following operand shall be supported:

- 106877 *file* A pathname of a directory entry to be removed.

#### 106878 STDIN

106879 The standard input shall be used to read an input line in response to each prompt specified in  
 106880 the STDOUT section. Otherwise, the standard input shall not be used.

#### 106881 INPUT FILES

106882 None.

#### 106883 ENVIRONMENT VARIABLES

106884 The following environment variables shall affect the execution of *rm*:

- 106885 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 106886 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 106887 variables used to determine the values of locale categories.)
- 106888 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 106889 internationalization variables.
- 106890 **LC\_COLLATE**  
 106891 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 106892 character collating elements used in the extended regular expression defined for  
 106893 the **yesexpr** locale keyword in the **LC\_MESSAGES** category.
- 106894 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 106895 characters (for example, single-byte as opposed to multi-byte characters in  
 106896 arguments) and the behavior of character classes within regular expressions used  
 106897 in the extended regular expression defined for the **yesexpr** locale keyword in the  
 106898 **LC\_MESSAGES** category.
- 106899 **LC\_MESSAGES**  
 106900 Determine the locale used to process affirmative responses, and the locale used to  
 106901 affect the format and contents of diagnostic messages and prompts written to  
 106902 standard error.



|        |     |                               |                                                                                                               |
|--------|-----|-------------------------------|---------------------------------------------------------------------------------------------------------------|
| 106903 | XSI | <b>NLSPATH</b>                | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                         |
| 106904 |     | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                               |
| 106905 |     |                               | Default.                                                                                                      |
| 106906 |     | <b>STDOUT</b>                 |                                                                                                               |
| 106907 |     |                               | Not used.                                                                                                     |
| 106908 |     | <b>STDERR</b>                 |                                                                                                               |
| 106909 |     |                               | Prompts shall be written to standard error under the conditions specified in the <b>DESCRIPTION</b>           |
| 106910 |     |                               | and <b>OPTIONS</b> sections. The prompts shall contain the <i>file</i> pathname, but their format is          |
| 106911 |     |                               | otherwise unspecified. The standard error also shall be used for diagnostic messages.                         |
| 106912 |     | <b>OUTPUT FILES</b>           |                                                                                                               |
| 106913 |     |                               | None.                                                                                                         |
| 106914 |     | <b>EXTENDED DESCRIPTION</b>   |                                                                                                               |
| 106915 |     |                               | None.                                                                                                         |
| 106916 |     | <b>EXIT STATUS</b>            |                                                                                                               |
| 106917 |     |                               | The following exit values shall be returned:                                                                  |
| 106918 |     | 0                             | Each directory entry was successfully removed, unless its removal was canceled by a non-                      |
| 106919 |     |                               | affirmative response to a prompt for confirmation.                                                            |
| 106920 |     | >0                            | An error occurred.                                                                                            |
| 106921 |     | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                               |
| 106922 |     |                               | Default.                                                                                                      |
| 106923 |     | <b>APPLICATION USAGE</b>      |                                                                                                               |
| 106924 |     |                               | The <i>rm</i> utility is forbidden to remove the names <i>dot</i> and <i>dot-dot</i> in order to avoid the    |
| 106925 |     |                               | consequences of inadvertently doing something like:                                                           |
| 106926 |     | <code>rm -r .*</code>         |                                                                                                               |
| 106927 |     |                               | Some implementations do not permit the removal of the last link to an executable binary file that             |
| 106928 |     |                               | is being executed; see the [EBUSY] error in the <i>unlink()</i> function defined in the System Interfaces     |
| 106929 |     |                               | volume of POSIX.1-2008. Thus, the <i>rm</i> utility can fail to remove such files.                            |
| 106930 |     |                               | The <i>-i</i> option causes <i>rm</i> to prompt and read the standard input even if the standard input is not |
| 106931 |     |                               | a terminal, but in the absence of <i>-i</i> the mode prompting is not done when the standard input is         |
| 106932 |     |                               | not a terminal.                                                                                               |
| 106933 |     | <b>EXAMPLES</b>               |                                                                                                               |
| 106934 |     | 1.                            | The following command:                                                                                        |
| 106935 |     |                               | <code>rm a.out core</code>                                                                                    |
| 106936 |     |                               | removes the directory entries: <b>a.out</b> and <b>core</b> .                                                 |
| 106937 |     | 2.                            | The following command:                                                                                        |
| 106938 |     |                               | <code>rm -Rf junk</code>                                                                                      |
| 106939 |     |                               | removes the directory <b>junk</b> and all its contents, without prompting.                                    |
| 106940 |     | <b>RATIONALE</b>              |                                                                                                               |
| 106941 |     |                               | For absolute clarity, paragraphs (2b) and (3) in the <b>DESCRIPTION</b> of <i>rm</i> describing the behavior  |
| 106942 |     |                               | when prompting for confirmation, should be interpreted in the following manner:                               |
| 106943 |     |                               | <code>if ((NOT f_option) AND</code>                                                                           |

106944           ((not\_writable AND input\_is\_terminal) OR i\_option))

106945       The exact format of the interactive prompts is unspecified. Only the general nature of the  
106946 contents of prompts are specified because implementations may desire more descriptive  
106947 prompts than those used on historical implementations. Therefore, an application not using the  
106948 `-f` option, or using the `-i` option, relies on the system to provide the most suitable dialog directly  
106949 with the user, based on the behavior specified.

106950       The `-r` option is historical practice on all known systems. The synonym `-R` option is provided  
106951 for consistency with the other utilities in this volume of POSIX.1-2008 that provide options  
106952 requesting recursive descent through the file hierarchy.

106953       The behavior of the `-f` option in historical versions of *rm* is inconsistent. In general, along with  
106954 “forcing” the unlink without prompting for permission, it always causes diagnostic messages to  
106955 be suppressed and the exit status to be unmodified for nonexistent operands and files that  
106956 cannot be unlinked. In some versions, however, the `-f` option suppresses usage messages and  
106957 system errors as well. Suppressing such messages is not a service to either shell scripts or users.

106958       It is less clear that error messages regarding files that cannot be unlinked (removed) should be  
106959 suppressed. Although this is historical practice, this volume of POSIX.1-2008 does not permit the  
106960 `-f` option to suppress such messages.

106961       When given the `-r` and `-i` options, historical versions of *rm* prompt the user twice for each  
106962 directory, once before removing its contents and once before actually attempting to delete the  
106963 directory entry that names it. This allows the user to “prune” the file hierarchy walk. Historical  
106964 versions of *rm* were inconsistent in that some did not do the former prompt for directories  
106965 named on the command line and others had obscure prompting behavior when the `-i` option  
106966 was specified and the permissions of the file did not permit writing. The POSIX Shell and  
106967 Utilities *rm* differs little from historic practice, but does require that prompts be consistent.  
106968 Historical versions of *rm* were also inconsistent in that prompts were done to both standard  
106969 output and standard error. This volume of POSIX.1-2008 requires that prompts be done to  
106970 standard error, for consistency with *cp* and *mv*, and to allow historical extensions to *rm* that  
106971 provide an option to list deleted files on standard output.

106972       The *rm* utility is required to descend to arbitrary depths so that any file hierarchy may be  
106973 deleted. This means, for example, that the *rm* utility cannot run out of file descriptors during its  
106974 descent (that is, if the number of file descriptors is limited, *rm* cannot be implemented in the  
106975 historical fashion where one file descriptor is used per directory level). Also, *rm* is not permitted  
106976 to fail because of path length restrictions, unless an operand specified by the user is longer than  
106977 {PATH\_MAX}.

106978       The *rm* utility removes symbolic links themselves, not the files they refer to, as a consequence of  
106979 the dependence on the *unlink()* functionality, per the DESCRIPTION. When removing  
106980 hierarchies with `-r` or `-R`, the prohibition on following symbolic links has to be made explicit.

#### 106981 FUTURE DIRECTIONS

106982       None.

#### 106983 SEE ALSO

106984       *rm*  
106985       *rm*  
106986       *rm*

106985       XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

106986       XSH *remove()*, *rmdir()*, *unlink()*

106987 **CHANGE HISTORY**

106988 First released in Issue 2.

106989 **Issue 5**

106990 The FUTURE DIRECTIONS section is added.

106991 **Issue 6**106992 Text is added to clarify actions relating to symbolic links as specified in the IEEE P1003.2b draft  
106993 standard.106994 **Issue 7**

106995 Austin Group Interpretations 1003.1-2001 #019 and #091 are applied.

106996 Austin Group Interpretation 1003.1-2001 #126 is applied, changing the description of the  
106997 `LC_MESSAGES` environment variable.

106998 **NAME**106999        rmdel — remove a delta from an SCCS file (**DEVELOPMENT**)107000 **SYNOPSIS**107001 XSI        `rmdel -r SID file...`107002 **DESCRIPTION**

107003        The *rmdel* utility shall remove the delta specified by the SID from each named SCCS file. The  
 107004        delta to be removed shall be the most recent delta in its branch in the delta chain of each named  
 107005        SCCS file. In addition, the application shall ensure that the SID specified is not that of a version  
 107006        being edited for the purpose of making a delta; that is, if a *p-file* (see *get*) exists for the named  
 107007        SCCS file, the SID specified shall not appear in any entry of the *p-file*.

107008        Removal of a delta shall be restricted to:

- 107009            1. The user who made the delta
- 107010            2. The owner of the SCCS file
- 107011            3. The owner of the directory containing the SCCS file

107012 **OPTIONS**107013        The *rmdel* utility shall conform to XBD [Section 12.2](#) (on page 216).

107014        The following option shall be supported:

107015        **-r SID**        Specify the SCCS identification string (*SID*) of the delta to be deleted.107016 **OPERANDS**

107017        The following operand shall be supported:

107018        *file*            A pathname of an existing SCCS file or a directory. If *file* is a directory, the *rmdel*  
 107019        utility shall behave as though each file in the directory were specified as a named  
 107020        file, except that non-SCCS files (last component of the pathname does not begin  
 107021        with **s.**) and unreadable files shall be silently ignored.

107022        If exactly one *file* operand appears, and it is **'-'**, the standard input shall be read;  
 107023        each line of the standard input is taken to be the name of an SCCS file to be  
 107024        processed. Non-SCCS files and unreadable files shall be silently ignored.

107025 **STDIN**

107026        The standard input shall be a text file used only when the *file* operand is specified as **'-'**. Each  
 107027        line of the text file shall be interpreted as an SCCS pathname.

107028 **INPUT FILES**

107029        The SCCS files shall be files of unspecified format.

107030 **ENVIRONMENT VARIABLES**107031        The following environment variables shall affect the execution of *rmdel*:

107032        **LANG**            Provide a default value for the internationalization variables that are unset or null.  
 107033        (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 107034        variables used to determine the values of locale categories.)

107035        **LC\_ALL**           If set to a non-empty string value, override the values of all the other  
 107036        internationalization variables.

107037        **LC\_CTYPE**        Determine the locale for the interpretation of sequences of bytes of text data as  
 107038        characters (for example, single-byte as opposed to multi-byte characters in  
 107039        arguments and input files).

107040 *LC\_MESSAGES*  
107041 Determine the locale that should be used to affect the format and contents of  
107042 diagnostic messages written to standard error.

107043 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

107044 **ASYNCHRONOUS EVENTS**  
107045 Default.

107046 **STDOUT**  
107047 Not used.

107048 **STDERR**  
107049 The standard error shall be used only for diagnostic messages.

107050 **OUTPUT FILES**  
107051 The SCCS files shall be files of unspecified format. During processing of a *file*, a temporary *x-file*,  
107052 as described in *admin*, may be created and deleted; a locking *z-file*, as described in *get*, may be  
107053 created and deleted.

107054 **EXTENDED DESCRIPTION**  
107055 None.

107056 **EXIT STATUS**  
107057 The following exit values shall be returned:  
107058 0 Successful completion.  
107059 >0 An error occurred.

107060 **CONSEQUENCES OF ERRORS**  
107061 Default.

107062 **APPLICATION USAGE**  
107063 None.

107064 **EXAMPLES**  
107065 None.

107066 **RATIONALE**  
107067 None.

107068 **FUTURE DIRECTIONS**  
107069 None.

107070 **SEE ALSO**  
107071 *admin*, *delta*, *get*, *prs*  
107072 XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

107073 **CHANGE HISTORY**  
107074 First released in Issue 2.

107075 **Issue 6**  
107076 The normative text is reworded to avoid use of the term “must” for application requirements.

107077 **NAME**

107078       rmdir — remove directories

107079 **SYNOPSIS**107080       rmdir [-p] *dir*...107081 **DESCRIPTION**107082       The *rmdir* utility shall remove the directory entry specified by each *dir* operand.107083       For each *dir* operand, the *rmdir* utility shall perform actions equivalent to the *rmdir()* function  
107084       called with the *dir* operand as its only argument.107085       Directories shall be processed in the order specified. If a directory and a subdirectory of that  
107086       directory are specified in a single invocation of the *rmdir* utility, the application shall specify the  
107087       subdirectory before the parent directory so that the parent directory will be empty when the  
107088       *rmdir* utility tries to remove it.107089 **OPTIONS**107090       The *rmdir* utility shall conform to XBD [Section 12.2](#) (on page 216).

107091       The following option shall be supported:

107092       **-p**           Remove all directories in a pathname. For each *dir* operand:

- 107093                   1. The directory entry it names shall be removed.
- 
- 107094                   2. If the
- dir*
- operand includes more than one pathname component, effects
- 
- 107095                   equivalent to the following command shall occur:

107096                   rmdir -p \$(dirname *dir*)107097 **OPERANDS**

107098       The following operand shall be supported:

107099       *dir*           A pathname of an empty directory to be removed.107100 **STDIN**

107101       Not used.

107102 **INPUT FILES**

107103       None.

107104 **ENVIRONMENT VARIABLES**107105       The following environment variables shall affect the execution of *rmdir*:107106       **LANG**       Provide a default value for the internationalization variables that are unset or null.  
107107       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
107108       variables used to determine the values of locale categories.)107109       **LC\_ALL**      If set to a non-empty string value, override the values of all the other  
107110       internationalization variables.107111       **LC\_CTYPE**   Determine the locale for the interpretation of sequences of bytes of text data as  
107112       characters (for example, single-byte as opposed to multi-byte characters in  
107113       arguments).107114       **LC\_MESSAGES**107115       Determine the locale that should be used to affect the format and contents of  
107116       diagnostic messages written to standard error.

- 107117 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 107118 **ASYNCHRONOUS EVENTS**
- 107119 Default.
- 107120 **STDOUT**
- 107121 Not used.
- 107122 **STDERR**
- 107123 The standard error shall be used only for diagnostic messages.
- 107124 **OUTPUT FILES**
- 107125 None.
- 107126 **EXTENDED DESCRIPTION**
- 107127 None.
- 107128 **EXIT STATUS**
- 107129 The following exit values shall be returned:
- 107130 0 Each directory entry specified by a *dir* operand was removed successfully.
- 107131 >0 An error occurred.
- 107132 **CONSEQUENCES OF ERRORS**
- 107133 Default.
- 107134 **APPLICATION USAGE**
- 107135 The definition of an empty directory is one that contains, at most, directory entries for dot and
- 107136 dot-dot.
- 107137 **EXAMPLES**
- 107138 If a directory **a** in the current directory is empty except it contains a directory **b** and **a/b** is empty
- 107139 except it contains a directory **c**:
- 107140 `rmdir -p a/b/c`
- 107141 removes all three directories.
- 107142 **RATIONALE**
- 107143 On historical System V systems, the **-p** option also caused a message to be written to the
- 107144 standard output. The message indicated whether the whole path was removed or whether part
- 107145 of the path remained for some reason. The **STDERR** section requires this diagnostic when the
- 107146 entire path specified by a *dir* operand is not removed, but does not allow the status message
- 107147 reporting success to be written as a diagnostic.
- 107148 The *rmdir* utility on System V also included a **-s** option that suppressed the informational
- 107149 message output by the **-p** option. This option has been omitted because the informational
- 107150 message is not specified by this volume of POSIX.1-2008.
- 107151 **FUTURE DIRECTIONS**
- 107152 None.
- 107153 **SEE ALSO**
- 107154 *rm*
- 107155 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)
- 107156 XSH *remove()*, *rmdir()*, *unlink()*

107157 **CHANGE HISTORY**

107158 First released in Issue 2.

107159 **Issue 6**

107160 The normative text is reworded to avoid use of the term “must” for application requirements.



107161 **NAME**107162       sact — print current SCCS file-editing activity (**DEVELOPMENT**)107163 **SYNOPSIS**107164 XSI       sact *file...*107165 **DESCRIPTION**

107166       The *sact* utility shall inform the user of any impending deltas to a named SCCS file by writing a  
 107167       list to standard output. This situation occurs when *get -e* has been executed previously without  
 107168       a subsequent execution of *delta*, *unget*, or *sccs unedit*.

107169 **OPTIONS**

107170       None.

107171 **OPERANDS**

107172       The following operand shall be supported:

107173       *file*       A pathname of an existing SCCS file or a directory. If *file* is a directory, the *sact*  
 107174       utility shall behave as though each file in the directory were specified as a named  
 107175       file, except that non-SCCS files (last component of the pathname does not begin  
 107176       with **s**.) and unreadable files shall be silently ignored.

107177       If exactly one *file* operand appears, and it is *'-'*, the standard input shall be read;  
 107178       each line of the standard input shall be taken to be the name of an SCCS file to be  
 107179       processed. Non-SCCS files and unreadable files shall be silently ignored.

107180 **STDIN**

107181       The standard input shall be a text file used only when the *file* operand is specified as *'-'*. Each  
 107182       line of the text file shall be interpreted as an SCCS pathname.

107183 **INPUT FILES**

107184       Any SCCS files interrogated are files of an unspecified format.

107185 **ENVIRONMENT VARIABLES**107186       The following environment variables shall affect the execution of *sact*:

107187       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 107188       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 107189       variables used to determine the values of locale categories.)

107190       *LC\_ALL*     If set to a non-empty string value, override the values of all the other  
 107191       internationalization variables.

107192       *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 107193       characters (for example, single-byte as opposed to multi-byte characters in  
 107194       arguments and input files).

107195       *LC\_MESSAGES*

107196       Determine the locale that should be used to affect the format and contents of  
 107197       diagnostic messages written to standard error.

107198       *NLSPATH*    Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

107199 **ASYNCHRONOUS EVENTS**

107200       Default.

**107201 STDOUT**

107202 The output for each named file shall consist of a line in the following format:

107203 "%sΔ%sΔ%sΔ%sΔ%s\n", <SID>, <new SID>, <login>, <date>, <time>

107204 <SID> Specifies the SID of a delta that currently exists in the SCCS file to which changes  
107205 are made to make the new delta.

107206 <new SID> Specifies the SID for the new delta to be created.

107207 <login> Contains the login name of the user who makes the delta (that is, who executed a  
107208 *get* for editing).

107209 <date> Contains the date that *get -e* was executed, in the format used by the *prs :D:* data  
107210 keyword.

107211 <time> Contains the time that *get -e* was executed, in the format used by the *prs :T:* data  
107212 keyword.

107213 If there is more than one named file or if a directory or standard input is named, each pathname  
107214 shall be written before each of the preceding lines:

107215 "\n%s:\n", <pathname>

**107216 STDERR**

107217 The standard error shall be used only for optional informative messages concerning SCCS files  
107218 with no impending deltas, and for diagnostic messages.

**107219 OUTPUT FILES**

107220 None.

**107221 EXTENDED DESCRIPTION**

107222 None.

**107223 EXIT STATUS**

107224 The following exit values shall be returned:

107225 0 Successful completion.

107226 >0 An error occurred.

**107227 CONSEQUENCES OF ERRORS**

107228 Default.

**107229 APPLICATION USAGE**

107230 None.

**107231 EXAMPLES**

107232 None.

**107233 RATIONALE**

107234 None.

**107235 FUTURE DIRECTIONS**

107236 None.

**107237 SEE ALSO**

107238 *delta*, *get*, *sccs*, *unget*

107239 XBD Chapter 8 (on page 173)

107240 **CHANGE HISTORY**  
107241 First released in Issue 2.

## 107242 NAME

107243 sccs — front end for the SCCS subsystem (DEVELOPMENT)

## 107244 SYNOPSIS

107245 XSI `sccs [-r] [-d path] [-p path] command [options...] [operands...]`

## 107246 DESCRIPTION

107247 The *sccs* utility is a front end to the SCCS programs. It also includes the capability to run set-  
107248 user-id to another user to provide additional protection.107249 The *sccs* utility shall invoke the specified *command* with the specified *options* and *operands*. By  
107250 default, each of the *operands* shall be modified by prefixing it with the string "SCCS/s. ".107251 The *command* can be the name of one of the SCCS utilities in this volume of POSIX.1-2008 (*admin*,  
107252 *delta*, *get*, *prs*, *rmDEL*, *sact*, *unget*, *val*, or *what*) or one of the pseudo-utilities listed in the  
107253 EXTENDED DESCRIPTION section.

## 107254 OPTIONS

107255 The *sccs* utility shall conform to XBD [Section 12.2](#) (on page 216), except that *options* operands are  
107256 actually options to be passed to the utility named by *command*. When the portion of the  
107257 command:107258 `command [options ...] [operands ...]`107259 is considered, all of the pseudo-utilities used as *command* shall support the Utility Syntax  
107260 Guidelines. Any of the other SCCS utilities that can be invoked in this manner support the  
107261 Guidelines to the extent indicated by their individual OPTIONS sections.107262 The following options shall be supported preceding the *command* operand:107263 **-d path** A pathname of a directory to be used as a root directory for the SCCS files. The  
107264 default shall be the current directory. The **-d** option shall take precedence over the  
107265 *PROJECTDIR* variable. See **-p**.107266 **-p path** A pathname of a directory in which the SCCS files are located. The default shall be  
107267 the **SCCS** directory.107268 The **-p** option differs from the **-d** option in that the **-d** option-argument shall be  
107269 prefixed to the entire pathname and the **-p** option-argument shall be inserted  
107270 before the final component of the pathname. For example:107271 `sccs -d /x -p y get a/b`

107272 converts to:

107273 `get /x/a/y/s.b`

107274 This allows the creation of aliases such as:

107275 `alias syssccs="sccs -d /usr/src"`

107276 which is used as:

107277 `syssccs get cmd/who.c`107278 **-r** Invoke *command* with the real user ID of the process, not any effective user ID that  
107279 the *sccs* utility is set to. Certain commands (*admin*, **check**, **clean**, **diffs**, **info**, *rmDEL*,  
107280 and **tell**) cannot be run set-user-ID by all users, since this would allow anyone to  
107281 change the authorizations. These commands are always run as the real user.

**107282 OPERANDS**

107283 The following operands shall be supported:

107284 *command* An SCCS utility name or the name of one of the pseudo-utilities listed in the  
107285 EXTENDED DESCRIPTION section.

107286 *options* An option or option-argument to be passed to *command*.

107287 *operands* An operand to be passed to *command*.

**107288 STDIN**

107289 See the utility description for the specified *command*.

**107290 INPUT FILES**

107291 See the utility description for the specified *command*.

**107292 ENVIRONMENT VARIABLES**

107293 The following environment variables shall affect the execution of *sccs*:

107294 *LANG* Provide a default value for the internationalization variables that are unset or null.  
107295 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
107296 variables used to determine the values of locale categories.)

107297 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
107298 internationalization variables.

107299 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
107300 characters (for example, single-byte as opposed to multi-byte characters in  
107301 arguments and input files).

107302 *LC\_MESSAGES*

107303 Determine the locale that should be used to affect the format and contents of  
107304 diagnostic messages written to standard error.

107305 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

107306 *PROJECTDIR*

107307 Provide a default value for the *-d path* option. If the value of *PROJECTDIR* begins  
107308 with a <slash>, it shall be considered an absolute pathname; otherwise, the value  
107309 of *PROJECTDIR* is treated as a user name and that user's initial working directory  
107310 shall be examined for a subdirectory *src* or *source*. If such a directory is found, it  
107311 shall be used. Otherwise, the value shall be used as a relative pathname.

107312 Additional environment variable effects may be found in the utility description for the specified  
107313 *command*.

**107314 ASYNCHRONOUS EVENTS**

107315 Default.

**107316 STDOUT**

107317 See the utility description for the specified *command*.

**107318 STDERR**

107319 See the utility description for the specified *command*.

**107320 OUTPUT FILES**

107321 See the utility description for the specified *command*.

## 107322 EXTENDED DESCRIPTION

107323 The following pseudo-utilities shall be supported as *command* operands. All options referred to  
 107324 in the following list are values given in the *options* operands following *command*.

107325 **check** Equivalent to **info**, except that nothing shall be printed if nothing is being edited, and a  
 107326 non-zero exit status shall be returned if anything is being edited. The intent is to have  
 107327 this included in an “install” entry in a makefile to ensure that everything is included  
 107328 into the SCCS file before a version is installed.

107329 **clean** Remove everything from the current directory that can be recreated from SCCS files,  
 107330 but do not remove any files being edited. If the **-b** option is given, branches shall be  
 107331 ignored in the determination of whether they are being edited; this is dangerous if  
 107332 branches are kept in the same directory.

107333 **create** Create an SCCS file, taking the initial contents from the file of the same name. Any  
 107334 options to *admin* are accepted. If the creation is successful, the original files shall be  
 107335 renamed by prefixing the basenames with a comma. These renamed files should be  
 107336 removed after it has been verified that the SCCS files have been created successfully.

107337 **delget** Perform a *delta* on the named files and then *get* new versions. The new versions shall  
 107338 have ID keywords expanded and shall not be editable. Any **-m**, **-p**, **-r**, **-s**, and **-y**  
 107339 options shall be passed to *delta*, and any **-b**, **-c**, **-e**, **-i**, **-k**, **-l**, **-s**, and **-x** options shall be  
 107340 passed to *get*.

107341 **deledit** Equivalent to **delget**, except that the *get* phase shall include the **-e** option. This option is  
 107342 useful for making a checkpoint of the current editing phase. The same options shall be  
 107343 passed to *delta* as described above, and all the options listed for *get* above except **-e**  
 107344 shall be passed to **edit**.

107345 **diffs** Write a difference listing between the current version of the files checked out for editing  
 107346 and the versions in SCCS format. Any **-r**, **-c**, **-i**, **-x**, and **-t** options shall be passed to  
 107347 *get*; any **-l**, **-s**, **-e**, **-f**, **-h**, and **-b** options shall be passed to *diff*. A **-C** option shall be  
 107348 passed to *diff* as **-c**.

107349 **edit** Equivalent to *get -e*.

107350 **fix** Remove the named delta, but leave a copy of the delta with the changes that were in it.  
 107351 It is useful for fixing small compiler bugs, and so on. The application shall ensure that it  
 107352 is followed by a **-r** *SID* option. Since **fix** does not leave audit trails, it should be used  
 107353 carefully.

107354 **info** Write a listing of all files being edited. If the **-b** option is given, branches (that is, SIDs  
 107355 with two or fewer components) shall be ignored. If a **-u** *user* option is given, then only  
 107356 files being edited by the named user shall be listed. A **-U** option shall be equivalent to  
 107357 **-u***<current user>*.

107358 **print** Write out verbose information about the named files, equivalent to *sccs prs*.

107359 **tell** Write a <newline>-separated list of the files being edited to standard output. Takes the  
 107360 **-b**, **-u**, and **-U** options like **info** and **check**.

107361 **unedit** This is the opposite of an **edit** or a *get -e*. It should be used with caution, since any  
 107362 changes made since the *get* are lost.

107363 **EXIT STATUS**

107364 The following exit values shall be returned:

107365 0 Successful completion.

107366 &gt;0 An error occurred.

107367 **CONSEQUENCES OF ERRORS**

107368 Default.

107369 **APPLICATION USAGE**

107370 Many of the SCCS utilities take directory names as operands as well as specific filenames. The  
 107371 pseudo-utilities supported by *sccs* are not described as having this capability, but are not  
 107372 prohibited from doing so.

107373 **EXAMPLES**

107374 1. To get a file for editing, edit it and produce a new delta:

107375 `sccs get -e file.c`107376 `ex file.c`107377 `sccs delta file.c`

107378 2. To get a file from another directory:

107379 `sccs -p /usr/src/sccs/s. get cc.c`

107380 or:

107381 `sccs get /usr/src/sccs/s.cc.c`

107382 3. To make a delta of a large number of files in the current directory:

107383 `sccs delta *.c`

107384 4. To get a list of files being edited that are not on branches:

107385 `sccs info -b`

107386 5. To delta everything being edited by the current user:

107387 `sccs delta $(sccs tell -U)`

107388 6. In a makefile, to get source files from an SCCS file if it does not already exist:

107389 `SRCS = <list of source files>`107390 `$(SRCS):`107391 `sccs get $(REL) $@`107392 **RATIONALE**

107393 *sccs* and its associated utilities are part of the XSI Development Utilities option within the XSI  
 107394 option.

107395 SCCS is an abbreviation for Source Code Control System. It is a maintenance and enhancement  
 107396 tracking tool. When a file is put under SCCS, the source code control system maintains the file  
 107397 and, when changes are made, identifies and stores them in the file with the original source code  
 107398 and/or documentation. As other changes are made, they too are identified and retained in the  
 107399 file.

107400 Retrieval of the original and any set of changes is possible. Any version of the file as it develops  
 107401 can be reconstructed for inspection or additional modification. History data can be stored with  
 107402 each version, documenting why the changes were made, who made them, and when they were  
 107403 made.

107404 **FUTURE DIRECTIONS**

107405 None.

107406 **SEE ALSO**107407 *admin, delta, get, make, prs, rmdel, sact, unget, val, what*107408 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)107409 **CHANGE HISTORY**

107410 First released in Issue 4.

107411 **Issue 6**

107412 In the ENVIRONMENT VARIABLES section, the *PROJECTDIR* description is updated from  
107413 “otherwise, the home directory of a user of that name is examined” to “otherwise, the value of  
107414 *PROJECTDIR* is treated as a user name and that user’s initial working directory is examined”.

107415 The normative text is reworded to avoid use of the term “must” for application requirements.

107416 **Issue 7**

107417 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



107418 **NAME**

107419       sed — stream editor

107420 **SYNOPSIS**107421       sed [-n] *script* [*file...*]107422       sed [-n] -e *script* [-e *script*]... [-f *script\_file*]... [*file...*]107423       sed [-n] [-e *script*]... -f *script\_file* [-f *script\_file*]... [*file...*]107424 **DESCRIPTION**

107425       The *sed* utility is a stream editor that shall read one or more text files, make editing changes  
 107426       according to a script of editing commands, and write the results to standard output. The script  
 107427       shall be obtained from either the *script* operand string or a combination of the option-arguments  
 107428       from the -e *script* and -f *script\_file* options.

107429 **OPTIONS**

107430       The *sed* utility shall conform to XBD [Section 12.2](#) (on page 216), except that the order of  
 107431       presentation of the -e and -f options is significant.

107432       The following options shall be supported:

107433       -e *script*     Add the editing commands specified by the *script* option-argument to the end of  
 107434       the script of editing commands.

107435       -f *script\_file* Add the editing commands in the file *script\_file* to the end of the script of editing  
 107436       commands.

107437       -n            Suppress the default output (in which each line, after it is examined for editing, is  
 107438       written to standard output). Only lines explicitly selected for output are written.

107439       If any -e or -f options are specified, the script of editing commands shall initially be empty. The  
 107440       commands specified by each -e or -f option shall be added to the script in the order specified.  
 107441       When each addition is made, if the previous addition (if any) was from a -e option, a <newline>  
 107442       shall be inserted before the new addition. The resulting script shall have the same properties as  
 107443       the *script* operand, described in the OPERANDS section.

107444 **OPERANDS**

107445       The following operands shall be supported:

107446       *file*         A pathname of a file whose contents are read and edited. If multiple *file* operands  
 107447       are specified, the named files shall be read in the order specified and the  
 107448       concatenation shall be edited. If no *file* operands are specified, the standard input  
 107449       shall be used.

107450       *script*       A string to be used as the script of editing commands. The application shall not  
 107451       present a *script* that violates the restrictions of a text file except that the final  
 107452       character need not be a <newline>.

107453 **STDIN**

107454       The standard input shall be used if no *file* operands are specified, and shall be used if a *file*  
 107455       operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
 107456       the standard input shall not be used. See the INPUT FILES section.

107457 **INPUT FILES**

107458       The input files shall be text files. The *script\_files* named by the -f option shall consist of editing  
 107459       commands.

107460 **ENVIRONMENT VARIABLES**

107461 The following environment variables shall affect the execution of *sed*:

107462 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 107463 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 107464 variables used to determine the values of locale categories.)

107465 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 107466 internationalization variables.

107467 *LC\_COLLATE*  
 107468 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 107469 character collating elements within regular expressions.

107470 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 107471 characters (for example, single-byte as opposed to multi-byte characters in  
 107472 arguments and input files), and the behavior of character classes within regular  
 107473 expressions.

107474 *LC\_MESSAGES*  
 107475 Determine the locale that should be used to affect the format and contents of  
 107476 diagnostic messages written to standard error.

107477 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

107478 **ASYNCHRONOUS EVENTS**

107479 Default.

107480 **STDOUT**

107481 The input files shall be written to standard output, with the editing commands specified in the  
 107482 script applied. If the *-n* option is specified, only those input lines selected by the script shall be  
 107483 written to standard output.

107484 **STDERR**

107485 The standard error shall be used only for diagnostic and warning messages. |

107486 **OUTPUT FILES**

107487 The output files shall be text files whose formats are dependent on the editing commands given.

107488 **EXTENDED DESCRIPTION**

107489 The *script* shall consist of editing commands of the following form:

107490 `[address[, address]]function`

107491 where *function* represents a single-character command verb from the list in [Editing Commands](#)  
 107492 [in sed](#) (on page 3207), followed by any applicable arguments.

107493 The command can be preceded by <blank> characters and/or <semicolon> characters. The  
 107494 function can be preceded by <blank> characters. These optional characters shall have no effect.

107495 In default operation, *sed* cyclically shall append a line of input, less its terminating <newline>  
 107496 character, into the pattern space. Reading from input shall be skipped if a <newline> was in the  
 107497 pattern space prior to a **D** command ending the previous cycle. The *sed* utility shall then apply in  
 107498 sequence all commands whose addresses select that pattern space, until a command starts the  
 107499 next cycle or quits. If no commands explicitly started a new cycle, then at the end of the script  
 107500 the pattern space shall be copied to standard output (except when *-n* is specified) and the  
 107501 pattern space shall be deleted. Whenever the pattern space is written to standard output or a  
 107502 named file, *sed* shall immediately follow it with a <newline>.

107503 Some of the editing commands use a hold space to save all or part of the pattern space for

subsequent retrieval. The pattern and hold spaces shall each be able to hold at least 8 192 bytes.

#### Addresses in sed

An address is either a decimal number that counts input lines cumulatively across files, a '\$' character that addresses the last line of input, or a context address (which consists of a BRE, as described in [Regular Expressions in sed](#), preceded and followed by a delimiter, usually a <slash>).

An editing command with no addresses shall select every pattern space.

An editing command with one address shall select each pattern space that matches the address.

An editing command with two addresses shall select the inclusive range from the first pattern space that matches the first address through the next pattern space that matches the second. (If the second address is a number less than or equal to the line number first selected, only one line shall be selected.) Starting at the first line following the selected range, *sed* shall look again for the first address. Thereafter, the process shall be repeated. Omitting either or both of the address components in the following form produces undefined results:

```
[address[ , address]]
```

#### Regular Expressions in sed

The *sed* utility shall support the BREs described in XBD [Section 9.3](#) (on page 183), with the following additions:

- In a context address, the construction "`\cBREc`", where *c* is any character other than <backslash> or <newline>, shall be identical to "`/BRE/`". If the character designated by *c* appears following a <backslash>, then it shall be considered to be that literal character, which shall not terminate the BRE. For example, in the context address "`\xabc\xdefx`", the second *x* stands for itself, so that the BRE is "`abcxdef`".
- The escape sequence '`\n`' shall match a <newline> embedded in the pattern space. A literal <newline> shall not be used in the BRE of a context address or in the substitute function.
- If an RE is empty (that is, no pattern is specified) *sed* shall behave as if the last RE used in the last command applied (either as an address or as part of a substitute command) was specified.

#### Editing Commands in sed

In the following list of editing commands, the maximum number of permissible addresses for each function is indicated by [*0addr*], [*1addr*], or [*2addr*], representing zero, one, or two addresses.

The argument *text* shall consist of one or more lines. Each embedded <newline> in the text shall be preceded by a <backslash>. Other <backslash> characters in text shall be removed, and the following character shall be treated literally.

The **r** and **w** command verbs, and the *w* flag to the **s** command, take an *rfile* (or *wfile*) parameter, separated from the command verb letter or flag by one or more <blank> characters; implementations may allow zero separation as an extension.

The argument *rfile* or the argument *wfile* shall terminate the editing command. Each *wfile* shall be created before processing begins. Implementations shall support at least ten *wfile* arguments in the script; the actual number (greater than or equal to 10) that is supported by the

107546 implementation is unspecified. The use of the *wfile* parameter shall cause that file to be initially  
107547 created, if it does not exist, or shall replace the contents of an existing file.

107548 The **b**, **r**, **s**, **t**, **w**, **y**, and **:** command verbs shall accept additional arguments. The following  
107549 synopses indicate which arguments shall be separated from the command verbs by a single  
107550 <space>.

107551 The **a** and **r** commands schedule text for later output. The text specified for the **a** command, and  
107552 the contents of the file specified for the **r** command, shall be written to standard output just  
107553 before the next attempt to fetch a line of input when executing the **N** or **n** commands, or when  
107554 reaching the end of the script. If written when reaching the end of the script, and the **-n** option  
107555 was not specified, the text shall be written after copying the pattern space to standard output.  
107556 The contents of the file specified for the **r** command shall be as of the time the output is written,  
107557 not the time the **r** command is applied. The text shall be output in the order in which the **a** and **r**  
107558 commands were applied to the input.

107559 Editing commands other than {...}, **a**, **b**, **c**, **i**, **r**, **t**, **w**, **:**, and **#** can be followed by a <semicolon>, |  
107560 optional <blank> characters, and another editing command. However, when an **s** editing |  
107561 command is used with the *w* flag, following it with another command in this manner produces  
107562 undefined results.

107563 A function can be preceded by a '!' character, in which case the function shall be applied if the |  
107564 addresses do not select the pattern space. Zero or more <blank> characters shall be accepted |  
107565 before the '!' character. It is unspecified whether <blank> characters can follow the '!' |  
107566 character, and conforming applications shall not follow the '!' character with <blank> |  
107567 characters.

107568 If a *label* argument (to a **b**, **t**, or **:** command) contains characters outside of the portable filename +  
107569 character set, or if a *label* is longer than 8 bytes, the behavior is unspecified. The implementation +  
107570 shall support *label* arguments recognized as unique up to at least 8 bytes; the actual length +  
107571 (greater than or equal to 8) supported by the implementation is unspecified. It is unspecified +  
107572 whether exceeding the maximum supported label length causes an error or a silent truncation.

107573 [2addr] {editing command

107574 editing command

107575 ...

107576 } Execute a list of *sed* editing commands only when the pattern space is selected. The  
107577 list of *sed* editing commands shall be surrounded by braces. The braces can be |  
107578 preceded or followed by <blank> characters. The <right-brace> shall be preceded |  
107579 by a <newline> or <semicolon> (before any optional <blank> characters preceding |  
107580 the <right-brace>).

107581 Each command in the list of commands shall be terminated by a <newline> |  
107582 character, or by a <semicolon> character if permitted when the command is used |  
107583 outside the braces. The editing commands can be preceded by <blank> characters, |  
107584 but shall not be followed by <blank> characters.

107585 [1addr]a\

107586 text Write text to standard output as described previously.

107587 [2addr]b [label]

107588 Branch to the **:** command verb bearing the *label* argument. If *label* is not specified, |  
107589 branch to the end of the script.

107590 [2addr]c\

107591 text Delete the pattern space. With a 0 or 1 address or at the end of a 2-address range,  
107592 place *text* on the output and start the next cycle.

|        |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
|--------|----------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 107593 | [2addr] <b>d</b>                                         | Delete the pattern space and start the next cycle.                                                                                                                                                                                                                                                                                                                                                                                                                                     |
| 107594 | [2addr] <b>D</b>                                         | If the pattern space contains no <newline>, delete the pattern space and start a normal new cycle as if the <b>d</b> command was issued. Otherwise, delete the initial segment of the pattern space through the first <newline>, and start the next cycle with the resultant pattern space and without reading any new input.                                                                                                                                                          |
| 107595 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107596 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107597 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107598 | [2addr] <b>g</b>                                         |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107599 | [2addr] <b>G</b>                                         | Append to the pattern space a <newline> followed by the contents of the hold space.                                                                                                                                                                                                                                                                                                                                                                                                    |
| 107600 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107601 | [2addr] <b>h</b>                                         | Replace the contents of the hold space with the contents of the pattern space.                                                                                                                                                                                                                                                                                                                                                                                                         |
| 107602 | [2addr] <b>H</b>                                         | Append to the hold space a <newline> followed by the contents of the pattern space.                                                                                                                                                                                                                                                                                                                                                                                                    |
| 107603 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107604 | [1addr] <b>i</b> \                                       | Write <i>text</i> to standard output.                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
| 107605 | <i>text</i>                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107606 | [2addr] <b>l</b>                                         | (The letter ell.) Write the pattern space to standard output in a visually unambiguous form. The characters listed in XBD Table 5-1 (on page 121) ('\\', '\a', '\b', '\f', '\r', '\t', '\v') shall be written as the corresponding escape sequence; the '\n' in that table is not applicable. Non-printable characters not in that table shall be written as one three-digit octal number (with a preceding <backslash>) for each byte in the character (most significant byte first). |
| 107607 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107608 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107609 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107610 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107611 |                                                          | Long lines shall be folded, with the point of folding indicated by writing a <backslash> followed by a <newline>; the length at which folding occurs is unspecified, but should be appropriate for the output device. The end of each line shall be marked with a '\$'.                                                                                                                                                                                                                |
| 107612 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107613 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107614 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107615 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107616 | [2addr] <b>n</b>                                         | Write the pattern space to standard output if the default output has not been suppressed, and replace the pattern space with the next line of input, less its terminating <newline>.                                                                                                                                                                                                                                                                                                   |
| 107617 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107618 |                                                          | If no next line of input is available, the <b>n</b> command verb shall branch to the end of the script and quit without starting a new cycle.                                                                                                                                                                                                                                                                                                                                          |
| 107619 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107620 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107621 | [2addr] <b>N</b>                                         | Append the next line of input, less its terminating <newline>, to the pattern space, using an embedded <newline> to separate the appended material from the original material. Note that the current line number changes.                                                                                                                                                                                                                                                              |
| 107622 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107623 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107624 |                                                          | If no next line of input is available, the <b>N</b> command verb shall branch to the end of the script and quit without starting a new cycle or copying the pattern space to standard output.                                                                                                                                                                                                                                                                                          |
| 107625 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107626 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107627 | [2addr] <b>p</b>                                         | Write the pattern space to standard output.                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 107628 | [2addr] <b>P</b>                                         | Write the pattern space, up to the first <newline>, to standard output.                                                                                                                                                                                                                                                                                                                                                                                                                |
| 107629 | [1addr] <b>q</b>                                         | Branch to the end of the script and quit without starting a new cycle.                                                                                                                                                                                                                                                                                                                                                                                                                 |
| 107630 | [1addr] <b>r</b> <i>rfile</i>                            | Copy the contents of <i>rfile</i> to standard output as described previously. If <i>rfile</i> does not exist or cannot be read, it shall be treated as if it were an empty file, causing no error condition.                                                                                                                                                                                                                                                                           |
| 107631 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107632 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107633 | [2addr] <b>s</b> /BRE/ <i>replacement</i> / <i>flags</i> | Substitute the replacement string for instances of the BRE in the pattern space. Any character other than <backslash> or <newline> can be used instead of a <slash> to                                                                                                                                                                                                                                                                                                                 |
| 107634 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| 107635 |                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |

107636                   delimit the BRE and the replacement. Within the BRE and the replacement, the  
 107637                   BRE delimiter itself can be used as a literal character if it is preceded by a  
 107638                   <backslash>.

107639                   The replacement string shall be scanned from beginning to end. An <ampersand>  
 107640                   (' & ') appearing in the replacement shall be replaced by the string matching the  
 107641                   BRE. The special meaning of ' & ' in this context can be suppressed by preceding it  
 107642                   by a <backslash>. The characters "\n", where *n* is a digit, shall be replaced by the  
 107643                   text matched by the corresponding back-reference expression. If the corresponding  
 107644                   back-reference expression does not match, then the characters "\n" shall be  
 107645                   replaced by the empty string. The special meaning of "\n" where *n* is a digit in  
 107646                   this context, can be suppressed by preceding it by a <backslash>. For each other  
 107647                   <backslash> encountered, the following character shall lose its special meaning (if  
 107648                   any). |

107649                   A line can be split by substituting a <newline> into it. The application shall escape  
 107650                   the <newline> in the replacement by preceding it by a <backslash>.

107651                   The meaning of an unescaped <backslash> immediately followed by any character +  
 107652                   other than ' & ', <backslash>, a digit, <newline>, or the delimiter character used for +  
 107653                   this command, is unspecified. +

107654                   A substitution shall be considered to have been performed even if the replacement  
 107655                   string is identical to the string that it replaces. Any <backslash> used to alter the  
 107656                   default meaning of a subsequent character shall be discarded from the BRE or the  
 107657                   replacement before evaluating the BRE or using the replacement.

107658                   The value of *flags* shall be zero or more of:

107659                   *n*                   Substitute for the *n*th occurrence only of the BRE found within the  
 107660                   pattern space.

107661                   *g*                   Globally substitute for all non-overlapping instances of the BRE  
 107662                   rather than just the first one. If both *g* and *n* are specified, the results  
 107663                   are unspecified.

107664                   *p*                   Write the pattern space to standard output if a replacement was  
 107665                   made.

107666                   *w wfile*           Write. Append the pattern space to *wfile* if a replacement was made.  
 107667                   A conforming application shall precede the *wfile* argument with one  
 107668                   or more <blank> characters. If the *w* flag is not the last flag value  
 107669                   given in a concatenation of multiple flag values, the results are  
 107670                   undefined.

107671                   [2addr]*t [label]*  
 107672                   Test. Branch to the : command verb bearing the *label* if any substitutions have been  
 107673                   made since the most recent reading of an input line or execution of a *t*. If *label* is  
 107674                   not specified, branch to the end of the script.

107675                   [2addr]*w wfile*  
 107676                   Append (write) the pattern space to *wfile*.

107677                   [2addr]*x*           Exchange the contents of the pattern and hold spaces.

107678                   [2addr]*y/string1/string2/*  
 107679                   Replace all occurrences of characters in *string1* with the corresponding characters  
 107680                   in *string2*. If a <backslash> followed by an 'n' appear in *string1* or *string2*, the two



characters shall be handled as a single <newline>. If the number of characters in *string1* and *string2* are not equal, or if any of the characters in *string1* appear more than once, the results are undefined. Any character other than <backslash> or <newline> can be used instead of <slash> to delimit the strings. If the delimiter is not 'n', within *string1* and *string2*, the delimiter itself can be used as a literal character if it is preceded by a <backslash>. If a <backslash> character is immediately followed by a <backslash> character in *string1* or *string2*, the two <backslash> characters shall be counted as a single literal <backslash> character. The meaning of a <backslash> followed by any character that is not 'n', a <backslash>, or the delimiter character is undefined.

[0addr]:label Do nothing. This command bears a *label* to which the **b** and **t** commands branch.

[1addr]= Write the following to standard output:

"%d\n", <current line number>

[0addr] Ignore this empty command.

[0addr]# Ignore the '#' and the remainder of the line (treat them as a comment), with the single exception that if the first two characters in the script are "#n", the default output shall be suppressed; this shall be the equivalent of specifying **-n** on the command line.

## 107699 EXIT STATUS

107700 The following exit values shall be returned:

107701 0 Successful completion.

107702 >0 An error occurred.

## 107703 CONSEQUENCES OF ERRORS

107704 Default.

## 107705 APPLICATION USAGE

Regular expressions match entire strings, not just individual lines, but a <newline> is matched by '\n' in a *sed* RE; a <newline> is not allowed by the general definition of regular expression in POSIX.1-2008. Also note that '\n' cannot be used to match a <newline> at the end of an arbitrary input line; <newline> characters appear in the pattern space as a result of the **N** editing command.

When using *sed* to process pathnames, it is recommended that LC\_ALL, or at least LC\_CTYPE and LC\_COLLATE, are set to POSIX or C in the environment, since pathnames can contain byte sequences that do not form valid characters in some locales, in which case the utility's behavior would be undefined. In the POSIX locale each byte is a valid single-byte character, and therefore this problem is avoided.

## 107716 EXAMPLES

107717 This *sed* script simulates the BSD *cat -s* command, squeezing excess empty lines from standard input.

```
107719 sed -n '
107720 # Write non-empty lines.
107721 ./ {
107722     p
107723     d
107724 }
107725 # Write a single empty line, then look for more empty lines.
```

```

107726 /^$/      p
107727 # Get next line, discard the held <newline> (empty line),
107728 # and look for more empty lines.
107729 :Empty
107730 /^$/      {
107731     N
107732     s/.//
107733     b Empty
107734 }
107735 # Write the non-empty line before going back to search
107736 # for the first in a set of empty lines.
107737 p
107738 '

```

107739 The following *sed* command is a much simpler method of squeezing empty lines, although it is  
 107740 not quite the same as *cat -s* since it removes any initial empty lines:

```

107741 sed -n '/./,/^$/p'

```

#### 107742 RATIONALE

107743 This volume of POSIX.1-2008 requires implementations to support at least ten distinct *wfiles*,  
 107744 matching historical practice on many implementations. Implementations are encouraged to  
 107745 support more, but conforming applications should not exceed this limit.

107746 The exit status codes specified here are different from those in System V. System V returns 2 for  
 107747 garbled *sed* commands, but returns zero with its usage message or if the input file could not be  
 107748 opened. The standard developers considered this to be a bug.

107749 The manner in which the *l* command writes non-printable characters was changed to avoid the  
 107750 historical backspace-overstrike method, and other requirements to achieve unambiguous output  
 107751 were added. See the RATIONALE for *ed* for details of the format chosen, which is the same as  
 107752 that chosen for *sed*.

107753 This volume of POSIX.1-2008 requires implementations to provide pattern and hold spaces of at  
 107754 least 8 192 bytes, larger than the 4 000 bytes spaces used by some historical implementations, but  
 107755 less than the 20 480 bytes limit used in an early proposal. Implementations are encouraged to  
 107756 allocate dynamically larger pattern and hold spaces as needed.

107757 The requirements for acceptance of <blank> and <space> characters in command lines has been  
 107758 made more explicit than in early proposals to describe clearly the historical practice and to  
 107759 remove confusion about the phrase “protect initial blanks [*sic*] and tabs from the stripping that is  
 107760 done on every script line” that appears in much of the historical documentation of the *sed* utility  
 107761 description of text. (Not all implementations are known to have stripped <blank> characters  
 107762 from text lines, although they all have allowed leading <blank> characters preceding the address  
 107763 on a command line.)

107764 The treatment of ‘#’ comments differs from the SVID which only allows a comment as the first  
 107765 line of the script, but matches BSD-derived implementations. The comment character is treated  
 107766 as a command, and it has the same properties in terms of being accepted with leading <blank>  
 107767 characters; the BSD implementation has historically supported this.

107768 Early proposals required that a *script\_file* have at least one non-comment line. Some historical  
 107769 implementations have behaved in unexpected ways if this were not the case. The standard  
 107770 developers considered that this was incorrect behavior and that application developers should  
 107771 not have to avoid this feature. A correct implementation of this volume of POSIX.1-2008 shall  
 107772 permit *script\_files* that consist only of comment lines.



Early proposals indicated that if **-e** and **-f** options were intermixed, all **-e** options were processed before any **-f** options. This has been changed to process them in the order presented because it matches historical practice and is more intuitive.

The treatment of the **p** flag to the **s** command differs between System V and BSD-based systems when the default output is suppressed. In the two examples:

```
echo a | sed 's/a/A/p'
echo a | sed -n 's/a/A/p'
```

this volume of POSIX.1-2008, BSD, System V documentation, and the SVID indicate that the first example should write two lines with **A**, whereas the second should write one. Some System V systems write the **A** only once in both examples because the **p** flag is ignored if the **-n** option is not specified.

This is a case of a diametrical difference between systems that could not be reconciled through the compromise of declaring the behavior to be unspecified. The SVID/BSD/System V documentation behavior was adopted for this volume of POSIX.1-2008 because:

- No known documentation for any historic system describes the interaction between the **p** flag and the **-n** option.
- The selected behavior is more correct as there is no technical justification for any interaction between the **p** flag and the **-n** option. A relationship between **-n** and the **p** flag might imply that they are only used together, but this ignores valid scripts that interrupt the cyclical nature of the processing through the use of the **D**, **d**, **q**, or branching commands. Such scripts rely on the **p** suffix to write the pattern space because they do not make use of the default output at the “bottom” of the script.
- Because the **-n** option makes the **p** flag unnecessary, any interaction would only be useful if *sed* scripts were written to run both with and without the **-n** option. This is believed to be unlikely. It is even more unlikely that programmers have coded the **p** flag expecting it to be unnecessary. Because the interaction was not documented, the likelihood of a programmer discovering the interaction and depending on it is further decreased.
- Finally, scripts that break under the specified behavior produce too much output instead of too little, which is easier to diagnose and correct.

The form of the substitute command that uses the **n** suffix was limited to the first 512 matches in an early proposal. This limit has been removed because there is no reason an editor processing lines of {LINE\_MAX} length should have this restriction. The command **s/a/A/2047** should be able to substitute the 2047th occurrence of **a** on a line.

The **b**, **t**, and **:** commands are documented to ignore leading white space, but no mention is made of trailing white space. Historical implementations of *sed* assigned different locations to the labels **'x'** and **"x "**. This is not useful, and leads to subtle programming errors, but it is historical practice, and changing it could theoretically break working scripts. Implementors are encouraged to provide warning messages about labels that are never referenced by a **b** or **t** command, jumps to labels that do not exist, and label arguments that are subject to truncation.

Earlier versions of this standard allowed for implementations with bytes other than eight bits, but this has been modified in this version.

## FUTURE DIRECTIONS

None.

107816 **SEE ALSO**107817 *awk, ed, grep*

107818 XBD Table 5-1 (on page 121), Chapter 8 (on page 173), Section 9.3 (on page 183), Section 12.2 (on  
107819 page 216)

107820 **CHANGE HISTORY**

107821 First released in Issue 2.

107822 **Issue 5**

107823 The FUTURE DIRECTIONS section is added.

107824 **Issue 6**

107825 The following new requirements on POSIX implementations derive from alignment with the  
107826 Single UNIX Specification:

- 107827 • Implementations are required to support at least ten *wfile* arguments in an editing  
107828 command.

107829 The EXTENDED DESCRIPTION is changed to align with the IEEE P1003.2b draft standard.

107830 IEEE PASC Interpretation 1003.2 #190 is applied.

107831 IEEE PASC Interpretation 1003.2 #203 is applied, clarifying the meaning of the  
107832 <backslash>-escape sequences in a replacement string for a BRE.

107833 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/28 is applied, removing text describing  
107834 behavior on systems with bytes consisting of more than eight bits.

107835 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/29 is applied, making an editorial  
107836 correction within the Editing Commands in *sed* section.

107837 **Issue 7**

107838 Austin Group Interpretations 1003.1-2001 #006, #036, and #092 are applied.

107839 SD5-XCU-ERN-97 and SD5-XCU-ERN-123 are applied, updating the SYNOPSIS.

107840 A second example is added.

107841 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0133 [262], XCU/TC1-2008/0134  
107842 [282,431], XCU/TC1-2008/0135 [269], and XCU/TC1-2008/0136 [282,431] are applied.

107843 **NAME**107844 `sh` — shell, the standard command language interpreter107845 **SYNOPSIS**107846 `sh` [`-abCefhimnuvx`] [`-o option`]... [`+abCefhimnuvx`] [`+o option`]...  
107847 [`command_file` [`argument...`]]107848 `sh -c` [`-abCefhimnuvx`] [`-o option`]... [`+abCefhimnuvx`] [`+o option`]...  
107849 `command_string` [`command_name` [`argument...`]]107850 `sh -s` [`-abCefhimnuvx`] [`-o option`]... [`+abCefhimnuvx`] [`+o option`]...  
107851 [`argument...`]107852 **DESCRIPTION**107853 The `sh` utility is a command language interpreter that shall execute commands read from a  
107854 command line string, the standard input, or a specified file. The application shall ensure that the  
107855 commands to be executed are expressed in the language described in [Chapter 2](#) (on page 2335).

107856 Pathname expansion shall not fail due to the size of a file.

107857 Shell input and output redirections have an implementation-defined offset maximum that is  
107858 established in the open file description.107859 **OPTIONS**107860 The `sh` utility shall conform to XBD [Section 12.2](#) (on page 216), with an extension for support of a  
107861 leading <plus-sign> ('+') as noted below.107862 The `-a`, `-b`, `-C`, `-e`, `-f`, `-m`, `-n`, `-o option`, `-u`, `-v`, and `-x` options are described as part of the `set`  
107863 utility in [Section 2.14](#) (on page 2374). The option letters derived from the `set` special built-in shall  
107864 also be accepted with a leading <plus-sign> ('+') instead of a leading <hyphen-minus>  
107865 (meaning the reverse case of the option as described in this volume of POSIX.1-2008).

107866 The following additional options shall be supported:

107867 `-c` Read commands from the `command_string` operand. Set the value of special  
107868 parameter 0 (see [Section 2.5.2](#), on page 2340) from the value of the `command_name`  
107869 operand and the positional parameters (\$1, \$2, and so on) in sequence from the  
107870 remaining `argument` operands. No commands shall be read from the standard  
107871 input.107872 `-i` Specify that the shell is *interactive*; see below. An implementation may treat  
107873 specifying the `-i` option as an error if the real user ID of the calling process does  
107874 not equal the effective user ID or if the real group ID does not equal the effective  
107875 group ID.107876 `-s` Read commands from the standard input.107877 If there are no operands and the `-c` option is not specified, the `-s` option shall be assumed.107878 If the `-i` option is present, or if there are no operands and the shell's standard input and  
107879 standard error are attached to a terminal, the shell is considered to be *interactive*.107880 **OPERANDS**

107881 The following operands shall be supported:

107882 `-` A single <hyphen-minus> shall be treated as the first operand and then ignored. If  
107883 both `'-'` and `"--"` are given as arguments, or if other operands precede the single  
107884 <hyphen-minus>, the results are undefined.

107885 *argument* The positional parameters (\$1, \$2, and so on) shall be set to *arguments*, if any.

107886 *command\_file* The pathname of a file containing commands. If the pathname contains one or  
 107887 more <slash> characters, the implementation attempts to read that file; the file  
 107888 need not be executable. If the pathname does not contain a <slash> character:

- The implementation shall attempt to read that file from the current working  
 107889 directory; the file need not be executable.
- If the file is not in the current working directory, the implementation may  
 107890 perform a search for an executable file using the value of *PATH*, as described  
 107891 in [Section 2.9.1.1](#) (on page 2357).

107892  
 107893  
 107894 Special parameter 0 (see [Section 2.5.2](#), on page 2340) shall be set to the value of  
 107895 *command\_file*. If *sh* is called using a synopsis form that omits *command\_file*, special  
 107896 parameter 0 shall be set to the value of the first argument passed to *sh* from its  
 107897 parent (for example, *argv*[0] for a C program), which is normally a pathname used  
 107898 to execute the *sh* utility.

107899 *command\_name*  
 107900 A string assigned to special parameter 0 when executing the commands in  
 107901 *command\_string*. If *command\_name* is not specified, special parameter 0 shall be set  
 107902 to the value of the first argument passed to *sh* from its parent (for example, *argv*[0]  
 107903 for a C program), which is normally a pathname used to execute the *sh* utility.

107904 *command\_string*  
 107905 A string that shall be interpreted by the shell as one or more commands, as if the  
 107906 string were the argument to the *system()* function defined in the System Interfaces  
 107907 volume of POSIX.1-2008. If the *command\_string* operand is an empty string, *sh* shall  
 107908 exit with a zero exit status.

107909 **STDIN**

107910 The standard input shall be used only if one of the following is true:

- The *-s* option is specified.
- The *-c* option is not specified and no operands are specified.
- The script executes one or more commands that require input from standard input (such as  
 107913 a *read* command that does not redirect its input).

107914  
 107915 See the INPUT FILES section.

107916 When the shell is using standard input and it invokes a command that also uses standard input,  
 107917 the shell shall ensure that the standard input file pointer points directly after the command it has  
 107918 read when the command begins execution. It shall not read ahead in such a manner that any  
 107919 characters intended to be read by the invoked command are consumed by the shell (whether  
 107920 interpreted by the shell or not) or that characters that are not read by the invoked command are  
 107921 not seen by the shell. When the command expecting to read standard input is started  
 107922 asynchronously by an interactive shell, it is unspecified whether characters are read by the  
 107923 command or interpreted by the shell.

107924 If the standard input to *sh* is a FIFO or terminal device and is set to non-blocking reads, then *sh*  
 107925 shall enable blocking reads on standard input. This shall remain in effect when the command  
 107926 completes.

107927 **INPUT FILES**

107928 The input file shall be a text file, except that line lengths shall be unlimited. If the input file  
 107929 consists solely of zero or more blank lines and comments, *sh* shall exit with a zero exit status.

107930 **ENVIRONMENT VARIABLES**

107931 The following environment variables shall affect the execution of *sh*:

107932 UP **ENV** This variable, when and only when an interactive shell is invoked, shall be  
 107933 subjected to parameter expansion (see [Section 2.6.2](#), on page 2344) by the shell, and  
 107934 the resulting value shall be used as a pathname of a file containing shell  
 107935 commands to execute in the current environment. The file need not be executable.  
 107936 If the expanded value of *ENV* is not an absolute pathname, the results are  
 107937 unspecified. *ENV* shall be ignored if the real and effective user IDs or real and  
 107938 effective group IDs of the process are different.

107939 UP **FCEDIT** This variable, when expanded by the shell, shall determine the default value for  
 107940 the *-e editor* option's *editor* option-argument. If *FCEDIT* is null or unset, *ed* shall be  
 107941 used as the editor.

107942 UP **HISTFILE** Determine a pathname naming a command history file. If the *HISTFILE* variable is  
 107943 not set, the shell may attempt to access or create a file *.sh\_history* in the directory  
 107944 referred to by the *HOME* environment variable. If the shell cannot obtain both read  
 107945 and write access to, or create, the history file, it shall use an unspecified  
 107946 mechanism that allows the history to operate properly. (References to history  
 107947 "file" in this section shall be understood to mean this unspecified mechanism in  
 107948 such cases.) An implementation may choose to access this variable only when  
 107949 initializing the history file; this initialization shall occur when *fc* or *sh* first attempt  
 107950 to retrieve entries from, or add entries to, the file, as the result of commands issued  
 107951 by the user, the file named by the *ENV* variable, or implementation-defined system  
 107952 start-up files. Implementations may choose to disable the history list mechanism  
 107953 for users with appropriate privileges who do not set *HISTFILE*; the specific  
 107954 circumstances under which this occurs are implementation-defined. If more than  
 107955 one instance of the shell is using the same history file, it is unspecified how  
 107956 updates to the history file from those shells interact. As entries are deleted from the  
 107957 history file, they shall be deleted oldest first. It is unspecified when history file  
 107958 entries are physically removed from the history file.

107959 **HISTSIZE** Determine a decimal number representing the limit to the number of previous  
 107960 commands that are accessible. If this variable is unset, an unspecified default  
 107961 greater than or equal to 128 shall be used. The maximum number of commands in  
 107962 the history list is unspecified, but shall be at least 128. An implementation may  
 107963 choose to access this variable only when initializing the history file, as described  
 107964 under *HISTFILE*. Therefore, it is unspecified whether changes made to *HISTSIZE*  
 107965 after the history file has been initialized are effective.

107966 UP **HOME** Determine the pathname of the user's home directory. The contents of *HOME* are  
 107967 used in tilde expansion as described in [Section 2.6.1](#) (on page 2344).

107968 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 107969 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 107970 variables used to determine the values of locale categories.)

107971 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 107972 internationalization variables.

|        |     |                                                                                                                                |                                                                                                   |
|--------|-----|--------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------|
| 107973 |     | <b>LC_COLLATE</b>                                                                                                              |                                                                                                   |
| 107974 |     | Determine the behavior of range expressions, equivalence classes, and multi-                                                   |                                                                                                   |
| 107975 |     | character collating elements within pattern matching.                                                                          |                                                                                                   |
| 107976 |     | <b>LC_CTYPE</b>                                                                                                                | Determine the locale for the interpretation of sequences of bytes of text data as                 |
| 107977 |     | characters (for example, single-byte as opposed to multi-byte characters in                                                    |                                                                                                   |
| 107978 |     | arguments and input files), which characters are defined as letters (character class                                           |                                                                                                   |
| 107979 |     | <b>alpha</b> ), and the behavior of character classes within pattern matching.                                                 |                                                                                                   |
| 107980 |     | <b>LC_MESSAGES</b>                                                                                                             |                                                                                                   |
| 107981 |     | Determine the locale that should be used to affect the format and contents of                                                  |                                                                                                   |
| 107982 |     | diagnostic messages written to standard error.                                                                                 |                                                                                                   |
| 107983 | UP  | <b>MAIL</b>                                                                                                                    | Determine a pathname of the user's mailbox file for purposes of incoming mail                     |
| 107984 |     | notification. If this variable is set, the shell shall inform the user if the file named                                       |                                                                                                   |
| 107985 |     | by the variable is created or if its modification time has changed. Informing the                                              |                                                                                                   |
| 107986 |     | user shall be accomplished by writing a string of unspecified format to standard                                               |                                                                                                   |
| 107987 |     | error prior to the writing of the next primary prompt string. Such check shall be                                              |                                                                                                   |
| 107988 |     | performed only after the completion of the interval defined by the <b>MAILCHECK</b>                                            |                                                                                                   |
| 107989 |     | variable after the last such check. The user shall be informed only if <b>MAIL</b> is set                                      |                                                                                                   |
| 107990 |     | and <b>MAILPATH</b> is not set.                                                                                                |                                                                                                   |
| 107991 | UP  | <b>MAILCHECK</b>                                                                                                               |                                                                                                   |
| 107992 |     | Establish a decimal integer value that specifies how often (in seconds) the shell                                              |                                                                                                   |
| 107993 |     | shall check for the arrival of mail in the files specified by the <b>MAILPATH</b> or <b>MAIL</b>                               |                                                                                                   |
| 107994 |     | variables. The default value shall be 600 seconds. If set to zero, the shell shall check                                       |                                                                                                   |
| 107995 |     | before issuing each primary prompt.                                                                                            |                                                                                                   |
| 107996 | UP  | <b>MAILPATH</b>                                                                                                                | Provide a list of pathnames and optional messages separated by <colon>                            |
| 107997 |     | characters. If this variable is set, the shell shall inform the user if any of the files                                       |                                                                                                   |
| 107998 |     | named by the variable are created or if any of their modification times change. (See                                           |                                                                                                   |
| 107999 |     | the preceding entry for <b>MAIL</b> for descriptions of mail arrival and user informing.)                                      |                                                                                                   |
| 108000 |     | Each pathname can be followed by ' % ' and a string that shall be subjected to                                                 |                                                                                                   |
| 108001 |     | parameter expansion and written to standard error when the modification time                                                   |                                                                                                   |
| 108002 |     | changes. If a ' % ' character in the pathname is preceded by a <backslash>, it shall                                           |                                                                                                   |
| 108003 |     | be treated as a literal ' % ' in the pathname. The default message is unspecified.                                             |                                                                                                   |
| 108004 |     | The <b>MAILPATH</b> environment variable takes precedence over the <b>MAIL</b> variable.                                       |                                                                                                   |
| 108005 | XSI | <b>NLSPATH</b>                                                                                                                 | Determine the location of message catalogs for the processing of <b>LC_MESSAGES</b> .             |
| 108006 |     | <b>PATH</b>                                                                                                                    | Establish a string formatted as described in XBD <a href="#">Chapter 8</a> (on page 173), used to |
| 108007 |     | effect command interpretation; see <a href="#">Section 2.9.1.1</a> (on page 2357).                                             |                                                                                                   |
| 108008 |     | <b>PWD</b>                                                                                                                     | This variable shall represent an absolute pathname of the current working                         |
| 108009 |     | directory. Assignments to this variable may be ignored.                                                                        |                                                                                                   |
| 108010 |     | <b>ASYNCHRONOUS EVENTS</b>                                                                                                     |                                                                                                   |
| 108011 |     | The <b>sh</b> utility shall take the standard action for all signals (see <a href="#">Section 1.4</a> , on page 2326) with the |                                                                                                   |
| 108012 |     | following exceptions.                                                                                                          |                                                                                                   |
| 108013 |     | If the shell is interactive, SIGINT signals received during command line editing shall be handled                              |                                                                                                   |
| 108014 |     | as described in the EXTENDED DESCRIPTION, and SIGINT signals received at other times                                           |                                                                                                   |
| 108015 |     | shall be caught but no action performed.                                                                                       |                                                                                                   |
| 108016 |     | If the shell is interactive:                                                                                                   |                                                                                                   |



- 108017 • SIGQUIT and SIGTERM signals shall be ignored.
  - 108018 • If the `-m` option is in effect, SIGTTIN, SIGTTOU, and SIGTSTP signals shall be ignored.
  - 108019 • If the `-m` option is not in effect, it is unspecified whether SIGTTIN, SIGTTOU, and
  - 108020 SIGTSTP signals are ignored, set to the default action, or caught. If they are caught, the
  - 108021 shell shall, in the signal-catching function, set the signal to the default action and raise the
  - 108022 signal (after taking any appropriate steps, such as restoring terminal settings).
- 108023 The standard actions, and the actions described above for interactive shells, can be overridden
- 108024 by use of the *trap* special built-in utility (see *trap* (on page 2410) and [Section 2.11](#), on page 2371).

## 108025 STDOUT

108026 See the STDERR section.

## 108027 STDERR

108028 Except as otherwise stated (by the descriptions of any invoked utilities or in interactive mode),

108029 standard error shall be used only for diagnostic messages.

## 108030 OUTPUT FILES

108031 None.

## 108032 EXTENDED DESCRIPTION

108033 UP See [Chapter 2](#). The functionality described in the rest of the EXTENDED DESCRIPTION section

108034 shall be provided on implementations that support the User Portability Utilities option (and the

108035 rest of this section is not further shaded for this option).

## 108036 Command History List

108037 When the *sh* utility is being used interactively, it shall maintain a list of commands previously

108038 entered from the terminal in the file named by the *HISTFILE* environment variable. The type,

108039 size, and internal format of this file are unspecified. Multiple *sh* processes can share access to the

108040 file for a user, if file access permissions allow this; see the description of the *HISTFILE*

108041 environment variable.

## 108042 Command Line Editing

108043 When *sh* is being used interactively from a terminal, the current command and the command

108044 history (see *fc*) can be edited using *vi*-mode command line editing. This mode uses commands,

108045 described below, similar to a subset of those described in the *vi* utility. Implementations may

108046 offer other command line editing modes corresponding to other editing utilities.

108047 The command `set -o vi` shall enable *vi*-mode editing and place *sh* into *vi* insert mode (see

108048 [Command Line Editing \(vi-mode\)](#), on page 3220). This command also shall disable any other

108049 editing mode that the implementation may provide. The command `set +o vi` disables *vi*-mode

108050 editing.

108051 Certain block-mode terminals may be unable to support shell command line editing. If a

108052 terminal is unable to provide either edit mode, it need not be possible to `set -o vi` when using the

108053 shell on this terminal.

108054 In the following sections, the characters *erase*, *interrupt*, *kill*, and *end-of-file* are those set by the

108055 *stty* utility.

## Command Line Editing (vi-mode)

In *vi* editing mode, there shall be a distinguished line, the edit line. All the editing operations which modify a line affect the edit line. The edit line is always the newest line in the command history buffer.

With *vi*-mode enabled, *sh* can be switched between insert mode and command mode.

When in insert mode, an entered character shall be inserted into the command line, except as noted in [vi Line Editing Insert Mode](#). Upon entering *sh* and after termination of the previous command, *sh* shall be in insert mode.

Typing an escape character shall switch *sh* into command mode (see [vi Line Editing Command Mode](#), on page 3221). In command mode, an entered character shall either invoke a defined operation, be used as part of a multi-character operation, or be treated as an error. A character that is not recognized as part of an editing command shall terminate any specific editing command and shall alert the terminal. If *sh* receives a SIGINT signal in command mode (whether generated by typing the *interrupt* character or by other means), it shall terminate command line editing on the current command line, reissue the prompt on the next line of the terminal, and reset the command history (see *fc*) so that the most recently executed command is the previous command (that is, the command that was being edited when it was interrupted is not re-entered into the history).

In the following sections, the phrase “move the cursor to the beginning of the word” shall mean “move the cursor to the first character of the current word” and the phrase “move the cursor to the end of the word” shall mean “move the cursor to the last character of the current word”. The phrase “beginning of the command line” indicates the point between the end of the prompt string issued by the shell (or the beginning of the terminal line, if there is no prompt string) and the first character of the command text.

## vi Line Editing Insert Mode

While in insert mode, any character typed shall be inserted in the current command line, unless it is from the following set.

**<newline>** Execute the current command line. If the current command line is not empty, this line shall be entered into the command history (see *fc*).

**erase** Delete the character previous to the current cursor position and move the current cursor position back one character. In insert mode, characters shall be erased from both the screen and the buffer when backspacing.

**interrupt** If *sh* receives a SIGINT signal in insert mode (whether generated by typing the *interrupt* character or by other means), it shall terminate command line editing with the same effects as described for interrupting command mode; see [Command Line Editing \(vi-mode\)](#).

**kill** Clear all the characters from the input line.

**<control>-V** Insert the next character input, even if the character is otherwise a special insert mode character.

**<control>-W** Delete the characters from the one preceding the cursor to the preceding word boundary. The word boundary in this case is the closer to the cursor of either the beginning of the line or a character that is in neither the **blank** nor **punct** character classification of the current locale.



108099 *end-of-file* Interpreted as the end of input in *sh*. This interpretation shall occur only at the  
 108100 beginning of an input line. If *end-of-file* is entered other than at the beginning of the  
 108101 line, the results are unspecified.

108102 <ESC> Place *sh* into command mode.

### 108103 **vi Line Editing Command Mode**

108104 In command mode for the command line editing feature, decimal digits not beginning with 0  
 108105 that precede a command letter shall be remembered. Some commands use these decimal digits  
 108106 as a count number that affects the operation.

108107 The term *motion command* represents one of the commands:

108108 <space> 0 b F l W ^ \$ ; E f T w | , B e h t

108109 If the current line is not the edit line, any command that modifies the current line shall cause the  
 108110 content of the current line to replace the content of the edit line, and the current line shall  
 108111 become the edit line. This replacement cannot be undone (see the **u** and **U** commands below).  
 108112 The modification requested shall then be performed to the edit line. When the current line is the  
 108113 edit line, the modification shall be done directly to the edit line.

108114 Any command that is preceded by *count* shall take a count (the numeric value of any preceding  
 108115 decimal digits). Unless otherwise noted, this count shall cause the specified operation to repeat  
 108116 by the number of times specified by the count. Also unless otherwise noted, a *count* that is out  
 108117 of range is considered an error condition and shall alert the terminal, but neither the cursor  
 108118 position, nor the command line, shall change.

108119 The terms *word* and *bigword* are used as defined in the *vi* description. The term *save buffer*  
 108120 corresponds to the term *unnamed buffer* in *vi*.

108121 The following commands shall be recognized in command mode:

108122 <newline> Execute the current command line. If the current command line is not empty, this  
 108123 line shall be entered into the command history (see *fc*).

108124 <control>-L Redraw the current command line. Position the cursor at the same location on the  
 108125 redrawn line.

108126 # Insert the character '#' at the beginning of the current command line and treat the  
 108127 resulting edit line as a comment. This line shall be entered into the command  
 108128 history; see *fc*.

108129 = Display the possible shell word expansions (see [Section 2.6](#), on page 2343) of the  
 108130 bigword at the current command line position.

108131 **Note:** This does not modify the content of the current line, and therefore does not cause  
 108132 the current line to become the edit line.

108133 These expansions shall be displayed on subsequent terminal lines. If the bigword  
 108134 contains none of the characters '?', '\*', or '[', an <asterisk> ('\*') shall be  
 108135 implicitly assumed at the end. If any directories are matched, these expansions  
 108136 shall have a '/' character appended. After the expansion, the line shall be  
 108137 redrawn, the cursor repositioned at the current cursor position, and *sh* shall be  
 108138 placed in command mode.

108139 \ Perform pathname expansion (see [Section 2.6.6](#), on page 2350) on the current  
 108140 bigword, up to the largest set of characters that can be matched uniquely. If the  
 108141 bigword contains none of the characters '?', '\*', or '[', an <asterisk> ('\*')  
 108142 shall be implicitly assumed at the end. This maximal expansion then shall replace

|        |                |                                                                                                        |
|--------|----------------|--------------------------------------------------------------------------------------------------------|
| 108143 |                | the original bigword in the command line, and the cursor shall be placed after this                    |
| 108144 |                | expansion. If the resulting bigword completely and uniquely matches a directory, a                     |
| 108145 |                | '/' character shall be inserted directly after the bigword. If some other file is                      |
| 108146 |                | completely matched, a single <space> shall be inserted after the bigword. After                        |
| 108147 |                | this operation, <i>sh</i> shall be placed in insert mode.                                              |
| 108148 | *              | Perform pathname expansion on the current bigword and insert all expansions                            |
| 108149 |                | into the command to replace the current bigword, with each expansion separated                         |
| 108150 |                | by a single <space>. If at the end of the line, the current cursor position shall be                   |
| 108151 |                | moved to the first column position following the expansions and <i>sh</i> shall be placed              |
| 108152 |                | in insert mode. Otherwise, the current cursor position shall be the last column                        |
| 108153 |                | position of the first character after the expansions and <i>sh</i> shall be placed in insert           |
| 108154 |                | mode. If the current bigword contains none of the characters '?', '*', or '[',                         |
| 108155 |                | before the operation, an <asterisk> ('*') shall be implicitly assumed at the end.                      |
| 108156 | @letter        | Insert the value of the alias named <i>_letter</i> . The symbol <i>letter</i> represents a single      |
| 108157 |                | alphabetic character from the portable character set; implementations may support                      |
| 108158 |                | additional characters as an extension. If the alias <i>_letter</i> contains other editing              |
| 108159 |                | commands, these commands shall be performed as part of the insertion. If no alias                      |
| 108160 |                | <i>_letter</i> is enabled, this command shall have no effect.                                          |
| 108161 | [count]~       | Convert, if the current character is a lowercase letter, to the equivalent uppercase                   |
| 108162 |                | letter and <i>vice versa</i> , as prescribed by the current locale. The current cursor position        |
| 108163 |                | then shall be advanced by one character. If the cursor was positioned on the last                      |
| 108164 |                | character of the line, the case conversion shall occur, but the cursor shall not                       |
| 108165 |                | advance. If the '~' command is preceded by a <i>count</i> , that number of characters                  |
| 108166 |                | shall be converted, and the cursor shall be advanced to the character position after                   |
| 108167 |                | the last character converted. If the <i>count</i> is larger than the number of characters              |
| 108168 |                | after the cursor, this shall not be considered an error; the cursor shall advance to                   |
| 108169 |                | the last character on the line.                                                                        |
| 108170 | [count].       | Repeat the most recent non-motion command, even if it was executed on an earlier                       |
| 108171 |                | command line. If the previous command was preceded by a <i>count</i> , and no count is                 |
| 108172 |                | given on the '.' command, the count from the previous command shall be                                 |
| 108173 |                | included as part of the repeated command. If the '.' command is preceded by a                          |
| 108174 |                | <i>count</i> , this shall override any <i>count</i> argument to the previous command. The <i>count</i> |
| 108175 |                | specified in the '.' command shall become the count for subsequent '.'                                 |
| 108176 |                | commands issued without a count.                                                                       |
| 108177 | [number]v      | Invoke the <i>vi</i> editor to edit the current command line in a temporary file. When the             |
| 108178 |                | editor exits, the commands in the temporary file shall be executed and placed in                       |
| 108179 |                | the command history. If a <i>number</i> is included, it specifies the command number in                |
| 108180 |                | the command history to be edited, rather than the current command line.                                |
| 108181 | [count]l (ell) |                                                                                                        |
| 108182 | [count]<space> |                                                                                                        |
| 108183 |                | Move the current cursor position to the next character position. If the cursor was                     |
| 108184 |                | positioned on the last character of the line, the terminal shall be alerted and the                    |
| 108185 |                | cursor shall not be advanced. If the <i>count</i> is larger than the number of characters              |
| 108186 |                | after the cursor, this shall not be considered an error; the cursor shall advance to                   |
| 108187 |                | the last character on the line.                                                                        |
| 108188 | [count]h       | Move the current cursor position to the <i>count</i> th (default 1) previous character                 |
| 108189 |                | position. If the cursor was positioned on the first character of the line, the terminal                |
| 108190 |                | shall be alerted and the cursor shall not be moved. If the count is larger than the                    |

|        |                  |                                                                                               |
|--------|------------------|-----------------------------------------------------------------------------------------------|
| 108191 |                  | number of characters before the cursor, this shall not be considered an error; the            |
| 108192 |                  | cursor shall move to the first character on the line.                                         |
| 108193 | <b>[count]w</b>  | Move to the start of the next word. If the cursor was positioned on the last                  |
| 108194 |                  | character of the line, the terminal shall be alerted and the cursor shall not be              |
| 108195 |                  | advanced. If the <i>count</i> is larger than the number of words after the cursor, this shall |
| 108196 |                  | not be considered an error; the cursor shall advance to the last character on the             |
| 108197 |                  | line.                                                                                         |
| 108198 | <b>[count]W</b>  | Move to the start of the next bigword. If the cursor was positioned on the last               |
| 108199 |                  | character of the line, the terminal shall be alerted and the cursor shall not be              |
| 108200 |                  | advanced. If the <i>count</i> is larger than the number of bigwords after the cursor, this    |
| 108201 |                  | shall not be considered an error; the cursor shall advance to the last character on           |
| 108202 |                  | the line.                                                                                     |
| 108203 | <b>[count]e</b>  | Move to the end of the current word. If at the end of a word, move to the end of              |
| 108204 |                  | the next word. If the cursor was positioned on the last character of the line, the            |
| 108205 |                  | terminal shall be alerted and the cursor shall not be advanced. If the <i>count</i> is larger |
| 108206 |                  | than the number of words after the cursor, this shall not be considered an error; the         |
| 108207 |                  | cursor shall advance to the last character on the line.                                       |
| 108208 | <b>[count]E</b>  | Move to the end of the current bigword. If at the end of a bigword, move to the               |
| 108209 |                  | end of the next bigword. If the cursor was positioned on the last character of the            |
| 108210 |                  | line, the terminal shall be alerted and the cursor shall not be advanced. If the <i>count</i> |
| 108211 |                  | is larger than the number of bigwords after the cursor, this shall not be considered          |
| 108212 |                  | an error; the cursor shall advance to the last character on the line.                         |
| 108213 | <b>[count]b</b>  | Move to the beginning of the current word. If at the beginning of a word, move to             |
| 108214 |                  | the beginning of the previous word. If the cursor was positioned on the first                 |
| 108215 |                  | character of the line, the terminal shall be alerted and the cursor shall not be              |
| 108216 |                  | moved. If the <i>count</i> is larger than the number of words preceding the cursor, this      |
| 108217 |                  | shall not be considered an error; the cursor shall return to the first character on the       |
| 108218 |                  | line.                                                                                         |
| 108219 | <b>[count]B</b>  | Move to the beginning of the current bigword. If at the beginning of a bigword,               |
| 108220 |                  | move to the beginning of the previous bigword. If the cursor was positioned on the            |
| 108221 |                  | first character of the line, the terminal shall be alerted and the cursor shall not be        |
| 108222 |                  | moved. If the <i>count</i> is larger than the number of bigwords preceding the cursor,        |
| 108223 |                  | this shall not be considered an error; the cursor shall return to the first character on      |
| 108224 |                  | the line.                                                                                     |
| 108225 | <b>^</b>         | Move the current cursor position to the first character on the input line that is not a       |
| 108226 |                  | <blank>.                                                                                      |
| 108227 | <b>\$</b>        | Move to the last character position on the current command line.                              |
| 108228 | <b>0</b>         | (Zero.) Move to the first character position on the current command line.                     |
| 108229 | <b>[count] l</b> | Move to the <i>count</i> th character position on the current command line. If no number      |
| 108230 |                  | is specified, move to the first position. The first character position shall be               |
| 108231 |                  | numbered 1. If the count is larger than the number of characters on the line, this            |
| 108232 |                  | shall not be considered an error; the cursor shall be placed on the last character on         |
| 108233 |                  | the line.                                                                                     |
| 108234 | <b>[count]fc</b> | Move to the first occurrence of the character 'c' that occurs after the current               |
| 108235 |                  | cursor position. If the cursor was positioned on the last character of the line, the          |
| 108236 |                  | terminal shall be alerted and the cursor shall not be advanced. If the character 'c'          |

|        |                       |                                                                                                          |
|--------|-----------------------|----------------------------------------------------------------------------------------------------------|
| 108237 |                       | does not occur in the line after the current cursor position, the terminal shall be                      |
| 108238 |                       | alerted and the cursor shall not be moved.                                                               |
| 108239 | <b>[count]Fc</b>      | Move to the first occurrence of the character 'c' that occurs before the current                         |
| 108240 |                       | cursor position. If the cursor was positioned on the first character of the line, the                    |
| 108241 |                       | terminal shall be alerted and the cursor shall not be moved. If the character 'c'                        |
| 108242 |                       | does not occur in the line before the current cursor position, the terminal shall be                     |
| 108243 |                       | alerted and the cursor shall not be moved.                                                               |
| 108244 | <b>[count]tc</b>      | Move to the character before the first occurrence of the character 'c' that occurs                       |
| 108245 |                       | after the current cursor position. If the cursor was positioned on the last character                    |
| 108246 |                       | of the line, the terminal shall be alerted and the cursor shall not be advanced. If the                  |
| 108247 |                       | character 'c' does not occur in the line after the current cursor position, the                          |
| 108248 |                       | terminal shall be alerted and the cursor shall not be moved.                                             |
| 108249 | <b>[count]Tc</b>      | Move to the character after the first occurrence of the character 'c' that occurs                        |
| 108250 |                       | before the current cursor position. If the cursor was positioned on the first                            |
| 108251 |                       | character of the line, the terminal shall be alerted and the cursor shall not be                         |
| 108252 |                       | moved. If the character 'c' does not occur in the line before the current cursor                         |
| 108253 |                       | position, the terminal shall be alerted and the cursor shall not be moved.                               |
| 108254 | <b>[count];</b>       | Repeat the most recent <b>f</b> , <b>F</b> , <b>t</b> , or <b>T</b> command. Any number argument on that |
| 108255 |                       | previous command shall be ignored. Errors are those described for the repeated                           |
| 108256 |                       | command.                                                                                                 |
| 108257 | <b>[count],</b>       | Repeat the most recent <b>f</b> , <b>F</b> , <b>t</b> , or <b>T</b> command. Any number argument on that |
| 108258 |                       | previous command shall be ignored. However, reverse the direction of that                                |
| 108259 |                       | command.                                                                                                 |
| 108260 | <b>a</b>              | Enter insert mode after the current cursor position. Characters that are entered                         |
| 108261 |                       | shall be inserted before the next character.                                                             |
| 108262 | <b>A</b>              | Enter insert mode after the end of the current command line.                                             |
| 108263 | <b>i</b>              | Enter insert mode at the current cursor position. Characters that are entered shall                      |
| 108264 |                       | be inserted before the current character.                                                                |
| 108265 | <b>I</b>              | Enter insert mode at the beginning of the current command line.                                          |
| 108266 | <b>R</b>              | Enter insert mode, replacing characters from the command line beginning at the                           |
| 108267 |                       | current cursor position.                                                                                 |
| 108268 | <b>[count]cmotion</b> |                                                                                                          |
| 108269 |                       | Delete the characters between the current cursor position and the cursor position                        |
| 108270 |                       | that would result from the specified motion command. Then enter insert mode                              |
| 108271 |                       | before the first character following any deleted characters. If <i>count</i> is specified, it            |
| 108272 |                       | shall be applied to the motion command. A <i>count</i> shall be ignored for the following                |
| 108273 |                       | motion commands:                                                                                         |
| 108274 |                       | 0      ^      \$      c                                                                                  |
| 108275 |                       | If the motion command is the character 'c', the current command line shall be                            |
| 108276 |                       | cleared and insert mode shall be entered. If the motion command would move the                           |
| 108277 |                       | current cursor position toward the beginning of the command line, the character                          |
| 108278 |                       | under the current cursor position shall not be deleted. If the motion command                            |
| 108279 |                       | would move the current cursor position toward the end of the command line, the                           |
| 108280 |                       | character under the current cursor position shall be deleted. If the <i>count</i> is larger              |
| 108281 |                       | than the number of characters between the current cursor position and the end of                         |

|        |                       |                                                                                                   |
|--------|-----------------------|---------------------------------------------------------------------------------------------------|
| 108282 |                       | the command line toward which the motion command would move the cursor, this                      |
| 108283 |                       | shall not be considered an error; all of the remaining characters in the                          |
| 108284 |                       | aforementioned range shall be deleted and insert mode shall be entered. If the                    |
| 108285 |                       | motion command is invalid, the terminal shall be alerted, the cursor shall not be                 |
| 108286 |                       | moved, and no text shall be deleted.                                                              |
| 108287 | <b>C</b>              | Delete from the current character to the end of the line and enter insert mode at the             |
| 108288 |                       | new end-of-line.                                                                                  |
| 108289 | <b>S</b>              | Clear the entire edit line and enter insert mode.                                                 |
| 108290 | <b>[count]rc</b>      | Replace the current character with the character 'c'. With a number <i>count</i> ,                |
| 108291 |                       | replace the current and the following <i>count</i> –1 characters. After this command, the         |
| 108292 |                       | current cursor position shall be on the last character that was changed. If the <i>count</i>      |
| 108293 |                       | is larger than the number of characters after the cursor, this shall not be considered            |
| 108294 |                       | an error; all of the remaining characters shall be changed.                                       |
| 108295 | <b>[count]_</b>       | Append a <space> after the current character position and then append the last                    |
| 108296 |                       | bigword in the previous input line after the <space>. Then enter insert mode after                |
| 108297 |                       | the last character just appended. With a number <i>count</i> , append the <i>count</i> th bigword |
| 108298 |                       | in the previous line.                                                                             |
| 108299 | <b>[count]x</b>       | Delete the character at the current cursor position and place the deleted characters              |
| 108300 |                       | in the save buffer. If the cursor was positioned on the last character of the line, the           |
| 108301 |                       | character shall be deleted and the cursor position shall be moved to the previous                 |
| 108302 |                       | character (the new last character). If the <i>count</i> is larger than the number of              |
| 108303 |                       | characters after the cursor, this shall not be considered an error; all the characters            |
| 108304 |                       | from the cursor to the end of the line shall be deleted.                                          |
| 108305 | <b>[count]X</b>       | Delete the character before the current cursor position and place the deleted                     |
| 108306 |                       | characters in the save buffer. The character under the current cursor position shall              |
| 108307 |                       | not change. If the cursor was positioned on the first character of the line, the                  |
| 108308 |                       | terminal shall be alerted, and the X command shall have no effect. If the line                    |
| 108309 |                       | contained a single character, the X command shall have no effect. If the line                     |
| 108310 |                       | contained no characters, the terminal shall be alerted and the cursor shall not be                |
| 108311 |                       | moved. If the <i>count</i> is larger than the number of characters before the cursor, this        |
| 108312 |                       | shall not be considered an error; all the characters from before the cursor to the                |
| 108313 |                       | beginning of the line shall be deleted.                                                           |
| 108314 | <b>[count]dmotion</b> |                                                                                                   |
| 108315 |                       | Delete the characters between the current cursor position and the character                       |
| 108316 |                       | position that would result from the motion command. A number <i>count</i> repeats the             |
| 108317 |                       | motion command <i>count</i> times. If the motion command would move toward the                    |
| 108318 |                       | beginning of the command line, the character under the current cursor position                    |
| 108319 |                       | shall not be deleted. If the motion command is <b>d</b> , the entire current command line         |
| 108320 |                       | shall be cleared. If the <i>count</i> is larger than the number of characters between the         |
| 108321 |                       | current cursor position and the end of the command line toward which the motion                   |
| 108322 |                       | command would move the cursor, this shall not be considered an error; all of the                  |
| 108323 |                       | remaining characters in the aforementioned range shall be deleted. The deleted                    |
| 108324 |                       | characters shall be placed in the save buffer.                                                    |
| 108325 | <b>D</b>              | Delete all characters from the current cursor position to the end of the line. The                |
| 108326 |                       | deleted characters shall be placed in the save buffer.                                            |

|        |                                |                                                                                                 |
|--------|--------------------------------|-------------------------------------------------------------------------------------------------|
| 108327 | <b>[count]ymotion</b>          |                                                                                                 |
| 108328 |                                | Yank (that is, copy) the characters from the current cursor position to the position            |
| 108329 |                                | resulting from the motion command into the save buffer. A number <i>count</i> shall be          |
| 108330 |                                | applied to the motion command. If the motion command would move toward the                      |
| 108331 |                                | beginning of the command line, the character under the current cursor position                  |
| 108332 |                                | shall not be included in the set of yanked characters. If the motion command is <b>y</b> ,      |
| 108333 |                                | the entire current command line shall be yanked into the save buffer. The current               |
| 108334 |                                | cursor position shall be unchanged. If the <i>count</i> is larger than the number of            |
| 108335 |                                | characters between the current cursor position and the end of the command line                  |
| 108336 |                                | toward which the motion command would move the cursor, this shall not be                        |
| 108337 |                                | considered an error; all of the remaining characters in the aforementioned range                |
| 108338 |                                | shall be yanked.                                                                                |
| 108339 | <b>Y</b>                       | Yank the characters from the current cursor position to the end of the line into the            |
| 108340 |                                | save buffer. The current character position shall be unchanged.                                 |
| 108341 | <b>[count]p</b>                | Put a copy of the current contents of the save buffer after the current cursor                  |
| 108342 |                                | position. The current cursor position shall be advanced to the last character put               |
| 108343 |                                | from the save buffer. A <i>count</i> shall indicate how many copies of the save buffer          |
| 108344 |                                | shall be put.                                                                                   |
| 108345 | <b>[count]P</b>                | Put a copy of the current contents of the save buffer before the current cursor                 |
| 108346 |                                | position. The current cursor position shall be moved to the last character put from             |
| 108347 |                                | the save buffer. A <i>count</i> shall indicate how many copies of the save buffer shall be      |
| 108348 |                                | put.                                                                                            |
| 108349 | <b>u</b>                       | Undo the last command that changed the edit line. This operation shall not undo                 |
| 108350 |                                | the copy of any command line to the edit line.                                                  |
| 108351 | <b>U</b>                       | Undo all changes made to the edit line. This operation shall not undo the copy of               |
| 108352 |                                | any command line to the edit line.                                                              |
| 108353 | <b>[count]k</b>                |                                                                                                 |
| 108354 | <b>[count]–</b>                | Set the current command line to be the <i>count</i> th previous command line in the shell       |
| 108355 |                                | command history. If <i>count</i> is not specified, it shall default to 1. The cursor shall be   |
| 108356 |                                | positioned on the first character of the new command. If a <b>k</b> or <b>–</b> command would   |
| 108357 |                                | retreat past the maximum number of commands in effect for this shell (affected by               |
| 108358 |                                | the <i>HISTSIZE</i> environment variable), the terminal shall be alerted, and the               |
| 108359 |                                | command shall have no effect.                                                                   |
| 108360 | <b>[count]j</b>                |                                                                                                 |
| 108361 | <b>[count]+</b>                | Set the current command line to be the <i>count</i> th next command line in the shell           |
| 108362 |                                | command history. If <i>count</i> is not specified, it shall default to 1. The cursor shall be   |
| 108363 |                                | positioned on the first character of the new command. If a <b>j</b> or <b>+</b> command         |
| 108364 |                                | advances past the edit line, the current command line shall be restored to the edit             |
| 108365 |                                | line and the terminal shall be alerted.                                                         |
| 108366 | <b>[number]G</b>               | Set the current command line to be the oldest command line stored in the shell                  |
| 108367 |                                | command history. With a number <i>number</i> , set the current command line to be the           |
| 108368 |                                | command line <i>number</i> in the history. If command line <i>number</i> does not exist, the    |
| 108369 |                                | terminal shall be alerted and the command line shall not be changed.                            |
| 108370 | <b>/pattern&lt;newline&gt;</b> |                                                                                                 |
| 108371 |                                | Move backwards through the command history, searching for the specified                         |
| 108372 |                                | pattern, beginning with the previous command line. Patterns use the pattern                     |
| 108373 |                                | matching notation described in <a href="#">Section 2.13</a> (on page 2372), except that the ‘^’ |



108374 character shall have special meaning when it appears as the first character of  
 108375 *pattern*. In this case, the '^' is discarded and the characters after the '^' shall be  
 108376 matched only at the beginning of a line. Commands in the command history shall  
 108377 be treated as strings, not as filenames. If the pattern is not found, the current  
 108378 command line shall be unchanged and the terminal shall be alerted. If it is found in  
 108379 a previous line, the current command line shall be set to that line and the cursor  
 108380 shall be set to the first character of the new command line.

108381 If *pattern* is empty, the last non-empty pattern provided to / or ? shall be used. If  
 108382 there is no previous non-empty pattern, the terminal shall be alerted and the  
 108383 current command line shall remain unchanged.

108384 ?*pattern*<newline>

108385 Move forwards through the command history, searching for the specified pattern,  
 108386 beginning with the next command line. Patterns use the pattern matching notation  
 108387 described in [Section 2.13](#) (on page 2372), except that the '^' character shall have  
 108388 special meaning when it appears as the first character of *pattern*. In this case, the  
 108389 '^' is discarded and the characters after the '^' shall be matched only at the  
 108390 beginning of a line. Commands in the command history shall be treated as strings,  
 108391 not as filenames. If the pattern is not found, the current command line shall be  
 108392 unchanged and the terminal shall be alerted. If it is found in a following line, the  
 108393 current command line shall be set to that line and the cursor shall be set to the first  
 108394 character of the new command line.

108395 If *pattern* is empty, the last non-empty pattern provided to / or ? shall be used. If  
 108396 there is no previous non-empty pattern, the terminal shall be alerted and the  
 108397 current command line shall remain unchanged.

108398 **n** Repeat the most recent / or ? command. If there is no previous / or ?, the terminal  
 108399 shall be alerted and the current command line shall remain unchanged.

108400 **N** Repeat the most recent / or ? command, reversing the direction of the search. If  
 108401 there is no previous / or ?, the terminal shall be alerted and the current command  
 108402 line shall remain unchanged.

## 108403 EXIT STATUS

108404 The following exit values shall be returned:

108405 0 The script to be executed consisted solely of zero or more blank lines or comments, or  
 108406 both.

108407 1-125 A non-interactive shell detected an error other than *command\_file* not found or  
 108408 executable, including but not limited to syntax, redirection, or variable assignment  
 108409 errors.

108410 126 A specified *command\_file* could not be executed due to an [ENOEXEC] error (see [Section](#)  
 108411 [2.9.1.1](#) (on page 2357), item 2).

108412 127 A specified *command\_file* could not be found by a non-interactive shell.

108413 Otherwise, the shell shall return the exit status of the last command it invoked or attempted to  
 108414 invoke (see also the *exit* utility in [Section 2.14](#), on page 2374).

## 108415 CONSEQUENCES OF ERRORS

108416 See [Section 2.8.1](#) (on page 2353).

## APPLICATION USAGE

Standard input and standard error are the files that determine whether a shell is interactive when `-i` is not specified. For example:

```
sh > file
```

and:

```
sh 2> file
```

create interactive and non-interactive shells, respectively. Although both accept terminal input, the results of error conditions are different, as described in [Section 2.8.1](#) (on page 2353); in the second example a redirection error encountered by a special built-in utility aborts the shell.

A conforming application must protect its first operand, if it starts with a `<plus-sign>`, by preceding it with the `--` argument that denotes the end of the options.

Applications should note that the standard *PATH* to the shell cannot be assumed to be either `/bin/sh` or `/usr/bin/sh`, and should be determined by interrogation of the *PATH* returned by *getconf PATH*, ensuring that the returned pathname is an absolute pathname and not a shell built-in.

For example, to determine the location of the standard *sh* utility:

```
command -v sh
```

On some implementations this might return:

```
/usr/xpg4/bin/sh
```

Furthermore, on systems that support executable scripts (the `"#!"` construct), it is recommended that applications using executable scripts install them using *getconf PATH* to determine the shell pathname and update the `"#!"` script appropriately as it is being installed (for example, with *sed*). For example:

```
#
# Installation time script to install correct POSIX shell pathname
#
# Get list of paths to check
#
Sifs=$IFS
Sifs_set=${IFS+y}
IFS=:
set -- $(getconf PATH)
if [ "$Sifs_set" = y ]
then
    IFS=$Sifs
else
    unset IFS
fi
#
# Check each path for 'sh'
#
for i
do
    if [ -x "${i}"/sh ]
    then
        Pshell=${i}/sh
```



```

108463         fi
108464     done
108465     #
108466     # This is the list of scripts to update. They should be of the
108467     # form '${name}.source' and will be transformed to '${name}'.
108468     # Each script should begin:
108469     #
108470     # #!INSTALLSHELLPATH
108471     #
108472     scripts="a b c"
108473     #
108474     # Transform each script
108475     #
108476     for i in ${scripts}
108477     do
108478         sed -e "s|INSTALLSHELLPATH|${Pshell}|" < ${i}.source > ${i}
108479     done

```

#### 108480 EXAMPLES

- 108481 1. Execute a shell command from a string:
- 108482 `sh -c "cat myfile"`
- 108483 2. Execute a shell script from a file in the current directory:
- 108484 `sh my_shell_cmds`

#### 108485 RATIONALE

108486 The *sh* utility and the *set* special built-in utility share a common set of options.

108487 The name *IFS* was originally an abbreviation of “Input Field Separators”; however, this name is  
 108488 misleading as the *IFS* characters are actually used as field terminators. One justification for  
 108489 ignoring the contents of *IFS* upon entry to the script, beyond security considerations, is to assist  
 108490 possible future shell compilers. Allowing *IFS* to be imported from the environment prevents  
 108491 many optimizations that might otherwise be performed via dataflow analysis of the script itself.

108492 The text in the STDIN section about non-blocking reads concerns an instance of *sh* that has been  
 108493 invoked, probably by a C-language program, with standard input that has been opened using  
 108494 the `O_NONBLOCK` flag; see `open()` in the System Interfaces volume of POSIX.1-2008. If the shell  
 108495 did not reset this flag, it would immediately terminate because no input data would be available  
 108496 yet and that would be considered the same as end-of-file.

108497 The options associated with a *restricted shell* (command name *rsh* and the `-r` option) were  
 108498 excluded because the standard developers considered that the implied level of security could  
 108499 not be achieved and they did not want to raise false expectations.

108500 On systems that support set-user-ID scripts, a historical trapdoor has been to link a script to the  
 108501 name `-i`. When it is called by a sequence such as:

108502 `sh -`

108503 or by:

108504 `#! usr/bin/sh -`

108505 the historical systems have assumed that no option letters follow. Thus, this volume of  
 108506 POSIX.1-2008 allows the single `<hyphen-minus>` to mark the end of the options, in addition to  
 108507 the use of the regular `--` argument, because it was considered that the older practice was so

pervasive. An alternative approach is taken by the KornShell, where real and effective user/group IDs must match for an interactive shell; this behavior is specifically allowed by this volume of POSIX.1-2008.

**Note:** There are other problems with set-user-ID scripts that the two approaches described here do not resolve.

The initialization process for the history file can be dependent on the system start-up files, in that they may contain commands that effectively preempt the user's settings of *HISTFILE* and *HISTSIZE*. For example, function definition commands are recorded in the history file, unless the *set -o nolog* option is set. If the system administrator includes function definitions in some system start-up file called before the *ENV* file, the history file is initialized before the user gets a chance to influence its characteristics. In some historical shells, the history file is initialized just after the *ENV* file has been processed. Therefore, it is implementation-defined whether changes made to *HISTFILE* after the history file has been initialized are effective.

The default messages for the various *MAIL*-related messages are unspecified because they vary across implementations. Typical messages are:

```
"you have mail\n"
```

or:

```
"you have new mail\n"
```

It is important that the descriptions of command line editing refer to the same shell as that in POSIX.1-2008 so that interactive users can also be application programmers without having to deal with programmatic differences in their two environments. It is also essential that the utility name *sh* be specified because this explicit utility name is too firmly rooted in historical practice of application programs for it to change.

Consideration was given to mandating a diagnostic message when attempting to set *vi*-mode on terminals that do not support command line editing. However, it is not historical practice for the shell to be cognizant of all terminal types and thus be able to detect inappropriate terminals in all cases. Implementations are encouraged to supply diagnostics in this case whenever possible, rather than leaving the user in a state where editing commands work incorrectly.

In early proposals, the KornShell-derived *emacs* mode of command line editing was included, even though the *emacs* editor itself was not. The community of *emacs* proponents was adamant that the full *emacs* editor not be standardized because they were concerned that an attempt to standardize this very powerful environment would encourage vendors to ship strictly conforming versions lacking the extensibility required by the community. The author of the original *emacs* program also expressed his desire to omit the program. Furthermore, there were a number of historical systems that did not include *emacs*, or included it without supporting it, but there were very few that did not include and support *vi*. The shell *emacs* command line editing mode was finally omitted because it became apparent that the KornShell version and the editor being distributed with the GNU system had diverged in some respects. The author of *emacs* requested that the POSIX *emacs* mode either be deleted or have a significant number of unspecified conditions. Although the KornShell author agreed to consider changes to bring the shell into alignment, the standard developers decided to defer specification at that time. At the time, it was assumed that convergence on an acceptable definition would occur for a subsequent draft, but that has not happened, and there appears to be no impetus to do so. In any case, implementations are free to offer additional command line editing modes based on the exact models of editors their users are most comfortable with.

Early proposals had the following list entry in [vi Line Editing Insert Mode](#) (on page 3220):

108554 \ If followed by the *erase* or *kill* character, that character shall be inserted into the input line.  
 108555 Otherwise, the <backslash> itself shall be inserted into the input line.

108556 However, this is not actually a feature of *sh* command line editing insert mode, but one of some  
 108557 historical terminal line drivers. Some conforming implementations continue to do this when the  
 108558 *stty ixten* flag is set.

108559 In interactive shells, SIGTERM is ignored so that *kill 0* does not kill the shell, and SIGINT is  
 108560 caught so that *wait* is interruptible. If the shell does not ignore SIGTTIN, SIGTTOU, and  
 108561 SIGTSTP signals when it is interactive and the *-m* option is not in effect, these signals suspend  
 108562 the shell if it is not a session leader. If it is a session leader, the signals are discarded if they  
 108563 would stop the process, as required by XSH Section 2.4.3 (on page 490) for orphaned process  
 108564 groups.

#### 108565 FUTURE DIRECTIONS

108566 None.

#### 108567 SEE ALSO

108568 Section 2.9.1.1 (on page 2357), Chapter 2 (on page 2335), *cd*, *echo*, *exit*, *fc*, *pwd*, *invalid*, *set*, *stty*,  
 108569 *test*, *trap*, *umask*, *vi*

108570 XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

108571 XSH *dup()*, *exec*, *exit()*, *fork()*, *open()*, *pipe()*, *signal()*, *system()*, *ulimit()*, *umask()*, *wait()*

#### 108572 CHANGE HISTORY

108573 First released in Issue 2.

#### 108574 Issue 5

108575 The FUTURE DIRECTIONS section is added.

108576 Text is added to the DESCRIPTION for the Large File Summit proposal.

#### 108577 Issue 6

108578 The Open Group Corrigendum U029/2 is applied, correcting the second SYNOPSIS.

108579 The Open Group Corrigendum U027/3 is applied, correcting a typographical error.

108580 The following new requirements on POSIX implementations derive from alignment with the  
 108581 Single UNIX Specification:

- 108582 • The option letters derived from the *set* special built-in are also accepted with a leading  
 108583 <plus-sign> ('+').

- 108584 • Large file extensions are added:

- 108585 — Pathname expansion does not fail due to the size of a file.

- 108586 — Shell input and output redirections have an implementation-defined offset maximum  
 108587 that is established in the open file description.

108588 In the ENVIRONMENT VARIABLES section, the text “user’s home directory” is updated to  
 108589 “directory referred to by the *HOME* environment variable”.

108590 Descriptions for the *ENV* and *PWD* environment variables are included to align with the  
 108591 IEEE P1003.2b draft standard.

108592 The normative text is reworded to avoid use of the term “must” for application requirements.

108593 **Issue 7**

108594 Austin Group Interpretation 1003.1-2001 #098 is applied, changing the definition of *IFS*.

108595 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

108596 Changes to the *pwd* utility and *PWD* environment variable have been made to match the  
108597 changes to the *getcwd()* function made for Austin Group Interpretation 1003.1-2001 #140.

108598 Minor editorial changes are made to the User Portability Utilities option shading. No normative  
108599 changes are implied.

108600 Minor changes are made to the install script example in the APPLICATION USAGE section.

108601 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0137 [152], XCU/TC1-2008/0138  
108602 [347], XCU/TC1-2008/0139 [347], XCU/TC1-2008/0140 [347], XCU/TC1-2008/0141 [299], and  
108603 XCU/TC1-2008/0142 [347] are applied.

108604 **NAME**

108605           sleep — suspend execution for an interval

108606 **SYNOPSIS**108607           sleep *time*108608 **DESCRIPTION**

108609           The *sleep* utility shall suspend execution for at least the integral number of seconds specified by

108610           the *time* operand.

108611 **OPTIONS**

108612           None.

108613 **OPERANDS**

108614           The following operand shall be supported:

108615           *time*           A non-negative decimal integer specifying the number of seconds for which to

108616                           suspend execution.

108617 **STDIN**

108618           Not used.

108619 **INPUT FILES**

108620           None.

108621 **ENVIRONMENT VARIABLES**108622           The following environment variables shall affect the execution of *sleep*:

108623           *LANG*           Provide a default value for the internationalization variables that are unset or null.

108624                           (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization

108625                           variables used to determine the values of locale categories.)

108626           *LC\_ALL*       If set to a non-empty string value, override the values of all the other

108627                           internationalization variables.

108628           *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as

108629                           characters (for example, single-byte as opposed to multi-byte characters in

108630                           arguments).

108631           *LC\_MESSAGES*

108632                           Determine the locale that should be used to affect the format and contents of

108633                           diagnostic messages written to standard error.

108634 XSI       *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

108635 **ASYNCHRONOUS EVENTS**108636           If the *sleep* utility receives a SIGALRM signal, one of the following actions shall be taken:

- 108637           1. Terminate normally with a zero exit status.
- 108638           2. Effectively ignore the signal.
- 108639           3. Provide the default behavior for signals described in the ASYNCHRONOUS EVENTS
- 108640               section of [Section 1.4](#) (on page 2326). This could include terminating with a non-zero exit
- 108641               status.

108642           The *sleep* utility shall take the standard action for all other signals.

**108643 STDOUT**

108644 Not used.

**108645 STDERR**

108646 The standard error shall be used only for diagnostic messages.

**108647 OUTPUT FILES**

108648 None.

**108649 EXTENDED DESCRIPTION**

108650 None.

**108651 EXIT STATUS**

108652 The following exit values shall be returned:

108653 0 The execution was successfully suspended for at least *time* seconds, or a SIGALRM signal  
108654 was received. See the ASYNCHRONOUS EVENTS section.

108655 >0 An error occurred.

**108656 CONSEQUENCES OF ERRORS**

108657 Default.

**108658 APPLICATION USAGE**

108659 None.

**108660 EXAMPLES**

108661 The *sleep* utility can be used to execute a command after a certain amount of time, as in:

108662 (*sleep 105; command*) &

108663 or to execute a command every so often, as in:

```
108664 while true  
108665 do  
108666     command  
108667     sleep 37  
108668 done
```

**108669 RATIONALE**

108670 The exit status is allowed to be zero when *sleep* is interrupted by the SIGALRM signal because  
108671 most implementations of this utility rely on the arrival of that signal to notify them that the  
108672 requested finishing time has been successfully attained. Such implementations thus do not  
108673 distinguish this situation from the successful completion case. Other implementations are  
108674 allowed to catch the signal and go back to sleep until the requested time expires or to provide  
108675 the normal signal termination procedures.

108676 As with all other utilities that take integral operands and do not specify subranges of allowed  
108677 values, *sleep* is required by this volume of POSIX.1-2008 to deal with *time* requests of up to  
108678 2 147 483 647 seconds. This may mean that some implementations have to make multiple calls to  
108679 the delay mechanism of the underlying operating system if its argument range is less than this.

**108680 FUTURE DIRECTIONS**

108681 None.

**108682 SEE ALSO**

108683 *wait*

108684 XBD Chapter 8 (on page 173)

108685 XSH *alarm()*, *sleep()*

|        |                            |
|--------|----------------------------|
| 108686 | <b>CHANGE HISTORY</b>      |
| 108687 | First released in Issue 2. |

108688 **NAME**108689 `sort` — sort, merge, or sequence check text files108690 **SYNOPSIS**108691 `sort [-m] [-o output] [-bdfinru] [-t char] [-k keydef]... [file...]`108692 `sort [-c|-C] [-bdfinru] [-t char] [-k keydef] [file]`108693 **DESCRIPTION**108694 The `sort` utility shall perform one of the following functions:

- 108695 1. Sort lines of all the named files together and write the result to the specified output.
- 108696 2. Merge lines of all the named (presorted) files together and write the result to the specified output.
- 108697 3. Check that a single input file is correctly presorted.

108699 Comparisons shall be based on one or more sort keys extracted from each line of input (or, if no  
 108700 sort keys are specified, the entire line up to, but not including, the terminating <newline>), and  
 108701 shall be performed using the collating sequence of the current locale. If this collating sequence  
 108702 does not have a total ordering of all characters (see XBD [Section 7.3.2](#), on page 147), any lines of  
 108703 input that collate equally should be further compared byte-by-byte using the collating sequence  
 108704 for the POSIX locale.

108705 **OPTIONS**

108706 The `sort` utility shall conform to XBD [Section 12.2](#) (on page 216), except for Guideline 9, and the  
 108707 `-k keydef` option should follow the `-b`, `-d`, `-f`, `-i`, `-n`, and `-r` options. In addition, `'+'` may be  
 108708 recognized as an option delimiter as well as `'-'`.

108709 The following options shall be supported:

- 108710 `-c` Check that the single input file is ordered as specified by the arguments and the  
 108711 collating sequence of the current locale. Output shall not be sent to standard  
 108712 output. The exit code shall indicate whether or not disorder was detected or an  
 108713 error occurred. If disorder (or, with `-u`, a duplicate key) is detected, a warning  
 108714 message shall be sent to standard error indicating where the disorder or duplicate  
 108715 key was found.
- 108716 `-C` Same as `-c`, except that a warning message shall not be sent to standard error if  
 108717 disorder or, with `-u`, a duplicate key is detected.
- 108718 `-m` Merge only; the input file shall be assumed to be already sorted.
- 108719 `-o output` Specify the name of an output file to be used instead of the standard output. This  
 108720 file can be the same as one of the input *files*.
- 108721 `-u` Unique: suppress all but one in each set of lines having equal keys. If used with  
 108722 the `-c` option, check that there are no lines with duplicate keys, in addition to  
 108723 checking that the input file is sorted.

108724 The following options shall override the default ordering rules. When ordering options appear  
 108725 independent of any key field specifications, the requested field ordering rules shall be applied  
 108726 globally to all sort keys. When attached to a specific key (see `-k`), the specified ordering options  
 108727 shall override all global ordering options for that key.

- 108728 `-d` Specify that only <blank> characters and alphanumeric characters, according to  
 108729 the current setting of `LC_CTYPE`, shall be significant in comparisons. The behavior  
 108730 is undefined for a sort key to which `-i` or `-n` also applies.



- 108731        **-f**        Consider all lowercase characters that have uppercase equivalents, according to  
 108732        the current setting of *LC\_CTYPE*, to be the uppercase equivalent for the purposes  
 108733        of comparison.
- 108734        **-i**        Ignore all characters that are non-printable, according to the current setting of  
 108735        *LC\_CTYPE*. The behavior is undefined for a sort key for which **-n** also applies.
- 108736        **-n**        Restrict the sort key to an initial numeric string, consisting of optional <blank>  
 108737        characters, optional <hyphen-minus> character, and zero or more digits with an  
 108738        optional radix character and thousands separators (as defined in the current  
 108739        locale), which shall be sorted by arithmetic value. An empty digit string shall be  
 108740        treated as zero. Leading zeros and signs on zeros shall not affect ordering.
- 108741        **-r**        Reverse the sense of comparisons.
- 108742        The treatment of field separators can be altered using the options:
- 108743        **-b**        Ignore leading <blank> characters when determining the starting and ending  
 108744        positions of a restricted sort key. If the **-b** option is specified before the first **-k**  
 108745        option, it shall be applied to all **-k** options. Otherwise, the **-b** option can be  
 108746        attached independently to each **-k** *field\_start* or *field\_end* option-argument (see  
 108747        below).
- 108748        **-t char**    Use *char* as the field separator character; *char* shall not be considered to be part of a  
 108749        field (although it can be included in a sort key). Each occurrence of *char* shall be  
 108750        significant (for example, <*char*><*char*> delimits an empty field). If **-t** is not  
 108751        specified, <blank> characters shall be used as default field separators; each  
 108752        maximal non-empty sequence of <blank> characters that follows a non-<blank>  
 108753        shall be a field separator.
- 108754        Sort keys can be specified using the options:
- 108755        **-k keydef**    The *keydef* argument is a restricted sort key field definition. The format of this  
 108756        definition is:
- 108757        *field\_start*[*type*][,*field\_end*[*type*]]
- 108758        where *field\_start* and *field\_end* define a key field restricted to a portion of the line  
 108759        (see the EXTENDED DESCRIPTION section), and *type* is one or more modifiers  
 108760        from the list of characters 'b', 'd', 'f', 'i', 'n', 'r'. The 'b' modifier shall  
 108761        behave like the **-b** option, but shall apply only to the *field\_start* or *field\_end* to  
 108762        which it is attached. The other modifiers shall behave like the corresponding  
 108763        options, but shall apply only to the key field to which they are attached; they shall  
 108764        have this effect if specified with *field\_start*, *field\_end*, or both. If any modifier is  
 108765        attached to a *field\_start* or to a *field\_end*, no option shall apply to either.  
 108766        Implementations shall support at least nine occurrences of the **-k** option, which  
 108767        shall be significant in command line order. If no **-k** option is specified, a default  
 108768        sort key of the entire line shall be used.
- 108769        When there are multiple key fields, later keys shall be compared only after all  
 108770        earlier keys compare equal. Except when the **-u** option is specified, lines that  
 108771        otherwise compare equal shall be ordered as if none of the options **-d**, **-f**, **-i**, **-n**, or  
 108772        **-k** were present (but with **-r** still in effect, if it was specified) and with all bytes in  
 108773        the lines significant to the comparison. The order in which lines that still compare  
 108774        equal are written is unspecified.

108775 **OPERANDS**

108776 The following operand shall be supported:

108777 *file* A pathname of a file to be sorted, merged, or checked. If no *file* operands are  
 108778 specified, or if a *file* operand is '-', the standard input shall be used. If *sort* |  
 108779 encounters an error when opening or reading a *file* operand, it may exit without |  
 108780 writing any output to standard output or processing later operands.

108781 **STDIN**

108782 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is '-'.  
 108783 See the INPUT FILES section.

108784 **INPUT FILES**

108785 The input files shall be text files, except that the *sort* utility shall add a <newline> to the end of a  
 108786 file ending with an incomplete last line.

108787 **ENVIRONMENT VARIABLES**108788 The following environment variables shall affect the execution of *sort*:

108789 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 108790 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 108791 variables used to determine the values of locale categories.)

108792 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 108793 internationalization variables.

108794 *LC\_COLLATE*  
 108795 Determine the locale for ordering rules.

108796 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 108797 characters (for example, single-byte as opposed to multi-byte characters in  
 108798 arguments and input files) and the behavior of character classification for the *-b*,  
 108799 *-d*, *-f*, *-i*, and *-n* options.

108800 *LC\_MESSAGES*  
 108801 Determine the locale that should be used to affect the format and contents of  
 108802 diagnostic messages written to standard error.

108803 *LC\_NUMERIC*  
 108804 Determine the locale for the definition of the radix character and thousands  
 108805 separator for the *-n* option.

108806 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

108807 **ASYNCHRONOUS EVENTS**

108808 Default.

108809 **STDOUT**108810 Unless the *-o* or *-c* options are in effect, the standard output shall contain the sorted input.108811 **STDERR**

108812 The standard error shall be used for diagnostic messages. When *-c* is specified, if disorder is  
 108813 detected (or if *-u* is also specified and a duplicate key is detected), a message shall be written to  
 108814 the standard error which identifies the input line at which disorder (or a duplicate key) was  
 108815 detected. A warning message about correcting an incomplete last line of an input file may be  
 108816 generated, but need not affect the final exit status.

108817 **OUTPUT FILES**

108818 If the **-o** option is in effect, the sorted input shall be written to the file *output*.

108819 **EXTENDED DESCRIPTION**

108820 The notation:

108821 `-k field_start[type][,field_end[type]]`

108822 shall define a key field that begins at *field\_start* and ends at *field\_end* inclusive, unless *field\_start*  
 108823 falls beyond the end of the line or after *field\_end*, in which case the key field is empty. A missing  
 108824 *field\_end* shall mean the last character of the line.

108825 A field comprises a maximal sequence of non-separating characters and, in the absence of option  
 108826 **-t**, any preceding field separator.

108827 The *field\_start* portion of the *keydef* option-argument shall have the form:

108828 `field_number[.first_character]`

108829 Fields and characters within fields shall be numbered starting with 1. The *field\_number* and  
 108830 *first\_character* pieces, interpreted as positive decimal integers, shall specify the first character to  
 108831 be used as part of a sort key. If *first\_character* is omitted, it shall refer to the first character of the  
 108832 field.

108833 The *field\_end* portion of the *keydef* option-argument shall have the form:

108834 `field_number[.last_character]`

108835 The *field\_number* shall be as described above for *field\_start*. The *last\_character* piece, interpreted  
 108836 as a non-negative decimal integer, shall specify the last character to be used as part of the sort  
 108837 key. If *last\_character* evaluates to zero or *last\_character* is omitted, it shall refer to the last  
 108838 character of the field specified by *field\_number*.

108839 If the **-b** option or **b** type modifier is in effect, characters within a field shall be counted from the  
 108840 first non-`<blank>` in the field. (This shall apply separately to *first\_character* and *last\_character*.)

108841 **EXIT STATUS**

108842 The following exit values shall be returned:

108843 0 All input files were output successfully, or **-c** was specified and the input file was correctly  
 108844 sorted.

108845 1 Under the **-c** option, the file was not ordered as specified, or if the **-c** and **-u** options were  
 108846 both specified, two input lines were found with equal keys.

108847 >1 An error occurred.

108848 **CONSEQUENCES OF ERRORS**

108849 The default requirements shall apply, except that if *sort* encounters an error when opening or  
 108850 reading a *file* operand, it may exit without writing any output to standard output or processing  
 108851 later operands.

APPLICATION USAGE

The default value for `-t`, `<blank>`, has different properties from, for example, `-t"<space>"`. If a line contains:

`<space><space>foo`

the following treatment would occur with default separation as opposed to specifically selecting a `<space>`:

| Field | Default                                    | <code>-t "&lt;space&gt;"</code> |
|-------|--------------------------------------------|---------------------------------|
| 1     | <code>&lt;space&gt;&lt;space&gt;foo</code> | <i>empty</i>                    |
| 2     | <i>empty</i>                               | <i>empty</i>                    |
| 3     | <i>empty</i>                               | foo                             |

The leading field separator itself is included in a field when `-t` is not used. For example, this command returns an exit status of zero, meaning the input was already sorted:

```
sort -c -k 2 <<eof
y<tab>b
x<space>a
eof
```

(assuming that a `<tab>` precedes the `<space>` in the current collating sequence). The field separator is not included in a field when it is explicitly set via `-t`. This is historical practice and allows usage such as:

```
sort -t "|" -k 2n <<eof
Atlanta|425022|Georgia
Birmingham|284413|Alabama
Columbia|100385|South Carolina
eof
```

where the second field can be correctly sorted numerically without regard to the non-numeric field separator.

The wording in the OPTIONS section clarifies that the `-b`, `-d`, `-f`, `-i`, `-n`, and `-r` options have to come before the first sort key specified if they are intended to apply to all specified keys. The way it is described in this volume of POSIX.1-2008 matches historical practice, not historical documentation. The results are unspecified if these options are specified after a `-k` option.

The `-f` option might not work as expected in locales where there is not a one-to-one mapping between an uppercase and a lowercase letter.

When using *sort* to process pathnames, it is recommended that `LC_ALL`, or at least `LC_CTYPE` and `LC_COLLATE`, are set to `POSIX` or `C` in the environment, since pathnames can contain byte sequences that do not form valid characters in some locales, in which case the utility's behavior would be undefined. In the `POSIX` locale each byte is a valid single-byte character, and therefore this problem is avoided.

If the collating sequence of the current locale does not have a total ordering of all characters, this can affect the behavior of *sort* in the following ways:

- As *sort* `-u` suppresses lines with duplicate keys, it suppresses lines that collate equally but are not identical.
- The output of *sort* (without `-u`) can contain identical lines that are not adjacent, if it does not implement the recommended further byte-by-byte comparison of lines that collate equally. This affects the use of *sort* with *comm* and *uniq*; see the APPLICATION USAGE for

108896 those utilities. +

## 108897 EXAMPLES

- 108898 1. The following command sorts the contents of **infile** with the second field as the sort key:  
 108899 `sort -k 2,2 infile`
- 108900 2. The following command sorts, in reverse order, the contents of **infile1** and **infile2**,  
 108901 placing the output in **outfile** and using the second character of the second field as the sort  
 108902 key (assuming that the first character of the second field is the field separator):  
 108903 `sort -r -o outfile -k 2.2,2.2 infile1 infile2`
- 108904 3. The following command sorts the contents of **infile1** and **infile2** using the second  
 108905 non-<blank> of the second field as the sort key:  
 108906 `sort -k 2.2b,2.2b infile1 infile2`
- 108907 4. The following command prints the System V password file (user database) sorted by the  
 108908 numeric user ID (the third <colon>-separated field):  
 108909 `sort -t : -k 3,3n /etc/passwd`
- 108910 5. The following command prints the lines of the already sorted file **infile**, suppressing all  
 108911 but one occurrence of lines having the same third field:  
 108912 `sort -um -k 3.1,3.0 infile`

## 108913 RATIONALE

108914 Examples in some historical documentation state that options **-um** with one input file keep the  
 108915 first in each set of lines with equal keys. This behavior was deemed to be an implementation  
 108916 artifact and was not standardized.

108917 The **-z** option was omitted; it is not standard practice on most systems and is inconsistent with  
 108918 using *sort* to sort several files individually and then merge them together. The text concerning **-z**  
 108919 in historical documentation appeared to require implementations to determine the proper buffer  
 108920 length during the sort phase of operation, but not during the merge.

108921 The **-y** option was omitted because of non-portability. The **-M** option, present in System V, was  
 108922 omitted because of non-portability in international usage.

108923 An undocumented **-T** option exists in some implementations. It is used to specify a directory for  
 108924 intermediate files. Implementations are encouraged to support the use of the *TMPDIR*  
 108925 environment variable instead of adding an option to support this functionality.

108926 The **-k** option was added to satisfy two objections. First, the zero-based counting used by *sort* is  
 108927 not consistent with other utility conventions. Second, it did not meet syntax guideline  
 108928 requirements.

108929 Historical documentation indicates that “setting **-n** implies **-b**”. The description of **-n** already  
 108930 states that optional leading <blank>s are tolerated in doing the comparison. If **-b** is enabled,  
 108931 rather than implied, by **-n**, this has unusual side-effects. When a character offset is used in a  
 108932 column of numbers (for example, to sort modulo 100), that offset is measured relative to the  
 108933 most significant digit, not to the column. Based upon a recommendation from the author of the  
 108934 original *sort* utility, the **-b** implication has been omitted from this volume of POSIX.1-2008, and  
 108935 an application wishing to achieve the previously mentioned side-effects has to code the **-b** flag  
 108936 explicitly.

108937 Earlier versions of this standard allowed the **-o** option to appear after operands. Historical  
 108938 practice allowed all options to be interspersed with operands. This version of the standard

allows implementations to accept options after operands but conforming applications should not use this form.

Earlier versions of this standard also allowed the *-number* and *+number* options. These options are no longer specified by POSIX.1-2008 but may be present in some implementations.

Historical implementations produced a message on standard error when *-c* was specified and disorder was detected, and when *-c* and *-u* were specified and a duplicate key was detected. An earlier version of this standard contained wording that did not make it clear that this message was allowed and some implementations removed this message to be sure that they conformed to the standard's requirements. Confronted with this difference in behavior, interactive users that wanted to be sure that they got visual feedback instead of just exit code 1 could have used a command like:

```
sort -c file || echo disorder
```

whether or not the *sort* utility provided a message in this case. But, it was not easy for a user to find where the disorder or duplicate key occurred on implementations that do not produce a message, especially when some parts of the input line were not part of the key and when one or more of the *-b*, *-d*, *-f*, *-i*, *-n*, or *-r* options or *keydef* type modifiers were in use. POSIX.1-2008 requires a message to be produced in this case. POSIX.1-2008 also contains the *-C* option giving users the ability to choose either behavior.

When a disorder or duplicate is found when the *-c* option is specified, some implementations print a message containing the first line that is out of order or contains a duplicate key; others print a message specifying the line number of the offending line. This standard allows either type of message.

Implementations are encouraged to perform the recommended further byte-by-byte comparison of lines that collate equally, even though this may affect efficiency. The impact on efficiency can be mitigated by only performing the additional comparison if the current locale's collating sequence does not have a total ordering of all characters (if the implementation provides a way to query this) or by only performing the additional comparison if the locale name associated with the LC\_COLLATE category has an '@' modifier in the name (since locales without an '@' modifier should have a total ordering of all characters — see XBD [Section 7.3.2](#), on page 147). Note that if the implementation provides a *stable sort* option as an extension (usually *-s*), the additional comparison should not be performed when this option has been specified.

## FUTURE DIRECTIONS

A future version of this standard may require that if the collating sequence of the current locale does not have a total ordering of all characters, any lines of input that collate equally when comparing them as whole lines are further compared byte-by-byte using the collating sequence for the POSIX locale.

## SEE ALSO

*comm*, *join*, *uniq*

XBD [Section 7.3.2](#) (on page 147), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

XSH *toupper()*

## CHANGE HISTORY

First released in Issue 2.

108981 **Issue 6**

108982 IEEE PASC Interpretation 1003.2 #174 is applied, updating the DESCRIPTION of comparisons.

108983 IEEE PASC Interpretation 1003.2 #168 is applied.

108984 **Issue 7**

108985 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying that Guideline 9 of the Utility  
108986 Syntax Guidelines does not apply and noting that '+' may be recognized as an option delimiter.

108987 Austin Group Interpretation 1003.1-2001 #120 is applied, clarifying the use of the -c option and  
108988 introducing the -C option.

108989 XCU-ERN-81 is applied, modifying the description of the -i option.

108990 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



108991 **NAME**108992 `split` — split a file into pieces |108993 **SYNOPSIS**108994 `split [-l line_count] [-a suffix_length] [file [name]]` |108995 `split -b n[k|m] [-a suffix_length] [file [name]]` |108996 **DESCRIPTION**

108997 The *split* utility shall read an input file and write zero or more output files. The default size of  
 108998 each output file shall be 1 000 lines. The size of the output files can be modified by specification  
 108999 of the `-b` or `-l` options. Each output file shall be created with a unique suffix. The suffix shall  
 109000 consist of exactly *suffix\_length* lowercase letters from the POSIX locale. The letters of the suffix  
 109001 shall be used as if they were a base-26 digit system, with the first suffix to be created consisting  
 109002 of all 'a' characters, the second with a 'b' replacing the last 'a', and so on, until a name of all  
 109003 'z' characters is created. By default, the names of the output files shall be 'x', followed by a  
 109004 two-character suffix from the character set as described above, starting with "aa", "ab", "ac",  
 109005 and so on, and continuing until the suffix "zz", for a maximum of 676 files.

109006 If the number of files required exceeds the maximum allowed by the suffix length provided,  
 109007 such that the last allowable file would be larger than the requested size, the *split* utility shall fail  
 109008 after creating the last file with a valid suffix; *split* shall not delete the files it created with valid  
 109009 suffixes. If the file limit is not exceeded, the last file created shall contain the remainder of the  
 109010 input file, and may be smaller than the requested size. If the input is an empty file, no output file  
 109011 shall be created and this shall not be considered to be an error.

109012 **OPTIONS**109013 The *split* utility shall conform to XBD [Section 12.2](#) (on page 216).

109014 The following options shall be supported:

109015 `-a suffix_length`

109016 Use *suffix\_length* letters to form the suffix portion of the filenames of the split file. If  
 109017 `-a` is not specified, the default suffix length shall be two. If the sum of the *name*  
 109018 operand and the *suffix\_length* option-argument would create a filename exceeding  
 109019 {NAME\_MAX} bytes, an error shall result; *split* shall exit with a diagnostic message  
 109020 and no files shall be created.

109021 `-b n` Split a file into pieces *n* bytes in size.109022 `-b nk` Split a file into pieces *n*\*1 024 bytes in size.109023 `-b nm` Split a file into pieces *n*\*1 048 576 bytes in size.

109024 `-l line_count` Specify the number of lines in each resulting file piece. The *line\_count* argument is  
 109025 an unsigned decimal integer. The default is 1 000. If the input does not end with a  
 109026 <newline>, the partial line shall be included in the last output file.

109027 **OPERANDS**

109028 The following operands shall be supported:

109029 *file* The pathname of the ordinary file to be split. If no input file is given or *file* is '-',  
 109030 the standard input shall be used.

109031 *name* The prefix to be used for each of the files resulting from the split operation. If no  
 109032 *name* argument is given, 'x' shall be used as the prefix of the output files. The  
 109033 combined length of the basename of *prefix* and *suffix\_length* cannot exceed  
 109034 {NAME\_MAX} bytes. See the OPTIONS section.



**109035 STDIN**

109036 See the INPUT FILES section.

**109037 INPUT FILES**

109038 Any file can be used as input.

**109039 ENVIRONMENT VARIABLES**

109040 The following environment variables shall affect the execution of *split*:

109041 *LANG* Provide a default value for the internationalization variables that are unset or null.  
109042 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
109043 variables used to determine the values of locale categories.)

109044 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
109045 internationalization variables.

109046 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
109047 characters (for example, single-byte as opposed to multi-byte characters in  
109048 arguments and input files).

109049 *LC\_MESSAGES*

109050 Determine the locale that should be used to affect the format and contents of  
109051 diagnostic messages written to standard error.

109052 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

**109053 ASYNCHRONOUS EVENTS**

109054 Default.

**109055 STDOUT**

109056 Not used.

**109057 STDERR**

109058 The standard error shall be used only for diagnostic messages.

**109059 OUTPUT FILES**

109060 The output files contain portions of the original input file; otherwise, unchanged.

**109061 EXTENDED DESCRIPTION**

109062 None.

**109063 EXIT STATUS**

109064 The following exit values shall be returned:

109065 0 Successful completion.

109066 >0 An error occurred.

**109067 CONSEQUENCES OF ERRORS**

109068 Default.

**109069 APPLICATION USAGE**

109070 None.

**109071 EXAMPLES**

109072 In the following examples **foo** is a text file that contains 5 000 lines.

109073 1. Create five files, **xaa**, **xab**, **xac**, **xad**, and **xae**:

109074 `split foo`

109075 2. Create five files, but the suffixed portion of the created files consists of three letters, **xaaa**,  
109076 **xaab**, **xaac**, **xaad**, and **xaae**:

109077 `split -a 3 foo`

109078 3. Create three files with four-letter suffixes and a supplied prefix, **bar\_aaaa**, **bar\_aaab**, and  
109079 **bar\_aaac**:

109080 `split -a 4 -l 2000 foo bar_`

109081 4. Create as many files as are necessary to contain at most 20\*1 024 bytes, each with the  
109082 default prefix of **x** and a five-letter suffix:

109083 `split -a 5 -b 20k foo`

**109084 RATIONALE**

109085 The **-b** option was added to provide a mechanism for splitting files other than by lines. While  
109086 most uses of the **-b** option are for transmitting files over networks, some believed it would have  
109087 additional uses.

109088 The **-a** option was added to overcome the limitation of being able to create only 676 files.

109089 Consideration was given to deleting this utility, using the rationale that the functionality  
109090 provided by this utility is available via the *csplit* utility (see *csplit*). Upon reconsideration of the  
109091 purpose of the User Portability Utilities option, it was decided to retain both this utility and the  
109092 *csplit* utility because users use both utilities and have historical expectations of their behavior.  
109093 Furthermore, the splitting on byte boundaries in *split* cannot be duplicated with the historical  
109094 *csplit*.

109095 The text “*split* shall not delete the files it created with valid suffixes” would normally be  
109096 assumed, but since the related utility, *csplit*, does delete files under some circumstances, the  
109097 historical behavior of *split* is made explicit to avoid misinterpretation.

109098 Earlier versions of this standard allowed a *-line\_count* option. This form is no longer specified by  
109099 POSIX.1-2008 but may be present in some implementations.

**109100 FUTURE DIRECTIONS**

109101 None.

**109102 SEE ALSO**

109103 *csplit*

109104 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

**109105 CHANGE HISTORY**

109106 First released in Issue 2.

**109107 Issue 6**

109108 This utility is marked as part of the User Portability Utilities option.

109109 The APPLICATION USAGE section is added.

- 109110 The obsolescent SYNOPSIS is removed.
- 109111 **Issue 7**
- 109112 Austin Group Interpretation 1003.1-2001 #027 is applied.
- 109113 The *split* utility is moved from the User Portability Utilities option to the Base. User Portability
- 109114 Utilities is now an option for interactive utilities.
- 109115 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

109116 **NAME**

109117 strings — find printable strings in files

109118 **SYNOPSIS**109119 strings [-a] [-t *format*] [-n *number*] [*file...*]109120 **DESCRIPTION**

109121 The *strings* utility shall look for printable strings in regular files and shall write those strings to  
109122 standard output. A printable string is any sequence of four (by default) or more printable  
109123 characters terminated by a <newline> or NUL character. Additional implementation-defined  
109124 strings may be written; see *localedef*.

109125 If the first argument is '-', the results are unspecified.

109126 **OPTIONS**109127 The *strings* utility shall conform to XBD [Section 12.2](#) (on page 216), except for the unspecified  
109128 usage of '-'.

109129 The following options shall be supported:

109130 -a Scan files in their entirety. If -a is not specified, it is implementation-defined what  
109131 portion of each file is scanned for strings.

109132 -n *number* Specify the minimum string length, where the *number* argument is a positive  
109133 decimal integer. The default shall be 4.

109134 -t *format* Write each string preceded by its byte offset from the start of the file. The format  
109135 shall be dependent on the single character used as the *format* option-argument:

109136 d The offset shall be written in decimal.

109137 o The offset shall be written in octal.

109138 x The offset shall be written in hexadecimal.

109139 **OPERANDS**

109140 The following operand shall be supported:

109141 *file* A pathname of a regular file to be used as input. If no *file* operand is specified, the  
109142 *strings* utility shall read from the standard input.

109143 **STDIN**

109144 See the INPUT FILES section.

109145 **INPUT FILES**

109146 The input files named by the utility arguments or the standard input shall be regular files of any  
109147 format.

109148 **ENVIRONMENT VARIABLES**109149 The following environment variables shall affect the execution of *strings*:

109150 LANG Provide a default value for the internationalization variables that are unset or null.  
109151 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
109152 variables used to determine the values of locale categories.)

109153 LC\_ALL If set to a non-empty string value, override the values of all the other  
109154 internationalization variables.

109155 LC\_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as  
109156 characters (for example, single-byte as opposed to multi-byte characters in  
109157 arguments and input files) and to identify printable strings.

109158 **LC\_MESSAGES**

109159 Determine the locale that should be used to affect the format and contents of  
 109160 diagnostic messages written to standard error.

109161 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

109162 **ASYNCHRONOUS EVENTS**

109163 Default.

109164 **STDOUT**

109165 Strings found shall be written to the standard output, one per line.

109166 When the **-t** option is not specified, the format of the output shall be:

109167 "%s", <string>

109168 With the **-t o** option, the format of the output shall be:

109169 "%o %s", <byte offset>, <string>

109170 With the **-t x** option, the format of the output shall be:

109171 "%x %s", <byte offset>, <string>

109172 With the **-t d** option, the format of the output shall be:

109173 "%d %s", <byte offset>, <string>

109174 **STDERR**

109175 The standard error shall be used only for diagnostic messages.

109176 **OUTPUT FILES**

109177 None.

109178 **EXTENDED DESCRIPTION**

109179 None.

109180 **EXIT STATUS**

109181 The following exit values shall be returned:

109182 0 Successful completion.

109183 >0 An error occurred.

109184 **CONSEQUENCES OF ERRORS**

109185 Default.

109186 **APPLICATION USAGE**

109187 By default the data area (as opposed to the text, "bss", or header areas) of a binary executable  
 109188 file is scanned. Implementations document which areas are scanned.

109189 Some historical implementations do not require NUL or <newline> terminators for strings to  
 109190 permit those languages that do not use NUL as a string terminator to have their strings written.

109191 **EXAMPLES**

109192 None.

109193 **RATIONALE**

109194 Apart from rationalizing the option syntax and slight difficulties with object and executable  
 109195 binary files, *strings* is specified to match historical practice closely. The **-a** and **-n** options were  
 109196 introduced to replace the non-conforming **-** and **-number** options. These options are no longer  
 109197 specified by POSIX.1-2008 but may be present in some implementations.

- 109198 The `-o` option historically means different things on different implementations. Some use it to  
109199 mean “*offset* in decimal”, while others use it as “*offset* in octal”. Instead of trying to decide which  
109200 way would be least objectionable, the `-t` option was added. It was originally named `-O` to mean  
109201 “*offset*”, but was changed to `-t` to be consistent with *od*.
- 109202 The ISO C standard function *isprint()* is restricted to a domain of **unsigned char**. This volume of  
109203 POSIX.1-2008 requires implementations to write strings as defined by the current locale.
- 109204 **FUTURE DIRECTIONS**
- 109205 None.
- 109206 **SEE ALSO**
- 109207 *localedef*, *nm*
- 109208 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)
- 109209 **CHANGE HISTORY**
- 109210 First released in Issue 4.
- 109211 **Issue 6**
- 109212 This utility is marked as part of the User Portability Utilities option.
- 109213 The obsolescent SYNOPSIS is removed.
- 109214 The normative text is reworded to avoid use of the term “must” for application requirements.
- 109215 **Issue 7**
- 109216 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying the behavior if the first  
109217 argument is ‘-’.
- 109218 The *strings* utility is moved from the User Portability Utilities option to the Base. User  
109219 Portability Utilities is now an option for interactive utilities.
- 109220 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

109221 **NAME**109222 strip — remove unnecessary information from strippable files (**DEVELOPMENT**)109223 **SYNOPSIS**

109224 SD strip file...

109225 **DESCRIPTION**

109226 XSI A strippable file is defined as a relocatable, object, or executable file. On XSI-conformant  
 109227 systems, a strippable file can also be an archive of object or relocatable files.

109228 The *strip* utility shall remove from strippable files named by the *file* operands any information  
 109229 the implementor deems unnecessary for execution of those files. The nature of that information  
 109230 is unspecified. The effect of *strip* on object and executable files shall be similar to the use of the  
 109231 XSI *-s* option to *c99* or *fort77*. The effect of *strip* on an archive of object files shall be similar to the  
 109232 use of the *-s* option to *c99* or *fort77* for each object file in the archive.

109233 **OPTIONS**

109234 None.

109235 **OPERANDS**

109236 The following operand shall be supported:

109237 *file* A pathname referring to a strippable file.

109238 **STDIN**

109239 Not used.

109240 **INPUT FILES**

109241 The input files shall be in the form of strippable files successfully produced by any compiler  
 109242 XSI defined by this volume of POSIX.1-2008 or produced by creating or updating an archive of such  
 109243 files using the *ar* utility.

109244 **ENVIRONMENT VARIABLES**109245 The following environment variables shall affect the execution of *strip*:

109246 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 109247 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 109248 variables used to determine the values of locale categories.)

109249 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 109250 internationalization variables.

109251 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 109252 characters (for example, single-byte as opposed to multi-byte characters in  
 109253 arguments).

109254 *LC\_MESSAGES*

109255 Determine the locale that should be used to affect the format and contents of  
 109256 diagnostic messages written to standard error.

109257 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

109258 **ASYNCHRONOUS EVENTS**

109259 Default.

109260 **STDOUT**

109261 Not used.

**109262 STDERR**

109263 The standard error shall be used only for diagnostic messages.

**109264 OUTPUT FILES**

109265 The *strip* utility shall produce strippable files of unspecified format.

**109266 EXTENDED DESCRIPTION**

109267 None.

**109268 EXIT STATUS**

109269 The following exit values shall be returned:

109270 0 Successful completion.

109271 >0 An error occurred.

**109272 CONSEQUENCES OF ERRORS**

109273 Default.

**109274 APPLICATION USAGE**

109275 None.

**109276 EXAMPLES**

109277 None.

**109278 RATIONALE**

109279 Historically, this utility has been used to remove the symbol table from a strippable file. It was  
109280 included since it is known that the amount of symbolic information can amount to several  
109281 megabytes; the ability to remove it in a portable manner was deemed important, especially for  
109282 smaller systems.

109283 The behavior of *strip* on object and executable files is said to be the same as the *-s* option to a  
109284 compiler. While the end result is essentially the same, it is not required to be identical.

109285 XSI-conformant systems support use of *strip* on archive files containing object files or relocatable  
109286 files.

**109287 FUTURE DIRECTIONS**

109288 None.

**109289 SEE ALSO**

109290 *ar*, *c99*, *fort77*

109291 XBD Chapter 8 (on page 173)

**109292 CHANGE HISTORY**

109293 First released in Issue 2.

**109294 Issue 6**

109295 This utility is marked as part of the Software Development Utilities option.

**109296 Issue 7**

109297 Austin Group Interpretation 1003.1-2001 #103 is applied.



109298 **NAME**

109299       stty — set the options for a terminal

109300 **SYNOPSIS**

109301       stty [-a|-g]

109302       stty operand...

109303 **DESCRIPTION**

109304       The *stty* utility shall set or report on terminal I/O characteristics for the device that is its  
 109305       standard input. Without options or operands specified, it shall report the settings of certain  
 109306       characteristics, usually those that differ from implementation-defined defaults. Otherwise, it  
 109307       shall modify the terminal state according to the specified operands. Detailed information about  
 109308       the modes listed in the first five groups below are described in XBD [Chapter 11](#) (on page 199).  
 109309       Operands in the Combination Modes group (see [Combination Modes](#), on page 3258) are  
 109310       implemented using operands in the previous groups. Some combinations of operands are  
 109311       mutually-exclusive on some terminal types; the results of using such combinations are  
 109312       unspecified.

109313       Typical implementations of this utility require a communications line configured to use the  
 109314       **termios** interface defined in the System Interfaces volume of POSIX.1-2008. On systems where  
 109315       none of these lines are available, and on lines not currently configured to support the **termios**  
 109316       interface, some of the operands need not affect terminal characteristics.

109317 **OPTIONS**109318       The *stty* utility shall conform to XBD [Section 12.2](#) (on page 216).

109319       The following options shall be supported:

109320       **-a**           Write to standard output all the current settings for the terminal.

109321       **-g**           Write to standard output all the current settings in an unspecified form that can be  
 109322       used as arguments to another invocation of the *stty* utility on the same system. The  
 109323       form used shall not contain any characters that would require quoting to avoid  
 109324       word expansion by the shell; see [Section 2.6](#) (on page 2343).

109325 **OPERANDS**

109326       The following operands shall be supported to set the terminal characteristics.

109327       **Control Modes**

109328       **parenb** (**-parenb**) Enable (disable) parity generation and detection. This shall have the effect of  
 109329       setting (not setting) PARENB in the **termios** *c\_flag* field, as defined in XBD  
 109330       [Chapter 11](#) (on page 199).

109331       **parodd** (**-parodd**) Select odd (even) parity. This shall have the effect of setting (not setting)  
 109332       PARODD in the **termios** *c\_flag* field, as defined in XBD [Chapter 11](#) (on page  
 109333       199).  
 109334

109335       **cs5 cs6 cs7 cs8** Select character size, if possible. This shall have the effect of setting CS5, CS6,  
 109336       CS7, and CS8, respectively, in the **termios** *c\_flag* field, as defined in XBD  
 109337       [Chapter 11](#) (on page 199).

109338       **number**       Set terminal baud rate to the number given, if possible. If the baud rate is set  
 109339       to zero, the modem control lines shall no longer be asserted. This shall have  
 109340       the effect of setting the input and output **termios** baud rate values as defined  
 109341       in XBD [Chapter 11](#) (on page 199).

- 109342 **ispeed** *number* Set terminal input baud rate to the number given, if possible. If the input baud  
 109343 rate is set to zero, the input baud rate shall be specified by the value of the  
 109344 output baud rate. This shall have the effect of setting the input **termios** baud  
 109345 rate values as defined in XBD [Chapter 11](#) (on page 199).
- 109346 **ospeed** *number* Set terminal output baud rate to the number given, if possible. If the output  
 109347 baud rate is set to zero, the modem control lines shall no longer be asserted.  
 109348 This shall have the effect of setting the output **termios** baud rate values as  
 109349 defined in XBD [Chapter 11](#) (on page 199).
- 109350 **hupcl** (**-hupcl**) Stop asserting modem control lines (do not stop asserting modem control  
 109351 lines) on last close. This shall have the effect of setting (not setting) HUPCL in  
 109352 the **termios** *c\_cflag* field, as defined in XBD [Chapter 11](#) (on page 199).
- 109353 **hup** (**-hup**) Equivalent to **hupcl**(**-hupcl**).
- 109354 **cstopb** (**-cstopb**) Use two (one) stop bits per character. This shall have the effect of setting (not  
 109355 setting) CSTOPB in the **termios** *c\_cflag* field, as defined in XBD [Chapter 11](#) (on  
 109356 page 199).
- 109357 **cread** (**-cread**) Enable (disable) the receiver. This shall have the effect of setting (not setting)  
 109358 CREAD in the **termios** *c\_cflag* field, as defined in XBD [Chapter 11](#) (on page  
 109359 199).
- 109360 **clocal** (**-clocal**) Assume a line without (with) modem control. This shall have the effect of  
 109361 setting (not setting) CLOCAL in the **termios** *c\_cflag* field, as defined in XBD  
 109362 [Chapter 11](#) (on page 199).

109363 It is unspecified whether *stty* shall report an error if an attempt to set a Control Mode fails.

#### 109364 Input Modes

- 109365 **ignbrk** (**-ignbrk**) Ignore (do not ignore) break on input. This shall have the effect of setting (not  
 109366 setting) IGNBRK in the **termios** *c\_iflag* field, as defined in XBD [Chapter 11](#) (on  
 109367 page 199).
- 109368 **brkint** (**-brkint**) Signal (do not signal) INTR on break. This shall have the effect of setting (not  
 109369 setting) BRKINT in the **termios** *c\_iflag* field, as defined in XBD [Chapter 11](#) (on  
 109370 page 199).
- 109371 **ignpar** (**-ignpar**) Ignore (do not ignore) bytes with parity errors. This shall have the effect of  
 109372 setting (not setting) IGNPAR in the **termios** *c\_iflag* field, as defined in XBD  
 109373 [Chapter 11](#) (on page 199).
- 109374 **parmrk** (**-parmrk**)  
 109375 Mark (do not mark) parity errors. This shall have the effect of setting (not  
 109376 setting) PARMRK in the **termios** *c\_iflag* field, as defined in XBD [Chapter 11](#) (on  
 109377 page 199).
- 109378 **inpck** (**-inpck**) Enable (disable) input parity checking. This shall have the effect of setting (not  
 109379 setting) INPCK in the **termios** *c\_iflag* field, as defined in XBD [Chapter 11](#) (on  
 109380 page 199).
- 109381 **istrip** (**-istrip**) Strip (do not strip) input characters to seven bits. This shall have the effect of  
 109382 setting (not setting) ISTRIP in the **termios** *c\_iflag* field, as defined in XBD  
 109383 [Chapter 11](#) (on page 199).

|        |                                |                                                                                                  |
|--------|--------------------------------|--------------------------------------------------------------------------------------------------|
| 109384 | <b>inlcr</b> ( <b>-inlcr</b> ) | Map (do not map) NL to CR on input. This shall have the effect of setting (not                   |
| 109385 |                                | setting) INLCR in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD Chapter 11 (on      |
| 109386 |                                | page 199).                                                                                       |
| 109387 | <b>igncr</b> ( <b>-igncr</b> ) | Ignore (do not ignore) CR on input. This shall have the effect of setting (not                   |
| 109388 |                                | setting) IGNCR in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD Chapter 11 (on      |
| 109389 |                                | page 199).                                                                                       |
| 109390 | <b>icrnl</b> ( <b>-icrnl</b> ) | Map (do not map) CR to NL on input. This shall have the effect of setting (not                   |
| 109391 |                                | setting) ICRNL in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD Chapter 11 (on      |
| 109392 |                                | page 199).                                                                                       |
| 109393 | <b>ixon</b> ( <b>-ixon</b> )   | Enable (disable) START/STOP output control. Output from the system is                            |
| 109394 |                                | stopped when the system receives STOP and started when the system receives                       |
| 109395 |                                | START. This shall have the effect of setting (not setting) IXON in the <b>termios</b>            |
| 109396 |                                | <i>c_iflag</i> field, as defined in XBD Chapter 11 (on page 199).                                |
| 109397 | <b>ixany</b> ( <b>-ixany</b> ) | Allow any character to restart output. This shall have the effect of setting (not                |
| 109398 |                                | setting) IXANY in the <b>termios</b> <i>c_iflag</i> field, as defined in XBD Chapter 11 (on      |
| 109399 |                                | page 199).                                                                                       |
| 109400 | <b>ixoff</b> ( <b>-ixoff</b> ) | Request that the system send (not send) STOP characters when the input                           |
| 109401 |                                | queue is nearly full and START characters to resume data transmission. This                      |
| 109402 |                                | shall have the effect of setting (not setting) IXOFF in the <b>termios</b> <i>c_iflag</i> field, |
| 109403 |                                | as defined in XBD Chapter 11 (on page 199).                                                      |

#### Output Modes

|            |                                  |                                                                                                |
|------------|----------------------------------|------------------------------------------------------------------------------------------------|
| 109404     | <b>opost</b> ( <b>-opost</b> )   | Post-process output (do not post-process output; ignore all other output                       |
| 109405     |                                  | modes). This shall have the effect of setting (not setting) OPOST in the <b>termios</b>        |
| 109406     |                                  | <i>c_oflag</i> field, as defined in XBD Chapter 11 (on page 199).                              |
| 109407     |                                  |                                                                                                |
| 109408 XSI | <b>onlcr</b> ( <b>-onlcr</b> )   | Map (do not map) NL to CR-NL on output. This shall have the effect of setting +                |
| 109409     |                                  | (not setting) ONLCR in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD Chapter 11 + |
| 109410     |                                  | (on page 199).                                                                                 |
| 109411 XSI | <b>ocrnl</b> ( <b>-ocrnl</b> )   | Map (do not map) CR to NL on output. This shall have the effect of setting                     |
| 109412     |                                  | (not setting) OCRNL in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD Chapter 11   |
| 109413     |                                  | (on page 199).                                                                                 |
| 109414 XSI | <b>onocr</b> ( <b>-onocr</b> )   | Do not (do) output CR at column zero. This shall have the effect of setting (not               |
| 109415     |                                  | setting) ONOCR in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD Chapter 11 (on    |
| 109416     |                                  | page 199).                                                                                     |
| 109417 XSI | <b>onlret</b> ( <b>-onlret</b> ) | The terminal newline key performs (does not perform) the CR function. This                     |
| 109418     |                                  | shall have the effect of setting (not setting) ONLRET in the <b>termios</b> <i>c_oflag</i>     |
| 109419     |                                  | field, as defined in XBD Chapter 11 (on page 199).                                             |
| 109420 XSI | <b>ofill</b> ( <b>-ofill</b> )   | Use fill characters (use timing) for delays. This shall have the effect of setting             |
| 109421     |                                  | (not setting) OFILL in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD Chapter 11   |
| 109422     |                                  | (on page 199).                                                                                 |
| 109423 XSI | <b>ofdel</b> ( <b>-ofdel</b> )   | Fill characters are DELs (NULs). This shall have the effect of setting (not                    |
| 109424     |                                  | setting) OFDEL in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD Chapter 11 (on    |
| 109425     |                                  | page 199).                                                                                     |

|        |     |                            |                                                                                                                                                                                                                                                                                                                                     |
|--------|-----|----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 109426 | XSI | <b>cr0 cr1 cr2 cr3</b>     | Select the style of delay for CRs. This shall have the effect of setting CRDLY to CR0, CR1, CR2, or CR3, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                      |
| 109427 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109428 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109429 | XSI | <b>nl0 nl1</b>             | Select the style of delay for NL. This shall have the effect of setting NLDLY to NL0 or NL1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                                  |
| 109430 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109431 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109432 | XSI | <b>tab0 tab1 tab2 tab3</b> | Select the style of delay for horizontal tabs. This shall have the effect of setting TABDLY to TAB0, TAB1, TAB2, or TAB3, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199). Note that TAB3 has the effect of expanding <tab> characters to <space> characters.  |
| 109433 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109434 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109435 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109436 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109437 | XSI | <b>tabs (-tabs)</b>        | Synonym for <b>tab0 (tab3)</b> .                                                                                                                                                                                                                                                                                                    |
| 109438 | XSI | <b>bs0 bs1</b>             | Select the style of delay for <backspace> characters. This shall have the effect of setting BSDLY to BS0 or BS1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                              |
| 109439 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109440 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109441 | XSI | <b>ff0 ff1</b>             | Select the style of delay for <form-feed> characters. This shall have the effect of setting FFDLY to FF0 or FF1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                              |
| 109442 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109443 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109444 | XSI | <b>vt0 vt1</b>             | Select the style of delay for <vertical-tab> characters. This shall have the effect of setting VTDLY to VT0 or VT1, respectively, in the <b>termios</b> <i>c_oflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                           |
| 109445 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109446 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109447 |     | <b>Local Modes</b>         |                                                                                                                                                                                                                                                                                                                                     |
| 109448 |     | <b>isig (-isig)</b>        | Enable (disable) the checking of characters against the special control characters INTR, QUIT, and SUSP. This shall have the effect of setting (not setting) ISIG in the <b>termios</b> <i>c_lflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                           |
| 109449 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109450 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109451 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109452 |     | <b>icanon (-icanon)</b>    | Enable (disable) canonical input (ERASE and KILL processing). This shall have the effect of setting (not setting) ICANON in the <b>termios</b> <i>c_lflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                    |
| 109453 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109454 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109455 |     | <b>iexten (-iexten)</b>    | Enable (disable) any implementation-defined special control characters not currently controlled by <b>icanon</b> , <b>isig</b> , <b>ixon</b> , or <b>ixoff</b> . This shall have the effect of setting (not setting) IEXTEN in the <b>termios</b> <i>c_lflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199). |
| 109456 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109457 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109458 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109459 |     | <b>echo (-echo)</b>        | Echo back (do not echo back) every character typed. This shall have the effect of setting (not setting) ECHO in the <b>termios</b> <i>c_lflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                                |
| 109460 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109461 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109462 |     | <b>echoe (-echoe)</b>      | The ERASE character visually erases (does not erase) the last character in the current line from the display, if possible. This shall have the effect of setting (not setting) ECHOE in the <b>termios</b> <i>c_lflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                        |
| 109463 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109464 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109465 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109466 |     | <b>echok (-echok)</b>      | Echo (do not echo) NL after KILL character. This shall have the effect of setting (not setting) ECHOK in the <b>termios</b> <i>c_lflag</i> field, as defined in XBD <a href="#">Chapter 11</a> (on page 199).                                                                                                                       |
| 109467 |     |                            |                                                                                                                                                                                                                                                                                                                                     |
| 109468 |     |                            |                                                                                                                                                                                                                                                                                                                                     |

**echonl** (**-echonl**) Echo (do not echo) NL, even if **echo** is disabled. This shall have the effect of setting (not setting) ECHONL in the **termios** *c\_lflag* field, as defined in XBD Chapter 11 (on page 199).

**noflsh** (**-noflsh**) Disable (enable) flush after INTR, QUIT, SUSP. This shall have the effect of setting (not setting) NOFLSH in the **termios** *c\_lflag* field, as defined in XBD Chapter 11 (on page 199).

**tostop** (**-tostop**) Send SIGTTOU for background output. This shall have the effect of setting (not setting) TOSTOP in the **termios** *c\_lflag* field, as defined in XBD Chapter 11 (on page 199).

## Special Control Character Assignments

*<control>-character string*

Set *<control>-character* to *string*. If *<control>-character* is one of the character sequences in the first column of the following table, the corresponding XBD Chapter 11 (on page 199) control character from the second column shall be recognized. This has the effect of setting the corresponding element of the **termios** *c\_cc* array (see XBD Chapter 13 (on page 219), **<termios.h>**).

**Table 4-20** Control Character Names in *stty*

| Control Character | c_cc Subscript | Description     |
|-------------------|----------------|-----------------|
| <b>eof</b>        | VEOF           | EOF character   |
| <b>eol</b>        | VEOL           | EOL character   |
| <b>erase</b>      | VERASE         | ERASE character |
| <b>intr</b>       | VINTR          | INTR character  |
| <b>kill</b>       | VKILL          | KILL character  |
| <b>quit</b>       | VQUIT          | QUIT character  |
| <b>susp</b>       | VSUSP          | SUSP character  |
| <b>start</b>      | VSTART         | START character |
| <b>stop</b>       | VSTOP          | STOP character  |

If *string* is a single character, the control character shall be set to that character. If *string* is the two-character sequence "**^**-" or the string *undef*, the control character shall be set to **\_POSIX\_VDISABLE**, if it is in effect for the device; if **\_POSIX\_VDISABLE** is not in effect for the device, it shall be treated as an error. In the POSIX locale, if *string* is a two-character sequence beginning with **<circumflex>** (**' ^ '**), and the second character is one of those listed in the "**^c**" column of the following table, the control character shall be set to the corresponding character value in the Value column of the table.

**Table 4-21** Circumflex Control Characters in *stty*

109504  
109505  
109506  
109507  
109508  
109509  
109510  
109511  
109512  
109513  
109514  
109515

| <b>^c</b> | <b>Value</b> | <b>^c</b> | <b>Value</b> | <b>^c</b> | <b>Value</b> |
|-----------|--------------|-----------|--------------|-----------|--------------|
| a, A      | <SOH>        | l, L      | <FF>         | w, W      | <ETB>        |
| b, B      | <STX>        | m, M      | <CR>         | x, X      | <CAN>        |
| c, C      | <ETX>        | n, N      | <SO>         | y, Y      | <EM>         |
| d, D      | <EOT>        | o, O      | <SI>         | z, Z      | <SUB>        |
| e, E      | <ENQ>        | p, P      | <DLE>        | [         | <ESC>        |
| f, F      | <ACK>        | q, Q      | <DC1>        | \         | <FS>         |
| g, G      | <BEL>        | r, R      | <DC2>        | ]         | <GS>         |
| h, H      | <BS>         | s, S      | <DC3>        | ^         | <RS>         |
| i, I      | <HT>         | t, T      | <DC4>        | _         | <US>         |
| j, J      | <LF>         | u, U      | <NAK>        | ?         | <DEL>        |
| k, K      | <VT>         | v, V      | <SYN>        |           |              |

109516  
109517  
109518  
  
109519  
109520  
109521

**min number**

Set the value of MIN to *number*. MIN is used in non-canonical mode input processing (**icanon**).

**time number**

Set the value of TIME to *number*. TIME is used in non-canonical mode input processing (**icanon**).

109522

**Combination Modes**

109523  
109524

**saved settings**

Set the current terminal characteristics to the saved settings produced by the **-g** option.

109525  
109526

**evenp or parity**

Enable **parenb** and **cs7**; disable **parodd**.

109527  
109528

**oddp**

Enable **parenb**, **cs7**, and **parodd**.

109529  
109530

**-parity, -evenp, or -oddp**

Disable **parenb**, and set **cs8**.

109531 XSI  
109532  
  
109533  
109534

**raw (-raw or cooked)**

Enable (disable) raw input and output. Raw mode shall be equivalent to setting:

```
stty cs8 erase ^- kill ^- intr ^- \  
quit ^- eof ^- eol ^- -post -inpck
```

109535  
109536  
  
109537  
  
109538  
109539

**nl (-nl)**

Disable (enable) **icrnl**. In addition, **-nl** unsets **inlcr** and **igncr**.

**ek** Reset ERASE and KILL characters back to system defaults.

**sane**

Reset all modes to some reasonable, unspecified, values.

109540 **STDIN**

109541  
109542

Although no input is read from standard input, standard input shall be used to get the current terminal I/O characteristics and to set new terminal I/O characteristics.



109543 **INPUT FILES**

109544       None.

109545 **ENVIRONMENT VARIABLES**109546       The following environment variables shall affect the execution of *stty*:

109547       **LANG**       Provide a default value for the internationalization variables that are unset or null.  
 109548                   (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 109549                   variables used to determine the values of locale categories.)

109550       **LC\_ALL**     If set to a non-empty string value, override the values of all the other  
 109551                   internationalization variables.

109552       **LC\_CTYPE**   This variable determines the locale for the interpretation of sequences of bytes of  
 109553                   text data as characters (for example, single-byte as opposed to multi-byte  
 109554                   characters in arguments) and which characters are in the class **print**.

109555       **LC\_MESSAGES**

109556                   Determine the locale that should be used to affect the format and contents of  
 109557                   diagnostic messages written to standard error.

109558 XSI       **NLSPATH**   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

109559 **ASYNCHRONOUS EVENTS**

109560       Default.

109561 **STDOUT**

109562       If operands are specified, no output shall be produced.

109563       If the **-g** option is specified, *stty* shall write to standard output the current settings in a form that  
 109564       can be used as arguments to another instance of *stty* on the same system.

109565       If the **-a** option is specified, all of the information as described in the OPERANDS section shall  
 109566       be written to standard output. Unless otherwise specified, this information shall be written as  
 109567       <space>-separated tokens in an unspecified format, on one or more lines, with an unspecified  
 109568       number of tokens per line. Additional information may be written.

109569       If no options or operands are specified, an unspecified subset of the information written for the  
 109570       **-a** option shall be written.

109571       If speed information is written as part of the default output, or if the **-a** option is specified and if  
 109572       the terminal input speed and output speed are the same, the speed information shall be written  
 109573       as follows:

109574       "speed %d baud;", &lt;speed&gt;

109575       Otherwise, speeds shall be written as:

109576       "ispeed %d baud; ospeed %d baud;", &lt;ispeed&gt;, &lt;ospeed&gt;

109577       In locales other than the POSIX locale, the word **baud** may be changed to something more  
 109578       appropriate in those locales.

109579       If control characters are written as part of the default output, or if the **-a** option is specified,  
 109580       control characters shall be written as:

109581       "%s = %s;", &lt;control-character name&gt;, &lt;value&gt;

109582       where <value> is either the character, or some visual representation of the character if it is non-  
 109583       printable, or the string *undef* if the character is disabled.

109584 **STDERR**

109585 The standard error shall be used only for diagnostic messages.

109586 **OUTPUT FILES**

109587 None.

109588 **EXTENDED DESCRIPTION**

109589 None.

109590 **EXIT STATUS**

109591 The following exit values shall be returned:

109592 0 The terminal options were read or set successfully.

109593 >0 An error occurred.

109594 **CONSEQUENCES OF ERRORS**

109595 Default.

109596 **APPLICATION USAGE**

109597 The **-g** flag is designed to facilitate the saving and restoring of terminal state from the shell level.

109598 For example, a program may:

```
109599 saveterm="$(stty -g)"          # save terminal state
109600 stty (new settings)           # set new state
109601 ...                           # ...
109602 stty $saveterm               # restore terminal state
```

109603 Since the format is unspecified, the saved value is not portable across systems.

109604 Since the **-a** format is so loosely specified, scripts that save and restore terminal settings should  
109605 use the **-g** option.

109606 **EXAMPLES**

109607 None.

109608 **RATIONALE**

109609 The original *stty* description was taken directly from System V and reflected the System V  
109610 terminal driver **termio**. It has been modified to correspond to the terminal driver **termios**.

109611 Output modes are specified only for XSI-conformant systems. All implementations are expected  
109612 to provide *stty* operands corresponding to all of the output modes they support.

109613 The *stty* utility is primarily used to tailor the user interface of the terminal, such as selecting the  
109614 preferred ERASE and KILL characters. As an application programming utility, *stty* can be used  
109615 within shell scripts to alter the terminal settings for the duration of the script.

109616 The **termios** section states that individual disabling of control characters is possible through the  
109617 option **\_POSIX\_VDISABLE**. If enabled, two conventions currently exist for specifying this:  
109618 System V uses "**^-**", and BSD uses *undef*. Both are accepted by *stty* in this volume of  
109619 POSIX.1-2008. The other BSD convention of using the letter '**u**' was rejected because it conflicts  
109620 with the actual letter '**u**', which is an acceptable value for a control character.

109621 Early proposals did not specify the mapping of "**^c**" to control characters because the control  
109622 characters were not specified in the POSIX locale character set description file requirements. The  
109623 control character set is now specified in XBD [Chapter 3](#) (on page 33), so the historical mapping is  
109624 specified. Note that although the mapping corresponds to control-character key assignments on  
109625 many terminals that use the ISO/IEC 646:1991 standard (or ASCII) character encodings, the  
109626 mapping specified here is to the control characters, not their keyboard encodings.



109627 Since **termios** supports separate speeds for input and output, two new options were added to  
109628 specify each distinctly.

109629 Some historical implementations use standard input to get and set terminal characteristics;  
109630 others use standard output. Since input from a login TTY is usually restricted to the owner while  
109631 output to a TTY is frequently open to anyone, using standard input provides fewer chances of  
109632 accidentally (or maliciously) altering the terminal settings of other users. Using standard input  
109633 also allows *stty -a* and *stty -g* output to be redirected for later use. Therefore, usage of standard  
109634 input is required by this volume of POSIX.1-2008.

#### 109635 FUTURE DIRECTIONS

109636 None.

#### 109637 SEE ALSO

109638 [Chapter 2](#) (on page 2335)

109639 XBD [Chapter 8](#) (on page 173), [Chapter 11](#) (on page 199), [Section 12.2](#) (on page 216), [<termios.h>](#)

#### 109640 CHANGE HISTORY

109641 First released in Issue 2.

#### 109642 Issue 5

109643 The description of **tabs** is clarified.

109644 The FUTURE DIRECTIONS section is added.

#### 109645 Issue 6

109646 The LEGACY items **iuclic(-iuclic)**, **xcase**, **olcuc(-olcuc)**, **lcase(-lcase)**, and **LCASE(-LCASE)** are  
109647 removed.

109648 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/37 is applied, applying IEEE PASC  
109649 Interpretation 1003.2 #133, fixing an error in the OPERANDS section for the Combination Modes  
109650 **nl(-nl)**.

#### 109651 Issue 7

109652 Austin Group Interpretation 1003.1-2001 #144 is applied, moving functionality relating to the  
109653 **IXANY** symbol from the **XSI** option to the Base.

109654 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

109655 **NAME**

109656 tabs — set terminal tabs

109657 **SYNOPSIS**

109658 XSI tabs [-n|-a|-a2|-c|-c2|-c3|-f|-p|-s|-u] [-T type]

109659 tabs [-T type] n[[sep][+]*n*]. . .]

109660 **DESCRIPTION**

109661 The *tabs* utility shall display a series of characters that first clears the hardware terminal tab  
109662 XSI settings and then initializes the tab stops at the specified positions and optionally adjusts the  
109663 margin.

109664 The phrase “tab-stop position *N*” shall be taken to mean that, from the start of a line of output,  
109665 tabbing to position *N* shall cause the next character output to be in the (*N*+1)th column position  
109666 on that line. The maximum number of tab stops allowed is terminal-dependent.

109667 It need not be possible to implement *tabs* on certain terminals. If the terminal type obtained from  
109668 the *TERM* environment variable or *-T* option represents such a terminal, an appropriate  
109669 diagnostic message shall be written to standard error and *tabs* shall exit with a status greater  
109670 than zero.

109671 **OPTIONS**

109672 XSI The *tabs* utility shall conform to XBD Section 12.2 (on page 216), except for various extensions:  
109673 the options *-a2*, *-c2*, and *-c3* are multi-character.

109674 The following options shall be supported:

109675 *-n* Specify repetitive tab stops separated by a uniform number of column positions, *n*,  
109676 where *n* is a single-digit decimal number. The default usage of *tabs* with no  
109677 arguments shall be equivalent to *tabs -8*. When *-0* is used, the tab stops shall be  
109678 cleared and no new ones set.

109679 XSI *-a* 1,10,16,36,72  
109680 Assembler, applicable to some mainframes.

109681 XSI *-a2* 1,10,16,40,72  
109682 Assembler, applicable to some mainframes.

109683 XSI *-c* 1,8,12,16,20,55  
109684 COBOL, normal format.

109685 XSI *-c2* 1,6,10,14,49  
109686 COBOL, compact format (columns 1 to 6 omitted).

109687 XSI *-c3* 1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67  
109688 COBOL compact format (columns 1 to 6 omitted), with more tabs than *-c2*.

109689 XSI *-f* 1,7,11,15,19,23  
109690 FORTRAN

109691 XSI *-p* 1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61  
109692 PL/1

109693 XSI *-s* 1,10,55  
109694 SNOBOL

109695 XSI *-u* 1,12,20,44  
109696 Assembler, applicable to some mainframes.

109697        **-T type**        Indicate the type of terminal. If this option is not supplied and the *TERM* variable  
 109698                               is unset or null, an unspecified default terminal type shall be used. The setting of  
 109699                               *type* shall take precedence over the value in *TERM*.

#### 109700 **OPERANDS**

109701        The following operand shall be supported:

109702        *n*[[*sep*[+]*n*]...] A single command line argument that consists of one or more tab-stop values (*n*)  
 109703                               separated by a separator character (*sep*) which is either a <comma> or a <blank>  
 109704                               character. The application shall ensure that the tab-stop values are positive decimal  
 109705                               integers in strictly ascending order. If any tab-stop value (except the first one) is  
 109706                               preceded by a <plus-sign>, it is taken as an increment to be added to the previous  
 109707                               value. For example, the tab lists 1,10,20,30 and "1 10 +10 +10" are considered  
 109708                               to be identical.

#### 109709 **STDIN**

109710        Not used.

#### 109711 **INPUT FILES**

109712        None.

#### 109713 **ENVIRONMENT VARIABLES**

109714        The following environment variables shall affect the execution of *tabs*:

109715        *LANG*        Provide a default value for the internationalization variables that are unset or null.  
 109716                               (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 109717                               variables used to determine the values of locale categories.)

109718        *LC\_ALL*       If set to a non-empty string value, override the values of all the other  
 109719                               internationalization variables.

109720        *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 109721                               characters (for example, single-byte as opposed to multi-byte characters in  
 109722                               arguments).

109723        *LC\_MESSAGES*

109724                               Determine the locale that should be used to affect the format and contents of  
 109725                               diagnostic messages written to standard error.

109726 XSI        *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

109727        *TERM*        Determine the terminal type. If this variable is unset or null, and if the *-T* option is  
 109728                               not specified, an unspecified default terminal type shall be used.

#### 109729 **ASYNCHRONOUS EVENTS**

109730        Default.

#### 109731 **STDOUT**

109732        If standard output is a terminal, the appropriate sequence to clear and set the tab stops may be  
 109733        written to standard output in an unspecified format. If standard output is not a terminal,  
 109734        undefined results occur.

#### 109735 **STDERR**

109736        The standard error shall be used only for diagnostic messages.

#### 109737 **OUTPUT FILES**

109738        None.

109739 **EXTENDED DESCRIPTION**

109740 None.

109741 **EXIT STATUS**

109742 The following exit values shall be returned:

109743 0 Successful completion.

109744 &gt;0 An error occurred.

109745 **CONSEQUENCES OF ERRORS**

109746 Default.

109747 **APPLICATION USAGE**109748 This utility makes use of the terminal's hardware tabs and the *stty tabs* option.

109749 This utility is not recommended for application use.

109750 Some integrated display units might not have escape sequences to set tab stops, but may be set  
109751 by internal system calls. On these terminals, *tabs* works if standard output is directed to the  
109752 terminal; if output is directed to another file, however, *tabs* fails.

109753 **EXAMPLES**

109754 None.

109755 **RATIONALE**

109756 Consideration was given to having the *tput* utility handle all of the functions described in *tabs*.  
109757 However, the separate *tabs* utility was retained because it seems more intuitive to use a  
109758 command named *tabs* than *tput* with a new option. The *tput* utility does not support setting or  
109759 clearing tabs, and no known historical version of *tabs* supports the capability of setting arbitrary  
109760 tab stops.

109761 The System V *tabs* interface is very complex; the version in this volume of POSIX.1-2008 has a  
109762 reduced feature list, but many of the features omitted were restored as part of the XSI option  
109763 even though the supported languages and coding styles are primarily historical.

109764 There was considerable sentiment for specifying only a means of resetting the tabs back to a  
109765 known state—presumably the “standard” of tabs every eight positions. The following features  
109766 were omitted:

- 109767 • Setting tab stops via the first line in a file, using *–file*. Since even the SVID has no  
109768 complete explanation of this feature, it is doubtful that it is in widespread use.

109769 In an early proposal, a *–t tablist* option was added for consistency with *expand*; this was later  
109770 removed when inconsistencies with the historical list of tabs were identified.

109771 Consideration was given to adding a *–p* option that would output the current tab settings so  
109772 that they could be saved and then later restored. This was not accepted because querying the tab  
109773 stops of the terminal is not a capability in historical *terminfo* or *termcap* facilities and might not be  
109774 supported on a wide range of terminals.

109775 **FUTURE DIRECTIONS**

109776 None.

109777 **SEE ALSO**109778 *expand*, *stty*, *tput*, *unexpand*

109779 XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

109780 **CHANGE HISTORY**

109781 First released in Issue 2.

109782 **Issue 6**

109783 This utility is marked as part of the User Portability Utilities option.

109784 The normative text is reworded to avoid use of the term “must” for application requirements.

109785 **Issue 7**109786 The *tabs* utility is removed from the User Portability Utilities option. User Portability Utilities is now an option for interactive utilities.

109788 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

109789 The SYNOPSIS and OPERANDS sections are updated.

109790 **NAME**

109791 tail — copy the last part of a file

109792 **SYNOPSIS**109793 tail [-f] [-c *number*|-n *number*] [*file*]109794 **DESCRIPTION**109795 The *tail* utility shall copy its input file to the standard output beginning at a designated place.

109796 Copying shall begin at the point in the file indicated by the *-c number* or *-n number* options. The  
 109797 option-argument *number* shall be counted in units of lines or bytes, according to the options *-n*  
 109798 and *-c*. Both line and byte counts start from 1.

109799 Tails relative to the end of the file may be saved in an internal buffer, and thus may be limited in  
 109800 length. Such a buffer, if any, shall be no smaller than {LINE\_MAX}\*10 bytes.

109801 **OPTIONS**

109802 The *tail* utility shall conform to XBD Section 12.2 (on page 216), except that '+' may be  
 109803 recognized as an option delimiter as well as '-'.

109804 The following options shall be supported:

109805 *-c number* The application shall ensure that the *number* option-argument is a decimal integer,  
 109806 optionally including a sign. The sign shall affect the location in the file, measured  
 109807 in bytes, to begin the copying:

| Sign        | Copying Starts                         |
|-------------|----------------------------------------|
| +           | Relative to the beginning of the file. |
| -           | Relative to the end of the file.       |
| <i>none</i> | Relative to the end of the file.       |

109812 The application shall ensure that if the sign of the *number* option-argument is '+',  
 109813 the *number* option-argument is a non-zero decimal integer.

109814 The origin for counting shall be 1; that is, *-c +1* represents the first byte of the file,  
 109815 *-c -1* the last.

109816 *-f* If the input file is a regular file or if the *file* operand specifies a FIFO, do not  
 109817 terminate after the last line of the input file has been copied, but read and copy  
 109818 further bytes from the input file when they become available. If no *file* operand is  
 109819 specified and standard input is a pipe or FIFO, the *-f* option shall be ignored. If the  
 109820 input file is not a FIFO, pipe, or regular file, it is unspecified whether or not the *-f*  
 109821 option shall be ignored.

109822 *-n number* This option shall be equivalent to *-c number*, except the starting location in the file  
 109823 shall be measured in lines instead of bytes. The origin for counting shall be 1; that  
 109824 is, *-n +1* represents the first line of the file, *-n -1* the last.

109825 If neither *-c* nor *-n* is specified, *-n 10* shall be assumed.109826 **OPERANDS**

109827 The following operand shall be supported:

109828 *file* A pathname of an input file. If no *file* operand is specified, the standard input shall  
 109829 be used.

**109830 STDIN**

109831 The standard input shall be used if no *file* operand is specified, and shall be used if the *file*  
 109832 operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
 109833 the standard input shall not be used. See the INPUT FILES section.

**109834 INPUT FILES**

109835 If the -c option is specified, the input file can contain arbitrary data; otherwise, the input file  
 109836 shall be a text file.

**109837 ENVIRONMENT VARIABLES**

109838 The following environment variables shall affect the execution of *tail*:

109839 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 109840 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 109841 variables used to determine the values of locale categories.)

109842 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 109843 internationalization variables.

109844 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 109845 characters (for example, single-byte as opposed to multi-byte characters in  
 109846 arguments and input files).

109847 *LC\_MESSAGES*

109848 Determine the locale that should be used to affect the format and contents of  
 109849 diagnostic messages written to standard error.

109850 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

**109851 ASYNCHRONOUS EVENTS**

109852 Default.

**109853 STDOUT**

109854 The designated portion of the input file shall be written to standard output.

**109855 STDERR**

109856 The standard error shall be used only for diagnostic messages.

**109857 OUTPUT FILES**

109858 None.

**109859 EXTENDED DESCRIPTION**

109860 None.

**109861 EXIT STATUS**

109862 The following exit values shall be returned:

109863 0 Successful completion.

109864 >0 An error occurred.

**109865 CONSEQUENCES OF ERRORS**

109866 Default.

**APPLICATION USAGE**

The `-c` option should be used with caution when the input is a text file containing multi-byte characters; it may produce output that does not start on a character boundary.

Although the input file to *tail* can be any type, the results might not be what would be expected on some character special device files or on file types not described by the System Interfaces volume of POSIX.1-2008. Since this volume of POSIX.1-2008 does not specify the block size used when doing input, *tail* need not read all of the data from devices that only perform block transfers.

When using *tail* to process pathnames, and the `-c` option is not specified, it is recommended that `LC_ALL`, or at least `LC_CTYPE` and `LC_COLLATE`, are set to `POSIX` or `C` in the environment, since pathnames can contain byte sequences that do not form valid characters in some locales, in which case the utility's behavior would be undefined. In the `POSIX` locale each byte is a valid single-byte character, and therefore this problem is avoided.

**EXAMPLES**

The `-f` option can be used to monitor the growth of a file that is being written by some other process. For example, the command:

```
tail -f fred
```

prints the last ten lines of the file **fred**, followed by any lines that are appended to **fred** between the time *tail* is initiated and killed. As another example, the command:

```
tail -f -c 15 fred
```

prints the last 15 bytes of the file **fred**, followed by any bytes that are appended to **fred** between the time *tail* is initiated and killed.

**RATIONALE**

This version of *tail* was created to allow conformance to the Utility Syntax Guidelines. The historical `-b` option was omitted because of the general non-portability of block-sized units of text. The `-c` option historically meant "characters", but this volume of POSIX.1-2008 indicates that it means "bytes". This was selected to allow reasonable implementations when multi-byte characters are possible; it was not named `-b` to avoid confusion with the historical `-b`.

The origin of counting both lines and bytes is 1, matching all widespread historical implementations. Hence *tail -n +0* is not conforming usage because it attempts to output line zero; but note that *tail -n 0* does conform, and outputs nothing.

Earlier versions of this standard allowed the following forms in the SYNOPSIS:

```
tail -[number][b|c|l][f] [file]
tail +[number][b|c|l][f] [file]
```

These forms are no longer specified by POSIX.1-2008, but may be present in some implementations.

The restriction on the internal buffer is a compromise between the historical System V implementation of 4096 bytes and the BSD 32768 bytes.

The `-f` option has been implemented as a loop that sleeps for 1 second and copies any bytes that are available. This is sufficient, but if more efficient methods of determining when new data are available are developed, implementations are encouraged to use them.

Historical documentation indicates that *tail* ignores the `-f` option if the input file is a pipe (pipe and FIFO on systems that support FIFOs). On BSD-based systems, this has been true; on System V-based systems, this was true when input was taken from standard input, but it did not ignore



109911 the `-f` flag if a FIFO was named as the *file* operand. Since the `-f` option is not useful on pipes and  
109912 all historical implementations ignore `-f` if no *file* operand is specified and standard input is a  
109913 pipe, this volume of POSIX.1-2008 requires this behavior. However, since the `-f` option is useful  
109914 on a FIFO, this volume of POSIX.1-2008 also requires that if a FIFO is named, the `-f` option shall  
109915 not be ignored. Earlier versions of this standard did not state any requirement for the case where  
109916 no *file* operand is specified and standard input is a FIFO. The standard has been updated to  
109917 reflect current practice which is to treat this case the same as a pipe on standard input. Although  
109918 historical behavior does not ignore the `-f` option for other file types, this is unspecified so that  
109919 implementations are allowed to ignore the `-f` option if it is known that the file cannot be  
109920 extended.

#### 109921 FUTURE DIRECTIONS

109922 None.

#### 109923 SEE ALSO

109924 *head*

109925 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

#### 109926 CHANGE HISTORY

109927 First released in Issue 2.

#### 109928 Issue 6

109929 The obsolescent SYNOPSIS lines and associated text are removed.

109930 The normative text is reworded to avoid use of the term “must” for application requirements.

#### 109931 Issue 7

109932 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying that ‘+’ may be recognized  
109933 as an option delimiter in the OPTIONS section.

109934 Austin Group Interpretation 1003.1-2001 #092 is applied.

109935 Austin Group Interpretation 1003.1-2001 #100 is applied, adding the requirement on applications  
109936 that if the sign of the option-argument *number* is ‘+’, the *number* option-argument is a non-zero  
109937 decimal integer.

109938 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

109939 SD5-XCU-ERN-114 is applied, updating the OPTIONS section (the `-f` option).

109940 SD5-XCU-ERN-149 is applied.

109941 **NAME**

109942 talk — talk to another user

109943 **SYNOPSIS**109944 UP `talk address [terminal]`109945 **DESCRIPTION**109946 The *talk* utility is a two-way, screen-oriented communication program.109947 When first invoked, *talk* shall send a message similar to:

```
109948 Message from <unspecified string>
109949 talk: connection requested by your_address
109950 talk: respond with: talk your_address
```

109951 to the specified *address*. At this point, the recipient of the message can reply by typing:109952 `talk your_address`109953 Once communication is established, the two parties can type simultaneously, with their output  
109954 displayed in separate regions of the screen. Characters shall be processed as follows:

- 109955 • Typing the <alert> character shall alert the recipient's terminal.
- 109956 • Typing <control>-L shall cause the sender's screen regions to be refreshed.
- 109957 • Typing the erase and kill characters shall affect the sender's terminal in the manner  
109958 described by the **termios** interface in XBD [Chapter 11](#) (on page 199).
- 109959 • Typing the interrupt or end-of-file characters shall terminate the local *talk* utility. Once the  
109960 *talk* session has been terminated on one side, the other side of the *talk* session shall be  
109961 notified that the *talk* session has been terminated and shall be able to do nothing except  
109962 exit.
- 109963 • Typing characters from *LC\_CTYPE* classifications **print** or **space** shall cause those  
109964 characters to be sent to the recipient's terminal.
- 109965 • When and only when the *stty iexten* local mode is enabled, the existence and processing of  
109966 additional special control characters and multi-byte or single-byte functions shall be  
109967 implementation-defined.
- 109968 • Typing other non-printable characters shall cause implementation-defined sequences of  
109969 printable characters to be sent to the recipient's terminal.

109970 Permission to be a recipient of a *talk* message can be denied or granted by use of the *mesg* utility.  
109971 However, a user's privilege may further constrain the domain of accessibility of other users'  
109972 terminals. The *talk* utility shall fail when the user lacks appropriate privileges to perform the  
109973 requested action.

109974 Certain block-mode terminals do not have all the capabilities necessary to support the  
109975 simultaneous exchange of messages required for *talk*. When this type of exchange cannot be  
109976 supported on such terminals, the implementation may support an exchange with reduced levels  
109977 of simultaneous interaction or it may report an error describing the terminal-related deficiency.

109978 **OPTIONS**

109979 None.

109980 **OPERANDS**

109981 The following operands shall be supported:

109982 *address* The recipient of the *talk* session. One form of *address* is the *<user name>*, as returned  
 109983 by the *who* utility. Other address formats and how they are handled are  
 109984 unspecified.

109985 *terminal* If the recipient is logged in more than once, the *terminal* argument can be used to  
 109986 indicate the appropriate terminal name. If *terminal* is not specified, the *talk* message  
 109987 shall be displayed on one or more accessible terminals in use by the recipient. The  
 109988 format of *terminal* shall be the same as that returned by the *who* utility.

109989 **STDIN**

109990 Characters read from standard input shall be copied to the recipient's terminal in an unspecified  
 109991 manner. If standard input is not a terminal, *talk* shall write a diagnostic message and exit with a  
 109992 non-zero status.

109993 **INPUT FILES**

109994 None.

109995 **ENVIRONMENT VARIABLES**109996 The following environment variables shall affect the execution of *talk*:

109997 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 109998 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 109999 variables used to determine the values of locale categories.)

110000 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 110001 internationalization variables.

110002 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 110003 characters (for example, single-byte as opposed to multi-byte characters in  
 110004 arguments and input files). If the recipient's locale does not use an *LC\_CTYPE*  
 110005 equivalent to the sender's, the results are undefined.

110006 *LC\_MESSAGES*

110007 Determine the locale that should be used to affect the format and contents of  
 110008 diagnostic messages written to standard error and informative messages written to  
 110009 standard output.

110010 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

110011 *TERM* Determine the name of the invoker's terminal type. If this variable is unset or null,  
 110012 an unspecified default terminal type shall be used.

110013 **ASYNCHRONOUS EVENTS**

110014 When the *talk* utility receives a SIGINT signal, the utility shall terminate and exit with a zero  
 110015 status. It shall take the standard action for all other signals.

110016 **STDOUT**

110017 If standard output is a terminal, characters copied from the recipient's standard input may be  
 110018 written to standard output. Standard output also may be used for diagnostic messages. If  
 110019 standard output is not a terminal, *talk* shall exit with a non-zero status.

110020 **STDERR**

110021 None.

110022 **OUTPUT FILES**

110023 None.

110024 **EXTENDED DESCRIPTION**

110025 None.

110026 **EXIT STATUS**

110027 The following exit values shall be returned:

110028 0 Successful completion.

110029 >0 An error occurred or *talk* was invoked on a terminal incapable of supporting it.110030 **CONSEQUENCES OF ERRORS**

110031 Default.

110032 **APPLICATION USAGE**

110033 Because the handling of non-printable, non-`<space>` characters is tied to the *stty* description of  
110034 *ixten*, implementation extensions within the terminal driver can be accessed. For example,  
110035 some implementations provide line editing functions with certain control character sequences.

110036 **EXAMPLES**

110037 None.

110038 **RATIONALE**

110039 The *write* utility was included in this volume of POSIX.1-2008 since it can be implemented on all  
110040 terminal types. The *talk* utility, which cannot be implemented on certain terminals, was  
110041 considered to be a “better” communications interface. Both of these programs are in widespread  
110042 use on historical implementations. Therefore, both utilities have been specified.

110043 All references to networking abilities (*talking* to a user on another system) were removed as  
110044 being outside the scope of this volume of POSIX.1-2008.

110045 Historical BSD and System V versions of *talk* terminate both of the conversations when either  
110046 user breaks out of the session. This can lead to adverse consequences if a user unwittingly  
110047 continues to enter text that is interpreted by the shell when the other terminates the session.  
110048 Therefore, the version of *talk* specified by this volume of POSIX.1-2008 requires both users to  
110049 terminate their end of the session explicitly.

110050 Only messages sent to the terminal of the invoking user can be internationalized in any way:

- 110051 • The original “Message from `<unspecified string>` ...” message sent to the terminal of the  
110052 recipient cannot be internationalized because the environment of the recipient is as yet  
110053 inaccessible to the *talk* utility. The environment of the invoking party is irrelevant.
- 110054 • Subsequent communication between the two parties cannot be internationalized because  
110055 the two parties may specify different languages in their environment (and non-portable  
110056 characters cannot be mapped from one language to another).
- 110057 • Neither party can be required to communicate in a language other than C and/or the one  
110058 specified by their environment because unavailable terminal hardware support (for  
110059 example, fonts) may be required.

110060 The text in the STDOUT section reflects the usage of the verb “display” in this section; some *talk*  
110061 implementations actually use standard output to write to the terminal, but this volume of  
110062 POSIX.1-2008 does not require that to be the case.

110063 The format of the terminal name is unspecified, but the descriptions of *ps*, *talk*, *who*, and *write*  
110064 require that they all use or accept the same format.

110065 The handling of non-printable characters is partially implementation-defined because the details  
110066 of mapping them to printable sequences is not needed by the user. Historical implementations,  
110067 for security reasons, disallow the transmission of non-printable characters that may send  
110068 commands to the other terminal.

110069 **FUTURE DIRECTIONS**

110070 None.

110071 **SEE ALSO**

110072 *mesg, stty, who, write*

110073 XBD [Chapter 8](#) (on page 173), [Chapter 11](#) (on page 199)

110074 **CHANGE HISTORY**

110075 First released in Issue 4.

110076 **Issue 6**

110077 This utility is marked as part of the User Portability Utilities option.

110078 **NAME**

110079       tee — duplicate standard input

110080 **SYNOPSIS**110081       tee [-ai] [*file*...]110082 **DESCRIPTION**110083       The *tee* utility shall copy standard input to standard output, making a copy in zero or more files.110084       The *tee* utility shall not buffer output.110085       If the **-a** option is not specified, output files shall be written (see [Section 1.1.1.4](#) (on page 2318)).110086 **OPTIONS**110087       The *tee* utility shall conform to XBD [Section 12.2](#) (on page 216).

110088       The following options shall be supported:

110089       **-a**           Append the output to the files.110090       **-i**           Ignore the SIGINT signal.110091 **OPERANDS**

110092       The following operands shall be supported:

110093       *file*           A pathname of an output file. If a *file* operand is '-', it shall refer to a file named  
 110094       -; implementations shall not treat it as meaning standard output. Processing of at  
 110095       least 13 *file* operands shall be supported.

110096 **STDIN**

110097       The standard input can be of any type.

110098 **INPUT FILES**

110099       None.

110100 **ENVIRONMENT VARIABLES**110101       The following environment variables shall affect the execution of *tee*:

110102       **LANG**           Provide a default value for the internationalization variables that are unset or null.  
 110103                       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 110104                       variables used to determine the values of locale categories.)

110105       **LC\_ALL**       If set to a non-empty string value, override the values of all the other  
 110106                       internationalization variables.

110107       **LC\_CTYPE**   Determine the locale for the interpretation of sequences of bytes of text data as  
 110108                       characters (for example, single-byte as opposed to multi-byte characters in  
 110109                       arguments).

110110       **LC\_MESSAGES**

110111                       Determine the locale that should be used to affect the format and contents of  
 110112                       diagnostic messages written to standard error.

110113 **XSI**       **NLSPATH**   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.110114 **ASYNCHRONOUS EVENTS**110115       Default, except that if the **-i** option was specified, SIGINT shall be ignored.110116 **STDOUT**

110117       The standard output shall be a copy of the standard input.

**110118 STDERR**

110119 The standard error shall be used only for diagnostic messages.

**110120 OUTPUT FILES**

110121 If any *file* operands are specified, the standard input shall be copied to each named file.

**110122 EXTENDED DESCRIPTION**

110123 None.

**110124 EXIT STATUS**

110125 The following exit values shall be returned:

110126 0 The standard input was successfully copied to all output files.

110127 >0 An error occurred.

**110128 CONSEQUENCES OF ERRORS**

110129 If a write to any successfully opened *file* operand fails, writes to other successfully opened *file*  
110130 operands and standard output shall continue, but the exit status shall be non-zero. Otherwise,  
110131 the default actions specified in [Section 1.4](#) (on page 2326) apply.

**110132 APPLICATION USAGE**

110133 The *tee* utility is usually used in a pipeline, to make a copy of the output of some utility.

110134 The *file* operand is technically optional, but *tee* is no more useful than *cat* when none is specified.

**110135 EXAMPLES**

110136 Save an unsorted intermediate form of the data in a pipeline:

110137 `... | tee unsorted | sort > sorted`

**110138 RATIONALE**

110139 The buffering requirement means that *tee* is not allowed to use ISO C standard fully buffered or  
110140 line-buffered writes. It does not mean that *tee* has to do 1-byte reads followed by 1-byte writes.

110141 It should be noted that early versions of BSD ignore any invalid options and accept a single `'-'`  
110142 as an alternative to `-i`. They also print a message if unable to open a file:

110143 `"tee: cannot access %s\n", <pathname>`

110144 Historical implementations ignore write errors. This is explicitly not permitted by this volume of  
110145 POSIX.1-2008.

110146 Some historical implementations use `O_APPEND` when providing append mode; others use the  
110147 `lseek()` function to seek to the end-of-file after opening the file without `O_APPEND`. This volume  
110148 of POSIX.1-2008 requires functionality equivalent to using `O_APPEND`; see [Section 1.1.1.4](#) (on  
110149 page 2318).

**110150 FUTURE DIRECTIONS**

110151 None.

**110152 SEE ALSO**

110153 [Chapter 1](#) (on page 2317), *cat*

110154 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

110155 XSH [lseek\(\)](#)

110156 **CHANGE HISTORY**

110157 First released in Issue 2.

110158 **Issue 6**

110159 IEEE PASC Interpretation 1003.2 #168 is applied.

110160 **Issue 7**

110161 Austin Group Interpretation 1003.1-2001 #092 is applied.

110162 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



110163 **NAME**

110164 test — evaluate expression

110165 **SYNOPSIS**110166 test [*expression*]110167 [ [*expression*] ]110168 **DESCRIPTION**

110169 The *test* utility shall evaluate the *expression* and indicate the result of the evaluation by its exit  
 110170 status. An exit status of zero indicates that the expression evaluated as true and an exit status of  
 110171 1 indicates that the expression evaluated as false.

110172 In the second form of the utility, where the utility name used is *[* rather than *test*, the application  
 110173 shall ensure that the closing square bracket is a separate argument. The *test* and *[* utilities may be  
 110174 implemented as a single linked utility which examines the basename of the zeroth command  
 110175 line argument to determine whether to behave as the *test* or *[* variant. Applications using the  
 110176 *exec()* family of functions to execute these utilities shall ensure that the argument passed in *arg0*  
 110177 or *argv[0]* is ' *[* ' when executing the *[* utility and has a basename of "test" when executing the  
 110178 *test* utility.

110179 **OPTIONS**

110180 The *test* utility shall not recognize the "--" argument in the manner specified by Guideline 10 in  
 110181 XBD [Section 12.2](#) (on page 216).

110182 No options shall be supported.

110183 **OPERANDS**

110184 The application shall ensure that all operators and elements of primaries are presented as  
 110185 separate arguments to the *test* utility.

110186 The following primaries can be used to construct *expression*:

110187 **-b *pathname*** True if *pathname* resolves to an existing directory entry for a block special file. False  
 110188 if *pathname* cannot be resolved, or if *pathname* resolves to an existing directory entry  
 110189 for a file that is not a block special file.

110190 **-c *pathname*** True if *pathname* resolves to an existing directory entry for a character special file.  
 110191 False if *pathname* cannot be resolved, or if *pathname* resolves to an existing directory  
 110192 entry for a file that is not a character special file.

110193 **-d *pathname*** True if *pathname* resolves to an existing directory entry for a directory. False if  
 110194 *pathname* cannot be resolved, or if *pathname* resolves to an existing directory entry  
 110195 for a file that is not a directory.

110196 **-e *pathname*** True if *pathname* resolves to an existing directory entry. False if *pathname* cannot be  
 110197 resolved.

110198 **-f *pathname*** True if *pathname* resolves to an existing directory entry for a regular file. False if  
 110199 *pathname* cannot be resolved, or if *pathname* resolves to an existing directory entry  
 110200 for a file that is not a regular file.

110201 **-g *pathname*** True if *pathname* resolves to an existing directory entry for a file that has its set-  
 110202 group-ID flag set. False if *pathname* cannot be resolved, or if *pathname* resolves to an  
 110203 existing directory entry for a file that does not have its set-group-ID flag set.

110204 **-h *pathname*** True if *pathname* resolves to an existing directory entry for a symbolic link. False if  
 110205 *pathname* cannot be resolved, or if *pathname* resolves to an existing directory entry  
 110206 for a file that is not a symbolic link. If the final component of *pathname* is a  
 110207 symbolic link, that symbolic link is not followed.

|        |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
|--------|----------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 110208 | <b>-L</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a symbolic link. False if <i>pathname</i> cannot be resolved, or if <i>pathname</i> resolves to an existing directory entry for a file that is not a symbolic link. If the final component of <i>pathname</i> is a symbolic link, that symbolic link is not followed.                                                                                               |
| 110209 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110210 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110211 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110212 | <b>-n</b> <i>string</i>          | True if the length of <i>string</i> is non-zero; otherwise, false.                                                                                                                                                                                                                                                                                                                                                                      |
| 110213 | <b>-p</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a FIFO. False if <i>pathname</i> cannot be resolved, or if <i>pathname</i> resolves to an existing directory entry for a file that is not a FIFO.                                                                                                                                                                                                                   |
| 110214 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110215 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110216 | <b>-r</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a file for which permission to read from the file will be granted, as defined in <a href="#">Section 1.1.1.4</a> (on page 2318). False if <i>pathname</i> cannot be resolved, or if <i>pathname</i> resolves to an existing directory entry for a file for which permission to read from the file will not be granted.                                              |
| 110217 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110218 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110219 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110220 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110221 | <b>-S</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a socket. False if <i>pathname</i> cannot be resolved, or if <i>pathname</i> resolves to an existing directory entry for a file that is not a socket.                                                                                                                                                                                                               |
| 110222 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110223 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110224 | <b>-s</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a file that has a size greater than zero. False if <i>pathname</i> cannot be resolved, or if <i>pathname</i> resolves to an existing directory entry for a file that does not have a size greater than zero.                                                                                                                                                        |
| 110225 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110226 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110227 | <b>-t</b> <i>file_descriptor</i> | True if file descriptor number <i>file_descriptor</i> is open and is associated with a terminal. False if <i>file_descriptor</i> is not a valid file descriptor number, or if file descriptor number <i>file_descriptor</i> is not open, or if it is open but is not associated with a terminal.                                                                                                                                        |
| 110228 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110229 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110230 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110231 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110232 | <b>-u</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a file that has its set-user-ID flag set. False if <i>pathname</i> cannot be resolved, or if <i>pathname</i> resolves to an existing directory entry for a file that does not have its set-user-ID flag set.                                                                                                                                                        |
| 110233 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110234 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110235 | <b>-w</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a file for which permission to write to the file will be granted, as defined in <a href="#">Section 1.1.1.4</a> (on page 2318). False if <i>pathname</i> cannot be resolved, or if <i>pathname</i> resolves to an existing directory entry for a file for which permission to write to the file will not be granted.                                                |
| 110236 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110237 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110238 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110239 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110240 | <b>-x</b> <i>pathname</i>        | True if <i>pathname</i> resolves to an existing directory entry for a file for which permission to execute the file (or search it, if it is a directory) will be granted, as defined in <a href="#">Section 1.1.1.4</a> (on page 2318). False if <i>pathname</i> cannot be resolved, or if <i>pathname</i> resolves to an existing directory entry for a file for which permission to execute (or search) the file will not be granted. |
| 110241 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110242 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110243 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110244 |                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| 110245 | <b>-z</b> <i>string</i>          | True if the length of string <i>string</i> is zero; otherwise, false.                                                                                                                                                                                                                                                                                                                                                                   |
| 110246 | <i>string</i>                    | True if the string <i>string</i> is not the null string; otherwise, false.                                                                                                                                                                                                                                                                                                                                                              |
| 110247 | <i>s1</i> = <i>s2</i>            | True if the strings <i>s1</i> and <i>s2</i> are identical; otherwise, false.                                                                                                                                                                                                                                                                                                                                                            |
| 110248 | <i>s1</i> != <i>s2</i>           | True if the strings <i>s1</i> and <i>s2</i> are not identical; otherwise, false.                                                                                                                                                                                                                                                                                                                                                        |
| 110249 | <i>n1</i> <b>-eq</b> <i>n2</i>   | True if the integers <i>n1</i> and <i>n2</i> are algebraically equal; otherwise, false.                                                                                                                                                                                                                                                                                                                                                 |

- 110250 *n1 -ne n2* True if the integers *n1* and *n2* are not algebraically equal; otherwise, false.
- 110251 *n1 -gt n2* True if the integer *n1* is algebraically greater than the integer *n2*; otherwise, false.
- 110252 *n1 -ge n2* True if the integer *n1* is algebraically greater than or equal to the integer *n2*;  
110253 otherwise, false.
- 110254 *n1 -lt n2* True if the integer *n1* is algebraically less than the integer *n2*; otherwise, false.
- 110255 *n1 -le n2* True if the integer *n1* is algebraically less than or equal to the integer *n2*; otherwise,  
110256 false.
- 110257 OB XSI *expression1 -a expression2*  
110258 True if both *expression1* and *expression2* are true; otherwise, false. The *-a* binary  
110259 primary is left associative. It has a higher precedence than *-o*.
- 110260 OB XSI *expression1 -o expression2*  
110261 True if either *expression1* or *expression2* is true; otherwise, false. The *-o* binary  
110262 primary is left associative.
- 110263 With the exception of the *-h pathname* and *-L pathname* primaries, if a *pathname* argument is a  
110264 symbolic link, *test* shall evaluate the expression by resolving the symbolic link and using the file  
110265 referenced by the link.
- 110266 These primaries can be combined with the following operators:
- 110267 *! expression* True if *expression* is false. False if *expression* is true.
- 110268 OB XSI *( expression )* True if *expression* is true. False if *expression* is false. The parentheses can be used to  
110269 alter the normal precedence and associativity.
- 110270 The primaries with two elements of the form:
- 110271 *-primary\_operator primary\_operand*
- 110272 are known as *unary primaries*. The primaries with three elements in either of the two forms:
- 110273 *primary\_operand -primary\_operator primary\_operand*  
110274 *primary\_operand primary\_operator primary\_operand*
- 110275 are known as *binary primaries*. Additional implementation-defined operators and  
110276 *primary\_operators* may be provided by implementations. They shall be of the form *-operator*  
110277 where the first character of *operator* is not a digit.
- 110278 The algorithm for determining the precedence of the operators and the return value that shall be  
110279 generated is based on the number of arguments presented to *test*. (However, when using the  
110280 "[...]" form, the <right-square-bracket> final argument shall not be counted in this  
110281 algorithm.)
- 110282 In the following list, \$1, \$2, \$3, and \$4 represent the arguments presented to *test*:
- 110283 0 arguments: Exit false (1).
- 110284 1 argument: Exit true (0) if \$1 is not null; otherwise, exit false.
- 110285 2 arguments:
- If \$1 is '!', exit true if \$2 is null, false if \$2 is not null.
  - If \$1 is a unary primary, exit true if the unary test is true, false if the  
110286 unary test is false.
  - 110287

|        |                              |                                                                                               |
|--------|------------------------------|-----------------------------------------------------------------------------------------------|
| 110288 |                              | • Otherwise, produce unspecified results.                                                     |
| 110289 | 3 arguments:                 | • If \$2 is a binary primary, perform the binary test of \$1 and \$3.                         |
| 110290 |                              | • If \$1 is ' ! ', negate the two-argument test of \$2 and \$3.                               |
| 110291 | OB XSI                       | • If \$1 is ' ( ' and \$3 is ' ) ', perform the unary test of \$2. On systems that            |
| 110292 |                              | do not support the XSI option, the results are unspecified if \$1 is ' ( ' and \$3 is ' ) '.  |
| 110293 |                              |                                                                                               |
| 110294 |                              | • Otherwise, produce unspecified results.                                                     |
| 110295 | 4 arguments:                 | • If \$1 is ' ! ', negate the three-argument test of \$2, \$3, and \$4.                       |
| 110296 | OB XSI                       | • If \$1 is ' ( ' and \$4 is ' ) ', perform the two-argument test of \$2 and \$3.             |
| 110297 |                              | On systems that do not support the XSI option, the results are                                |
| 110298 |                              | unspecified if \$1 is ' ( ' and \$4 is ' ) '.                                                 |
| 110299 |                              | • Otherwise, the results are unspecified.                                                     |
| 110300 | >4 arguments:                | The results are unspecified.                                                                  |
| 110301 | OB XSI                       | On XSI-conformant systems, combinations of primaries and operators shall be                   |
| 110302 |                              | evaluated using the precedence and associativity rules described previously.                  |
| 110303 |                              | In addition, the string comparison binary primaries ' = ' and ' != ' shall have               |
| 110304 |                              | a higher precedence than any unary primary.                                                   |
| 110305 | <b>STDIN</b>                 |                                                                                               |
| 110306 |                              | Not used.                                                                                     |
| 110307 | <b>INPUT FILES</b>           |                                                                                               |
| 110308 |                              | None.                                                                                         |
| 110309 | <b>ENVIRONMENT VARIABLES</b> |                                                                                               |
| 110310 |                              | The following environment variables shall affect the execution of <i>test</i> :               |
| 110311 | <i>LANG</i>                  | Provide a default value for the internationalization variables that are unset or null.        |
| 110312 |                              | (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization |
| 110313 |                              | variables used to determine the values of locale categories.)                                 |
| 110314 | <i>LC_ALL</i>                | If set to a non-empty string value, override the values of all the other                      |
| 110315 |                              | internationalization variables.                                                               |
| 110316 | <i>LC_CTYPE</i>              | Determine the locale for the interpretation of sequences of bytes of text data as             |
| 110317 |                              | characters (for example, single-byte as opposed to multi-byte characters in                   |
| 110318 |                              | arguments).                                                                                   |
| 110319 | <i>LC_MESSAGES</i>           |                                                                                               |
| 110320 |                              | Determine the locale that should be used to affect the format and contents of                 |
| 110321 |                              | diagnostic messages written to standard error.                                                |
| 110322 | XSI <i>NLSPATH</i>           | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .         |
| 110323 | <b>ASYNCHRONOUS EVENTS</b>   |                                                                                               |
| 110324 |                              | Default.                                                                                      |
| 110325 | <b>STDOUT</b>                |                                                                                               |
| 110326 |                              | Not used.                                                                                     |

110327 **STDERR**

110328 The standard error shall be used only for diagnostic messages.

110329 **OUTPUT FILES**

110330 None.

110331 **EXTENDED DESCRIPTION**

110332 None.

110333 **EXIT STATUS**

110334 The following exit values shall be returned:

110335 0 *expression* evaluated to true.

110336 1 *expression* evaluated to false or *expression* was missing.

110337 >1 An error occurred.

110338 **CONSEQUENCES OF ERRORS**

110339 Default.

110340 **APPLICATION USAGE**

110341 The XSI extensions specifying the **-a** and **-o** binary primaries and the **'( ' and ' )'** operators  
 110342 have been marked obsolescent. (Many expressions using them are ambiguously defined by the  
 110343 grammar depending on the specific expressions being evaluated.) Scripts using these  
 110344 expressions should be converted to the forms given below. Even though many implementations  
 110345 will continue to support these obsolescent forms, scripts should be extremely careful when  
 110346 dealing with user-supplied input that could be confused with these and other primaries and  
 110347 operators. Unless the application developer knows all the cases that produce input to the script,  
 110348 invocations like:

110349 `test "$1" -a "$2"`

110350 should be written as:

110351 `test "$1" && test "$2"`

110352 to avoid problems if a user supplied values such as \$1 set to **'!'** and \$2 set to the null string.  
 110353 That is, in cases where maximal portability is of concern, replace:

110354 `test expr1 -a expr2`

110355 with:

110356 `test expr1 && test expr2`

110357 and replace:

110358 `test expr1 -o expr2`

110359 with:

110360 `test expr1 || test expr2`

110361 but note that, in *test*, **-a** has higher precedence than **-o** while **"&&"** and **"||"** have equal  
 110362 precedence in the shell.

110363 Parentheses or braces can be used in the shell command language to effect grouping.

110364 Parentheses must be escaped when using *sh*; for example:

110365 `test \( expr1 -a expr2 \) -o expr3`

110366 This command is not always portable even on XSI-conformant systems depending on the

expressions specified by *expr1*, *expr2*, and *expr3*. The following form can be used instead:

```
( test expr1 && test expr2 ) || test expr3
```

The two commands:

```
test "$1"
test ! "$1"
```

could not be used reliably on some historical systems. Unexpected results would occur if such a *string* expression were used and \$1 expanded to '!', '(', or a known unary primary. Better constructs are:

```
test -n "$1"
test -z "$1"
```

respectively.

Historical systems have also been unreliable given the common construct:

```
test "$response" = "expected string"
```

One of the following is a more reliable form:

```
test "X$response" = "Xexpected string"
test "expected string" = "$response"
```

Note that the second form assumes that *expected string* could not be confused with any unary primary. If *expected string* starts with '-', '(', '!', or even '=', the first form should be used instead. Using the preceding rules without the XSI marked extensions, any of the three comparison forms is reliable, given any input. (However, note that the strings are quoted in all cases.)

Because the string comparison binary primaries, '=' and '!=', have a higher precedence than any unary primary in the greater than 4 argument case, unexpected results can occur if arguments are not properly prepared. For example, in:

```
test -d $1 -o -d $2
```

If \$1 evaluates to a possible directory name of '=', the first three arguments are considered a string comparison, which shall cause a syntax error when the second -d is encountered. One of the following forms prevents this; the second is preferred:

```
test \( -d "$1" \) -o \( -d "$2" \)
test -d "$1" || test -d "$2"
```

Also in the greater than 4 argument case:

```
test "$1" = "bat" -a "$2" = "ball"
```

syntax errors occur if \$1 evaluates to '(' or '!'. One of the following forms prevents this; the third is preferred:

```
test "X$1" = "Xbat" -a "X$2" = "Xball"
test "$1" = "bat" && test "$2" = "ball"
test "X$1" = "Xbat" && test "X$2" = "Xball"
```

Note that none of the following examples are permitted by the syntax described:

```
[-f file]
[-f file ]
[ -f file]
```

+

+

+

+

```

110408 [ -f file +
110409 test -f file ] +
110410 In the first two cases, if a utility named [ -f exists, that utility would be invoked, and not test. In +
110411 the remaining cases, the brackets are mismatched, and the behavior is unspecified. However: +
110412 test ! ] +
110413 does have a defined meaning, and must exit with status 1. Similarly: +
110414 test ] +
110415 must exit with status 0.

```

#### 110416 EXAMPLES

```

110417 1. Exit if there are not two or three arguments (two variations):
110418     if [ $# -ne 2 ] && [ $# -ne 3 ]; then exit 1; fi
110419     if [ $# -lt 2 ] || [ $# -gt 3 ]; then exit 1; fi
110420 2. Perform a mkdir if a directory does not exist:
110421     test ! -d tempdir && mkdir tempdir
110422 3. Wait for a file to become non-readable:
110423     while test -r thefile
110424     do
110425         sleep 30
110426     done
110427     echo "thefile" is no longer readable'
110428 4. Perform a command if the argument is one of three strings (two variations):
110429     if [ "$1" = "pear" ] || [ "$1" = "grape" ] || [ "$1" = "apple" ]
110430     then
110431         command
110432     fi
110433     case "$1" in
110434         pear|grape|apple) command ;;
110435     esac

```

#### 110436 RATIONALE

```

110437 The KornShell-derived conditional command (double bracket [[ ]] was removed from the shell
110438 command language description in an early proposal. Objections were raised that the real
110439 problem is misuse of the test command (!), and putting it into the shell is the wrong way to fix
110440 the problem. Instead, proper documentation and a new shell reserved word (!) are sufficient.
110441 Tests that require multiple test operations can be done at the shell level using individual
110442 invocations of the test command and shell logicals, rather than using the error-prone -o flag of
110443 test.
110444 XSI-conformant systems support more than four arguments.
110445 XSI-conformant systems support the combining of primaries with the following constructs:
110446 expression1 -a expression2
110447 True if both expression1 and expression2 are true.

```



*expression1 -o expression2*

True if at least one of *expression1* and *expression2* are true.

*( expression )*

True if *expression* is true.

In evaluating these more complex combined expressions, the following precedence rules are used:

- The unary primaries have higher precedence than the algebraic binary primaries.
- The unary primaries have lower precedence than the string binary primaries.
- The unary and binary primaries have higher precedence than the unary *string* primary.
- The **!** operator has higher precedence than the **-a** operator, and the **-a** operator has higher precedence than the **-o** operator.
- The **-a** and **-o** operators are left associative.
- The parentheses can be used to alter the normal precedence and associativity.

The BSD and System V versions of **-f** are not the same. The BSD definition was:

**-f file** True if *file* exists and is not a directory.

The SVID version (true if the file exists and is a regular file) was chosen for this volume of POSIX.1-2008 because its use is consistent with the **-b**, **-c**, **-d**, and **-p** operands (*file* exists and is a specific file type).

The **-e** primary, possessing similar functionality to that provided by the C shell, was added because it provides the only way for a shell script to find out if a file exists without trying to open the file. Since implementations are allowed to add additional file types, a portable script cannot use:

```
test -b foo -o -c foo -o -d foo -o -f foo -o -p foo
```

to find out if **foo** is an existing file. On historical BSD systems, the existence of a file could be determined by:

```
test -f foo -o -d foo
```

but there was no easy way to determine that an existing file was a regular file. An early proposal used the KornShell **-a** primary (with the same meaning), but this was changed to **-e** because there were concerns about the high probability of humans confusing the **-a** primary with the **-a** binary operator.

The following options were not included in this volume of POSIX.1-2008, although they are provided by some implementations. These operands should not be used by new implementations for other purposes:

**-k file** True if *file* exists and its sticky bit is set.

**-C file** True if *file* is a contiguous file.

**-V file** True if *file* is a version file.

The following option was not included because it was undocumented in most implementations, has been removed from some implementations (including System V), and the functionality is provided by the shell (see [Section 2.6.2](#) (on page 2344)).



- 110487        **-l string**        The length of the string *string*.
- 110488        The **-b**, **-c**, **-g**, **-p**, **-u**, and **-x** operands are derived from the SVID; historical BSD does not  
110489        provide them. The **-k** operand is derived from System V; historical BSD does not provide it.
- 110490        On historical BSD systems, *test -w directory* always returned false because *test* tried to open the  
110491        directory for writing, which always fails.
- 110492        Some additional primaries newly invented or from the KornShell appeared in an early proposal  
110493        as part of the conditional command ([[]]): *s1 > s2*, *s1 < s2*, *str = pattern*, *str != pattern*, *f1 -nt f2*, *f1*  
110494        *-ot f2*, and *f1 -ef f2*. They were not carried forward into the *test* utility when the conditional  
110495        command was removed from the shell because they have not been included in the *test* utility  
110496        built into historical implementations of the *sh* utility.
- 110497        The **-t file\_descriptor** primary is shown with a mandatory argument because the grammar is  
110498        ambiguous if it can be omitted. Historical implementations have allowed it to be omitted,  
110499        providing a default of 1.
- 110500        It is noted that ' [ ' is not part of the portable filename character set; however, since it is required  
110501        to be encoded by a single byte, and is part of the portable character set, the name of this utility  
110502        forms a character string across all supported locales.
- 110503        **FUTURE DIRECTIONS**
- 110504        None.
- 110505        **SEE ALSO**
- 110506        [Section 1.1.1.4](#) (on page 2318), *find*
- 110507        [XBD Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)
- 110508        **CHANGE HISTORY**
- 110509        First released in Issue 2.
- 110510        **Issue 5**
- 110511        The FUTURE DIRECTIONS section is added.
- 110512        **Issue 6**
- 110513        The **-h** operand is added for symbolic links, and access permission requirements are clarified for  
110514        the **-r**, **-w**, and **-x** operands to align with the IEEE P1003.2b draft standard.
- 110515        The normative text is reworded to avoid use of the term “must” for application requirements.
- 110516        The **-L** and **-S** operands are added for symbolic links and sockets.
- 110517        IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/38 is applied, adding XSI margin  
110518        marking and shading to a line in the OPERANDS section referring to the use of parentheses as  
110519        arguments to the *test* utility.
- 110520        IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/30 is applied, rewording the existence  
110521        primaries for the *test* utility.
- 110522        **Issue 7**
- 110523        Austin Group Interpretation 1003.1-2001 #107 is applied.
- 110524        POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0143 [291] is applied.

110525 **NAME**

110526       time — time a simple command

110527 **SYNOPSIS**110528       time [-p] *utility* [*argument...*]110529 **DESCRIPTION**

110530       The *time* utility shall invoke the utility named by the *utility* operand with arguments supplied as  
 110531       the *argument* operands and write a message to standard error that lists timing statistics for the  
 110532       utility. The message shall include the following information:

- 110533       • The elapsed (real) time between invocation of *utility* and its termination.
- 110534       • The User CPU time, equivalent to the sum of the *tms\_utime* and *tms\_cutime* fields returned  
 110535       by the *times()* function defined in the System Interfaces volume of POSIX.1-2008 for the  
 110536       process in which *utility* is executed.
- 110537       • The System CPU time, equivalent to the sum of the *tms\_stime* and *tms\_cstime* fields  
 110538       returned by the *times()* function for the process in which *utility* is executed.

110539       The precision of the timing shall be no less than the granularity defined for the size of the clock  
 110540       tick unit on the system, but the results shall be reported in terms of standard time units (for  
 110541       example, 0.02 seconds, 00:00:00.02, 1m33.75s, 365.21 seconds), not numbers of clock ticks.

110542       When *time* is used as part of a pipeline, the times reported are unspecified, except when it is the  
 110543       sole command within a grouping command (see [Section 2.9.4.1](#), on page 2361) in that pipeline.  
 110544       For example, the commands on the left are unspecified; those on the right report on utilities **a**  
 110545       and **c**, respectively:

```
110546       time a | b | c       { time a; } | b | c
110547       a | b | time c       a | b | (time c)
```

110548 **OPTIONS**110549       The *time* utility shall conform to XBD [Section 12.2](#) (on page 216).

110550       The following option shall be supported:

110551       **-p**       Write the timing output to standard error in the format shown in the STDERR  
 110552       section.

110553 **OPERANDS**

110554       The following operands shall be supported:

110555       *utility*       The name of a utility that is to be invoked. If the *utility* operand names any of the  
 110556       special built-in utilities in [Section 2.14](#) (on page 2374), the results are undefined.

110557       *argument*   Any string to be supplied as an argument when invoking the utility named by the  
 110558       *utility* operand.

110559 **STDIN**

110560       Not used.

110561 **INPUT FILES**

110562       None.

110563 **ENVIRONMENT VARIABLES**110564       The following environment variables shall affect the execution of *time*:

110565       **LANG**       Provide a default value for the internationalization variables that are unset or null.  
 110566       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 110567       variables used to determine the values of locale categories.)

|        |                             |                                                                                                                                                                           |
|--------|-----------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 110568 | <i>LC_ALL</i>               | If set to a non-empty string value, override the values of all the other internationalization variables.                                                                  |
| 110569 |                             |                                                                                                                                                                           |
| 110570 | <i>LC_CTYPE</i>             | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments). |
| 110571 |                             |                                                                                                                                                                           |
| 110572 |                             |                                                                                                                                                                           |
| 110573 | <i>LC_MESSAGES</i>          |                                                                                                                                                                           |
| 110574 |                             | Determine the locale that should be used to affect the format and contents of diagnostic and informative messages written to standard error.                              |
| 110575 |                             |                                                                                                                                                                           |
| 110576 | <i>LC_NUMERIC</i>           |                                                                                                                                                                           |
| 110577 |                             | Determine the locale for numeric formatting.                                                                                                                              |
| 110578 | XSI <i>NLSPATH</i>          | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                     |
| 110579 | <i>PATH</i>                 | Determine the search path that shall be used to locate the utility to be invoked; see XBD Chapter 8 (on page 173).                                                        |
| 110580 |                             |                                                                                                                                                                           |
| 110581 | <b>ASYNCHRONOUS EVENTS</b>  |                                                                                                                                                                           |
| 110582 |                             | Default.                                                                                                                                                                  |
| 110583 | <b>STDOUT</b>               |                                                                                                                                                                           |
| 110584 |                             | Not used.                                                                                                                                                                 |
| 110585 | <b>STDERR</b>               |                                                                                                                                                                           |
| 110586 |                             | If the <i>utility</i> utility is invoked, the standard error shall be used to write the timing statistics and                                                             |
| 110587 |                             | may be used to write a diagnostic message if the utility terminates abnormally; otherwise, the                                                                            |
| 110588 |                             | standard error shall be used to write diagnostic messages and may also be used to write the                                                                               |
| 110589 |                             | timing statistics.                                                                                                                                                        |
| 110590 |                             | If <b>-p</b> is specified, the following format shall be used for the timing statistics in the POSIX locale:                                                              |
| 110591 |                             | "real %f\nuser %f\nsys %f\n", <i>&lt;real seconds&gt;</i> , <i>&lt;user seconds&gt;</i> ,                                                                                 |
| 110592 |                             | <i>&lt;system seconds&gt;</i>                                                                                                                                             |
| 110593 |                             | where each floating-point number shall be expressed in seconds. The precision used may be less                                                                            |
| 110594 |                             | than the default six digits of %f, but shall be sufficiently precise to accommodate the size of the                                                                       |
| 110595 |                             | clock tick on the system (for example, if there were 60 clock ticks per second, at least two digits                                                                       |
| 110596 |                             | shall follow the radix character). The number of digits following the radix character shall be no                                                                         |
| 110597 |                             | less than one, even if this always results in a trailing zero. The implementation may append                                                                              |
| 110598 |                             | white space and additional information following the format shown here. The implementation                                                                                |
| 110599 |                             | may also prepend a single empty line before the format shown here.                                                                                                        |
| 110600 | <b>OUTPUT FILES</b>         |                                                                                                                                                                           |
| 110601 |                             | None.                                                                                                                                                                     |
| 110602 | <b>EXTENDED DESCRIPTION</b> |                                                                                                                                                                           |
| 110603 |                             | None.                                                                                                                                                                     |
| 110604 | <b>EXIT STATUS</b>          |                                                                                                                                                                           |
| 110605 |                             | If the <i>utility</i> utility is invoked, the exit status of <i>time</i> shall be the exit status of <i>utility</i> ; otherwise,                                          |
| 110606 |                             | the <i>time</i> utility shall exit with one of the following values:                                                                                                      |
| 110607 | 1-125                       | An error occurred in the <i>time</i> utility.                                                                                                                             |
| 110608 | 126                         | The utility specified by <i>utility</i> was found but could not be invoked.                                                                                               |

110609           127    The utility specified by *utility* could not be found.

## 110610 CONSEQUENCES OF ERRORS

110611           Default.

## 110612 APPLICATION USAGE

110613           The *command*, *env*, *nice*, *nohup*, *time*, and *xargs* utilities have been specified to use exit code 127 if  
110614           an error occurs so that applications can distinguish “failure to find a utility” from “invoked  
110615           utility exited with an error indication”. The value 127 was chosen because it is not commonly  
110616           used for other meanings; most utilities use small values for “normal error conditions” and the  
110617           values above 128 can be confused with termination due to receipt of a signal. The value 126 was  
110618           chosen in a similar manner to indicate that the utility could be found, but not invoked. Some  
110619           scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
110620           between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
110621           *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
110622           any other reason.

## 110623 EXAMPLES

110624           It is frequently desirable to apply *time* to pipelines or lists of commands. This can be done by  
110625           placing pipelines and command lists in a single file; this file can then be invoked as a utility, and  
110626           the *time* applies to everything in the file.

110627           Alternatively, the following command can be used to apply *time* to a complex command:

110628           time sh -c 'complex-command-line'

## 110629 RATIONALE

110630           When the *time* utility was originally proposed to be included in the ISO POSIX-2:1993 standard,  
110631           questions were raised about its suitability for inclusion on the grounds that it was not useful for  
110632           conforming applications, specifically:

- 110633           • The underlying CPU definitions from the System Interfaces volume of POSIX.1-2008 are  
110634           vague, so the numeric output could not be compared accurately between systems or even  
110635           between invocations.
- 110636           • The creation of portable benchmark programs was outside the scope this volume of  
110637           POSIX.1-2008.

110638           However, *time* does fit in the scope of user portability. Human judgement can be applied to the  
110639           analysis of the output, and it could be very useful in hands-on debugging of applications or in  
110640           providing subjective measures of system performance. Hence it has been included in this  
110641           volume of POSIX.1-2008.

110642           The default output format has been left unspecified because historical implementations differ  
110643           greatly in their style of depicting this numeric output. The **-p** option was invented to provide  
110644           scripts with a common means of obtaining this information.

110645           In the KornShell, *time* is a shell reserved word that can be used to time an entire pipeline, rather  
110646           than just a simple command. The POSIX definition has been worded to allow this  
110647           implementation. Consideration was given to invalidating this approach because of the historical  
110648           model from the C shell and System V shell. However, since the System V *time* utility historically  
110649           has not produced accurate results in pipeline timing (because the constituent processes are not  
110650           all owned by the same parent process, as allowed by POSIX), it did not seem worthwhile to  
110651           break historical KornShell usage.

110652           The term *utility* is used, rather than *command*, to highlight the fact that shell compound  
110653           commands, pipelines, special built-ins, and so on, cannot be used directly. However, *utility*  
110654           includes user application programs and shell scripts, not just the standard utilities.

110655 **FUTURE DIRECTIONS**

110656 None.

110657 **SEE ALSO**110658 [Chapter 2](#) (on page 2335), *sh*110659 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)110660 XSH *times()*110661 **CHANGE HISTORY**

110662 First released in Issue 2.

110663 **Issue 6**

110664 This utility is marked as part of the User Portability Utilities option.

110665 **Issue 7**110666 The *time* utility is moved from the User Portability Utilities option to the Base. User Portability Utilities is now an option for interactive utilities.

110668 SD5-XCU-ERN-115 is applied, updating the example in the DESCRIPTION.

110669 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0144 [266] is applied.

110670 **NAME**

110671 touch — change file access and modification times

110672 **SYNOPSIS**110673 touch [-acm] [-r *ref\_file*|-t *time*|-d *date\_time*] *file*...110674 **DESCRIPTION**110675 The *touch* utility shall change the last data modification timestamps, the last data access  
110676 timestamps, or both.110677 The time used can be specified by the -t *time* option-argument, the corresponding *time* fields of  
110678 the file referenced by the -r *ref\_file* option-argument, or the -d *date\_time* option-argument, as  
110679 specified in the following sections. If none of these are specified, *touch* shall use the current time.110680 For each *file* operand, *touch* shall perform actions equivalent to the following functions defined  
110681 in the System Interfaces volume of POSIX.1-2008:

- 110682 1. If *file* does not exist:
  - 110683 a. The *creat()* function is called with the following arguments:
    - 110684 — The *file* operand is used as the *path* argument.
    - 110685 — The value of the bitwise-inclusive OR of S\_IRUSR, S\_IWUSR, S\_IRGRP,  
110686 S\_IWGRP, S\_IROTH, and S\_IWOTH is used as the *mode* argument.
  - 110687 b. The *futimens()* function is called with the following arguments:
    - 110688 — The file descriptor opened in step 1a.
    - 110689 — The access time and the modification time, set as described in the OPTIONS  
110690 section, are used as the first and second elements of the *times* array argument,  
110691 respectively.
- 110692 2. If *file* exists, the *utimensat()* function is called with the following arguments:
  - 110693 a. The AT\_FDCWD special value is used as the *fd* argument.
  - 110694 b. The *file* operand is used as the *path* argument.
  - 110695 c. The access time and the modification time, set as described in the OPTIONS  
110696 section, are used as the first and second elements of the *times* array argument,  
110697 respectively.
  - 110698 d. The *flag* argument is set to zero.

110699 **OPTIONS**110700 The *touch* utility shall conform to XBD [Section 12.2](#) (on page 216).

110701 The following options shall be supported:

- 110702 -a Change the access time of *file*. Do not change the modification time unless -m is  
110703 also specified.
- 110704 -c Do not create a specified *file* if it does not exist. Do not write any diagnostic  
110705 messages concerning this condition.
- 110706 -d *date\_time* Use the specified *date\_time* instead of the current time. The option-argument shall  
110707 be a string of the form:  
110708 YYYY-MM-DDThh:mm:ss[.frac][tz]  
110709 or:

110710 `YYYY-MM-DDThh:mm:ss[ ,frac][tz]`

110711 where:

- 110712 • YYYY are at least four decimal digits giving the year.
- 110713 • MM, DD, hh, mm, and SS are as with `-t time`.
- 110714 • T is the time designator, and can be replaced by a single <space>.
- 110715 • [ ,frac] and [ ,frac] are either empty, or a <period> ( ' . ' ) or a <comma>
- 110716 ( ' , ' ) respectively, followed by one or more decimal digits, specifying a
- 110717 fractional second.
- 110718 • [tz] is either empty, signifying local time, or the letter ' Z ', signifying UTC.
- 110719 If [tz] is empty, the resulting time shall be affected by the value of the TZ
- 110720 environment variable.

110721 If the resulting time precedes the Epoch, the behavior is implementation-defined. If

110722 the time cannot be represented as the file's timestamp, *touch* shall exit immediately

110723 with an error status.

110724 **-m** Change the modification time of *file*. Do not change the access time unless **-a** is

110725 also specified.

110726 **-r ref\_file** Use the corresponding time of the file named by the pathname *ref\_file* instead of

110727 the current time.

110728 **-t time** Use the specified *time* instead of the current time. The option-argument shall be a

110729 decimal number of the form:

110730 `[ [CC]YY]MMDDhhmm[ .SS]`

110731 where each two digits represents the following:

|        |    |                                                 |
|--------|----|-------------------------------------------------|
| 110732 | MM | The month of the year [01,12].                  |
| 110733 | DD | The day of the month [01,31].                   |
| 110734 | hh | The hour of the day [00,23].                    |
| 110735 | mm | The minute of the hour [00,59].                 |
| 110736 | CC | The first two digits of the year (the century). |
| 110737 | YY | The second two digits of the year.              |
| 110738 | SS | The second of the minute [00,60].               |

110739 Both CC and YY shall be optional. If neither is given, the current year shall be

110740 assumed. If YY is specified, but CC is not, CC shall be derived as follows:

|        |                  |                    |
|--------|------------------|--------------------|
| 110741 | <b>If YY is:</b> | <b>CC becomes:</b> |
| 110742 | [69,99]          | 19                 |
| 110743 | [00,68]          | 20                 |

110744 **Note:** It is expected that in a future version of this standard the default century inferred

110745 from a 2-digit year will change. (This would apply to all commands accepting a

110746 2-digit year as input.)

110747 The resulting time shall be affected by the value of the TZ environment variable. If

110748 the resulting time value precedes the Epoch, the behavior is implementation-

110749 defined. If the time is out of range for the file's timestamp, *touch* shall exit



110750 immediately with an error status. The range of valid times past the Epoch is  
 110751 implementation-defined, but it shall extend to at least the time 0 hours, 0 minutes,  
 110752 0 seconds, January 1, 2038, Coordinated Universal Time. Some implementations  
 110753 may not be able to represent dates beyond January 18, 2038, because they use  
 110754 **signed int** as a time holder.

110755 The range for *SS* is [00,60] rather than [00,59] because of leap seconds. If *SS* is 60,  
 110756 and the resulting time, as affected by the *TZ* environment variable, does not refer  
 110757 to a leap second, the resulting time shall be one second after a time where *SS* is 59.  
 110758 If *SS* is not given a value, it is assumed to be zero.

110759 If neither the **-a** nor **-m** options were specified, *touch* shall behave as if both the **-a** and **-m**  
 110760 options were specified.

#### 110761 OPERANDS

110762 The following operands shall be supported:

110763 *file* A pathname of a file whose times shall be modified.

#### 110764 STDIN

110765 Not used.

#### 110766 INPUT FILES

110767 None.

#### 110768 ENVIRONMENT VARIABLES

110769 The following environment variables shall affect the execution of *touch*:

110770 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 110771 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 110772 variables used to determine the values of locale categories.)

110773 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 110774 internationalization variables.

110775 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 110776 characters (for example, single-byte as opposed to multi-byte characters in  
 110777 arguments).

110778 *LC\_MESSAGES*

110779 Determine the locale that should be used to affect the format and contents of  
 110780 diagnostic messages written to standard error.

110781 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

110782 *TZ* Determine the timezone to be used for interpreting the *time* option-argument. If *TZ*  
 110783 is unset or null, an unspecified default timezone shall be used.

#### 110784 ASYNCHRONOUS EVENTS

110785 Default.

#### 110786 STDOUT

110787 Not used.

#### 110788 STDERR

110789 The standard error shall be used only for diagnostic messages.



110790 **OUTPUT FILES**

110791 None.

110792 **EXTENDED DESCRIPTION**

110793 None.

110794 **EXIT STATUS**

110795 The following exit values shall be returned:

110796 0 The utility executed successfully and all requested changes were made.

110797 &gt;0 An error occurred.

110798 **CONSEQUENCES OF ERRORS**

110799 Default.

110800 **APPLICATION USAGE**

110801 The interpretation of time is taken to be *seconds since the Epoch* (see XBD [Section 4.16](#), on page  
 110802 113). It should be noted that implementations conforming to the System Interfaces volume of  
 110803 POSIX.1-2008 do not take leap seconds into account when computing seconds since the Epoch.  
 110804 When *SS=60* is used, the resulting time always refers to 1 plus *seconds since the Epoch* for a time  
 110805 when *SS=59*.

110806 Although the *-t time* option-argument specifies values in 1969, the access time and modification  
 110807 time fields are defined in terms of seconds since the Epoch (00:00:00 on 1 January 1970 UTC).  
 110808 Therefore, depending on the value of *TZ* when *touch* is run, there is never more than a few valid  
 110809 hours in 1969 and there need not be any valid times in 1969.

110810 If the *T* time designator is replaced by a <space> for the *-d date\_time* option-argument, the  
 110811 <space> must be quoted to prevent the shell from splitting the argument.

110812 **EXAMPLES**

110813 Create or update a file called **dwc**; the resulting file has both the last data modification and last  
 110814 data access timestamps set to November 12, 2007 at 10:15:30 local time:

110815 `touch -d 2007-11-12T10:15:30 dwc`

110816 Create or update a file called **nick**; the resulting file has both the last data modification and last  
 110817 data access timestamps set to November 12, 2007 at 10:15:30 UTC:

110818 `touch -d 2007-11-12T10:15:30Z nick`

110819 Create or update a file called **gwc**; the resulting file has both the last data modification and last  
 110820 data access timestamps set to November 12, 2007 at 10:15:30 local time with a fractional second  
 110821 timestamp of .002 seconds:

110822 `touch -d 2007-11-12T10:15:30.002 gwc`

110823 Create or update a file called **ajosey**; the resulting file has both the last data modification and  
 110824 last data access timestamps set to November 12, 2007 at 10:15:30 UTC with a fractional second  
 110825 timestamp of .002 seconds:

110826 `touch -d "2007-11-12 10:15:30.002Z" ajosey`

110827 Create or update a file called **cathy**; the resulting file has both the last data modification and last  
 110828 data access timestamps set to November 12, 2007 at 10:15:00 local time:

110829 `touch -t 200711121015 cathy`

110830 Create or update a file called **drepper**; the resulting file has both the last data modification and  
 110831 last data access timestamps set to November 12, 2007 at 10:15:30 local time:

110832 touch -t 200711121015.30 drepper

110833 Create or update a file called **ebb9**; the resulting file has both the last data modification and last  
110834 data access timestamps set to November 12, 2007 at 10:15:30 local time:

110835 touch -t 0711121015.30 ebb9

110836 Create or update a file called **eggert**; the resulting file has the last data access timestamp set to  
110837 the corresponding time of the file named **mark** instead of the current time. If the file exists, the  
110838 last data modification time is not changed:

110839 touch -a -r mark eggert

#### 110840 RATIONALE

110841 The functionality of *touch* is described almost entirely through references to functions in the  
110842 System Interfaces volume of POSIX.1-2008. In this way, there is no duplication of effort required  
110843 for describing such side-effects as the relationship of user IDs to the user database, permissions,  
110844 and so on.

110845 There are some significant differences between the *touch* utility in this volume of POSIX.1-2008  
110846 and those in System V and BSD systems. They are upwards-compatible for historical  
110847 applications from both implementations:

110848 1. In System V, an ambiguity exists when a pathname that is a decimal number leads the  
110849 operands; it is treated as a time value. In BSD, no *time* value is allowed; files may only be  
110850 *touched* to the current time. The **-t** *time* construct solves these problems for future  
110851 conforming applications (note that the **-t** option is not historical practice).

110852 2. The inclusion of the century digits, *CC*, is also new. Note that a ten-digit *time* value is  
110853 treated as if *YY*, and not *CC*, were specified. The caveat about the range of dates  
110854 following the Epoch was included as recognition that some implementations are not able  
110855 to represent dates beyond 18 January 2038 because they use **signed int** as a time holder.

110856 The **-r** option was added because several comments requested this capability. This option was  
110857 named **-f** in an early proposal, but was changed because the **-f** option is used in the BSD  
110858 version of *touch* with a different meaning.

110859 At least one historical implementation of *touch* incremented the exit code if **-c** was specified and  
110860 the file did not exist. This volume of POSIX.1-2008 requires exit status zero if no errors occur.

110861 In previous version of the standard, if at least two operands are specified, and the first operand  
110862 is an eight or ten-digit decimal integer, the first operand was assumed to be a *date\_time* operand.  
110863 This usage was removed in this version of the standard since it had been marked obsolescent  
110864 previously.

110865 The **-d** *date\_time* format is an ISO 8601:2004 standard complete representation of date and time  
110866 extended format with an optional decimal point or <comma> followed by a string of digits  
110867 following the seconds portion to specify fractions of a second. It is not necessary to recognize  
110868 "[+/-]hh:mm" and "[+/-]hh" to specify timezones other than local time and UTC. The *T*  
110869 time designator in the ISO 8601:2004 standard extended format may be replaced by <space>.

#### 110870 FUTURE DIRECTIONS

110871 None.

#### 110872 SEE ALSO

110873 *date*

110874 XBD [Section 4.16](#) (on page 113), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216), [<sys/stat.h>](#)

110875 XSH [creat\(\)](#), [futimens\(\)](#), [time\(\)](#), [utime\(\)](#)

**CHANGE HISTORY**

110876 First released in Issue 2.

**Issue 6**

110879 The obsolescent *date\_time* operand is removed.

110880 The Open Group Corrigendum U027/1 is applied. This extends the range of valid time past the  
110881 Epoch to at least the time 0 hours, 0 minutes, 0 seconds, January 1, 2038, Coordinated Universal  
110882 Time. This is a new requirement on POSIX implementations.

110883 The range for seconds is changed from [00,61] to [00,60] to align with the ISO/IEC 9899:1999  
110884 standard, and to allow for positive leap seconds.

**Issue 7**

110886 Austin Group Interpretation 1003.1-2001 #118 is applied.

110887 Austin Group Interpretation 1003.1-2001 #193 is applied, adding support for subsecond  
110888 timestamps.

110889 SD5-XCU-ERN-45 is applied, adding a new paragraph to the RATIONALE.

110890 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

110891 SD5-XCU-ERN-110 is applied, updating the OPTIONS section.

110892 Changes are made related to support for finegrained timestamps.

110893 **NAME**

110894       tput — change terminal characteristics

110895 **SYNOPSIS**110896       tput [-T *type*] *operand*...110897 **DESCRIPTION**

110898       The *tput* utility shall display terminal-dependent information. The manner in which this  
 110899       information is retrieved is unspecified. The information displayed shall clear the terminal  
 110900       screen, initialize the user's terminal, or reset the user's terminal, depending on the operand  
 110901       given. The exact consequences of displaying this information are unspecified.

110902 **OPTIONS**110903       The *tput* utility shall conform to XBD [Section 12.2](#) (on page 216).

110904       The following option shall be supported:

110905       -T *type*       Indicate the type of terminal. If this option is not supplied and the *TERM* variable  
 110906       is unset or null, an unspecified default terminal type shall be used. The setting of  
 110907       *type* shall take precedence over the value in *TERM*.

110908 **OPERANDS**

110909       The following strings shall be supported as operands by the implementation in the POSIX locale:

110910       **clear**       Display the clear-screen sequence.

110911       **init**       Display the sequence that initializes the user's terminal in an implementation-  
 110912       defined manner.

110913       **reset**       Display the sequence that resets the user's terminal in an implementation-defined  
 110914       manner.

110915       If a terminal does not support any of the operations described by these operands, this shall not  
 110916       be considered an error condition.

110917 **STDIN**

110918       Not used.

110919 **INPUT FILES**

110920       None.

110921 **ENVIRONMENT VARIABLES**110922       The following environment variables shall affect the execution of *tput*:

110923       LANG       Provide a default value for the internationalization variables that are unset or null.  
 110924       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 110925       variables used to determine the values of locale categories.)

110926       LC\_ALL       If set to a non-empty string value, override the values of all the other  
 110927       internationalization variables.

110928       LC\_CTYPE     Determine the locale for the interpretation of sequences of bytes of text data as  
 110929       characters (for example, single-byte as opposed to multi-byte characters in  
 110930       arguments).

110931       LC\_MESSAGES

110932       Determine the locale that should be used to affect the format and contents of  
 110933       diagnostic messages written to standard error.

|        |     |                               |                                                                                                           |
|--------|-----|-------------------------------|-----------------------------------------------------------------------------------------------------------|
| 110934 | XSI | <b>NLSPATH</b>                | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                     |
| 110935 |     | <b>TERM</b>                   | Determine the terminal type. If this variable is unset or null, and if the <b>-T</b> option is            |
| 110936 |     |                               | not specified, an unspecified default terminal type shall be used.                                        |
| 110937 |     | <b>ASYNCHRONOUS EVENTS</b>    |                                                                                                           |
| 110938 |     |                               | Default.                                                                                                  |
| 110939 |     | <b>STDOUT</b>                 |                                                                                                           |
| 110940 |     |                               | If standard output is a terminal device, it may be used for writing the appropriate sequence to           |
| 110941 |     |                               | clear the screen or reset or initialize the terminal. If standard output is not a terminal device,        |
| 110942 |     |                               | undefined results occur.                                                                                  |
| 110943 |     | <b>STDERR</b>                 |                                                                                                           |
| 110944 |     |                               | The standard error shall be used only for diagnostic messages.                                            |
| 110945 |     | <b>OUTPUT FILES</b>           |                                                                                                           |
| 110946 |     |                               | None.                                                                                                     |
| 110947 |     | <b>EXTENDED DESCRIPTION</b>   |                                                                                                           |
| 110948 |     |                               | None.                                                                                                     |
| 110949 |     | <b>EXIT STATUS</b>            |                                                                                                           |
| 110950 |     |                               | The following exit values shall be returned:                                                              |
| 110951 |     | 0                             | The requested string was written successfully.                                                            |
| 110952 |     | 1                             | Unspecified.                                                                                              |
| 110953 |     | 2                             | Usage error.                                                                                              |
| 110954 |     | 3                             | No information is available about the specified terminal type.                                            |
| 110955 |     | 4                             | The specified operand is invalid.                                                                         |
| 110956 |     | >4                            | An error occurred.                                                                                        |
| 110957 |     | <b>CONSEQUENCES OF ERRORS</b> |                                                                                                           |
| 110958 |     |                               | If one of the operands is not available for the terminal, <i>tput</i> continues processing the remaining  |
| 110959 |     |                               | operands.                                                                                                 |
| 110960 |     | <b>APPLICATION USAGE</b>      |                                                                                                           |
| 110961 |     |                               | The difference between resetting and initializing a terminal is left unspecified, as they vary            |
| 110962 |     |                               | greatly based on hardware types. In general, resetting is a more severe action.                           |
| 110963 |     |                               | Some terminals use control characters to perform the stated functions, and on such terminals it           |
| 110964 |     |                               | might make sense to use <i>tput</i> to store the initialization strings in a file or environment variable |
| 110965 |     |                               | for later use. However, because other terminals might rely on system calls to do this work, the           |
| 110966 |     |                               | standard output cannot be used in a portable manner, such as the following non-portable                   |
| 110967 |     |                               | constructs:                                                                                               |
| 110968 |     |                               | ClearVar='tput clear'                                                                                     |
| 110969 |     |                               | tput reset   mailx -s "Wake Up" ddg                                                                       |
| 110970 |     | <b>EXAMPLES</b>               |                                                                                                           |
| 110971 |     | 1.                            | Initialize the terminal according to the type of terminal in the environmental variable                   |
| 110972 |     |                               | <i>TERM</i> . This command can be included in a <b>.profile</b> file.                                     |
| 110973 |     |                               | tput init                                                                                                 |

110974           2.   Reset a 450 terminal.  
110975           tput -T 450 reset

#### 110976 **RATIONALE**

110977           The list of operands was reduced to a minimum for the following reasons:

- 110978           • The only features chosen were those that were likely to be used by human users interacting  
110979           with a terminal.
- 110980           • Specifying the full *terminfo* set was not considered desirable, but the standard developers  
110981           did not want to select among operands.
- 110982           • This volume of POSIX.1-2008 does not attempt to provide applications with sophisticated  
110983           terminal handling capabilities, as that falls outside of its assigned scope and intersects with  
110984           the responsibilities of other standards bodies.

110985           The difference between resetting and initializing a terminal is left unspecified as this varies  
110986           greatly based on hardware types. In general, resetting is a more severe action.

110987           The exit status of 1 is historically reserved for finding out if a Boolean operand is not set.  
110988           Although the operands were reduced to a minimum, the exit status of 1 should still be reserved  
110989           for the Boolean operands, for those sites that wish to support them.

#### 110990 **FUTURE DIRECTIONS**

110991           None.

#### 110992 **SEE ALSO**

110993           [\*stty\*](#), [\*tabs\*](#)

110994           XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

#### 110995 **CHANGE HISTORY**

110996           First released in Issue 4.

#### 110997 **Issue 6**

110998           This utility is marked as part of the User Portability Utilities option.

#### 110999 **Issue 7**

111000           The *tput* utility is moved from the User Portability Utilities option to the Base. User Portability  
111001           Utilities is now an option for interactive utilities.

111002 **NAME**111003 `tr` — translate characters111004 **SYNOPSIS**111005 `tr [-c|-C] [-s] string1 string2`111006 `tr -s [-c|-C] string1`111007 `tr -d [-c|-C] string1`111008 `tr -ds [-c|-C] string1 string2`111009 **DESCRIPTION**

111010 The *tr* utility shall copy the standard input to the standard output with substitution or deletion  
 111011 of selected characters. The options specified and the *string1* and *string2* operands shall control  
 111012 translations that occur while copying characters and single-character collating elements.

111013 **OPTIONS**111014 The *tr* utility shall conform to XBD [Section 12.2](#) (on page 216).

111015 The following options shall be supported:

111016 **-c** Complement the set of values specified by *string1*. See the EXTENDED  
 111017 DESCRIPTION section.

111018 **-C** Complement the set of characters specified by *string1*. See the EXTENDED  
 111019 DESCRIPTION section.

111020 **-d** Delete all occurrences of input characters that are specified by *string1*.

111021 **-s** Replace instances of repeated characters with a single character, as described in the  
 111022 EXTENDED DESCRIPTION section.

111023 **OPERANDS**

111024 The following operands shall be supported:

111025 *string1, string2*

111026 Translation control strings. Each string shall represent a set of characters to be  
 111027 converted into an array of characters used for the translation. For a detailed  
 111028 description of how the strings are interpreted, see the EXTENDED DESCRIPTION  
 111029 section.

111030 **STDIN**

111031 The standard input can be any type of file.

111032 **INPUT FILES**

111033 None.

111034 **ENVIRONMENT VARIABLES**111035 The following environment variables shall affect the execution of *tr*:

111036 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 111037 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 111038 variables used to determine the values of locale categories.)

111039 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 111040 internationalization variables.

111041 **LC\_COLLATE**

111042 Determine the locale for the behavior of range expressions and equivalence classes.

|        |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 111043 | <i>LC_CTYPE</i>                                                                                                                                                                                                                                                                                                                                                                              | Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments) and the behavior of character classes.                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111044 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111045 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111046 | <i>LC_MESSAGES</i>                                                                                                                                                                                                                                                                                                                                                                           |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111047 |                                                                                                                                                                                                                                                                                                                                                                                              | Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 111048 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111049 | XSI <i>NLSPATH</i>                                                                                                                                                                                                                                                                                                                                                                           | Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
| 111050 | <b>ASYNCHRONOUS EVENTS</b>                                                                                                                                                                                                                                                                                                                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111051 | Default.                                                                                                                                                                                                                                                                                                                                                                                     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111052 | <b>STDOUT</b>                                                                                                                                                                                                                                                                                                                                                                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111053 | The <i>tr</i> output shall be identical to the input, with the exception of the specified transformations.                                                                                                                                                                                                                                                                                   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111054 | <b>STDERR</b>                                                                                                                                                                                                                                                                                                                                                                                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111055 | The standard error shall be used only for diagnostic messages.                                                                                                                                                                                                                                                                                                                               |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111056 | <b>OUTPUT FILES</b>                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111057 | None.                                                                                                                                                                                                                                                                                                                                                                                        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111058 | <b>EXTENDED DESCRIPTION</b>                                                                                                                                                                                                                                                                                                                                                                  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111059 | The operands <i>string1</i> and <i>string2</i> (if specified) define two arrays of characters. The constructs in the following list can be used to specify characters or single-character collating elements. If any of the constructs result in multi-character collating elements, <i>tr</i> shall exclude, without a diagnostic, those multi-character elements from the resulting array. |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111060 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111061 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111062 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111063 | <i>character</i>                                                                                                                                                                                                                                                                                                                                                                             | Any character not described by one of the conventions below shall represent itself.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |
| 111064 | <i>\octal</i>                                                                                                                                                                                                                                                                                                                                                                                | Octal sequences can be used to represent characters with specific coded values. An octal sequence shall consist of a <backslash> followed by the longest sequence of one, two, or three-octal-digit characters (01234567). The sequence shall cause the value whose encoding is represented by the one, two, or three-digit octal integer to be placed into the array. Multi-byte characters require multiple, concatenated escape sequences of this type, including the leading <backslash> for each byte.                                                                                                                                    |
| 111065 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111066 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111067 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111068 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111069 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111070 | <i>\character</i>                                                                                                                                                                                                                                                                                                                                                                            | The <backslash>-escape sequences in XBD <a href="#">Table 5-1</a> (on page 121) ('\\', '\a', '\b', '\f', '\n', '\r', '\t', '\v') shall be supported. The results of using any other character, other than an octal digit, following the <backslash> are unspecified. Also, if there is no character following the <backslash>, the results are unspecified.                                                                                                                                                                                                                                                                                    |
| 111071 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111072 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111073 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111074 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111075 | <i>c-c</i>                                                                                                                                                                                                                                                                                                                                                                                   | In the POSIX locale, this construct shall represent the range of collating elements between the range endpoints (as long as neither endpoint is an octal sequence of the form <i>\octal</i> ), inclusive, as defined by the collation sequence. The characters or collating elements in the range shall be placed in the array in ascending collation sequence. If the second endpoint precedes the starting endpoint in the collation sequence, it is unspecified whether the range of collating elements is empty, or this construct is treated as invalid. In locales other than the POSIX locale, this construct has unspecified behavior. |
| 111076 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111077 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111078 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111079 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111080 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111081 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111082 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111083 |                                                                                                                                                                                                                                                                                                                                                                                              | If either or both of the range endpoints are octal sequences of the form <i>\octal</i> , this shall represent the range of specific coded values between the two range endpoints, inclusive.                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| 111084 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| 111085 |                                                                                                                                                                                                                                                                                                                                                                                              |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |



|        |                                             |                                                                                                                     |
|--------|---------------------------------------------|---------------------------------------------------------------------------------------------------------------------|
| 111086 | [:class:]                                   | Represents all characters belonging to the defined character class, as defined by the                               |
| 111087 |                                             | current setting of the <i>LC_CTYPE</i> locale category. The following character class                               |
| 111088 |                                             | names shall be accepted when specified in <i>string1</i> :                                                          |
| 111089 |                                             | <b>alnum    blank    digit    lower    punct    upper</b>                                                           |
| 111090 |                                             | <b>alpha    cntrl    graph    print    space    xdigit</b>                                                          |
| 111091 | XSI                                         | In addition, character class expressions of the form <i>[:name:]</i> shall be recognized in                         |
| 111092 |                                             | those locales where the <i>name</i> keyword has been given a <b>charclass</b> definition in the                     |
| 111093 |                                             | <i>LC_CTYPE</i> category.                                                                                           |
| 111094 |                                             | When both the <b>-d</b> and <b>-s</b> options are specified, any of the character class names                       |
| 111095 |                                             | shall be accepted in <i>string2</i> . Otherwise, only character class names <b>lower</b> or <b>upper</b>            |
| 111096 |                                             | are valid in <i>string2</i> and then only if the corresponding character class ( <b>upper</b> and                   |
| 111097 |                                             | <b>lower</b> , respectively) is specified in the same relative position in <i>string1</i> . Such a                  |
| 111098 |                                             | specification shall be interpreted as a request for case conversion. When <i>[:lower:]</i>                          |
| 111099 |                                             | appears in <i>string1</i> and <i>[:upper:]</i> appears in <i>string2</i> , the arrays shall contain the             |
| 111100 |                                             | characters from the <b>toupper</b> mapping in the <i>LC_CTYPE</i> category of the current                           |
| 111101 |                                             | locale. When <i>[:upper:]</i> appears in <i>string1</i> and <i>[:lower:]</i> appears in <i>string2</i> , the arrays |
| 111102 |                                             | shall contain the characters from the <b>tolower</b> mapping in the <i>LC_CTYPE</i> category                        |
| 111103 |                                             | of the current locale. The first character from each mapping pair shall be in the                                   |
| 111104 |                                             | array for <i>string1</i> and the second character from each mapping pair shall be in the                            |
| 111105 |                                             | array for <i>string2</i> in the same relative position.                                                             |
| 111106 |                                             | Except for case conversion, the characters specified by a character class expression                                |
| 111107 |                                             | shall be placed in the array in an unspecified order.                                                               |
| 111108 |                                             | If the name specified for <i>class</i> does not define a valid character class in the current                       |
| 111109 |                                             | locale, the behavior is undefined.                                                                                  |
| 111110 | [=equiv=]                                   | Represents all characters or collating elements belonging to the same equivalence                                   |
| 111111 |                                             | class as <i>equiv</i> , as defined by the current setting of the <i>LC_COLLATE</i> locale category.                 |
| 111112 |                                             | An equivalence class expression shall be allowed only in <i>string1</i> , or in <i>string2</i> when                 |
| 111113 |                                             | it is being used by the combined <b>-d</b> and <b>-s</b> options. The characters belonging to                       |
| 111114 |                                             | the equivalence class shall be placed in the array in an unspecified order.                                         |
| 111115 | [x*n]                                       | Represents <i>n</i> repeated occurrences of the character <i>x</i> . Because this expression is                     |
| 111116 |                                             | used to map multiple characters to one, it is only valid when it occurs in <i>string2</i> . If                      |
| 111117 |                                             | <i>n</i> is omitted or is zero, it shall be interpreted as large enough to extend the                               |
| 111118 |                                             | <i>string2</i> -based sequence to the length of the <i>string1</i> -based sequence. If <i>n</i> has a               |
| 111119 |                                             | leading zero, it shall be interpreted as an octal value. Otherwise, it shall be                                     |
| 111120 |                                             | interpreted as a decimal value.                                                                                     |
| 111121 | When the <b>-d</b> option is not specified: |                                                                                                                     |
| 111122 | •                                           | If <i>string2</i> is present, each input character found in the array specified by <i>string1</i> shall be          |
| 111123 |                                             | replaced by the character in the same relative position in the array specified by <i>string2</i> . If               |
| 111124 |                                             | the array specified by <i>string2</i> is shorter than the one specified by <i>string1</i> , or if a character       |
| 111125 |                                             | occurs more than once in <i>string1</i> , the results are unspecified.                                              |
| 111126 | •                                           | If the <b>-C</b> option is specified, the complements of the characters specified by <i>string1</i> (the set        |
| 111127 |                                             | of all characters in the current character set, as defined by the current setting of <i>LC_CTYPE</i> ,              |
| 111128 |                                             | except for those actually specified in the <i>string1</i> operand) shall be placed in the array in                  |
| 111129 |                                             | ascending collation sequence, as defined by the current setting of <i>LC_COLLATE</i> .                              |

- 111130 • If the **-c** option is specified, the complement of the values specified by *string1* shall be
- 111131 placed in the array in ascending order by binary value.
- 111132 • Because the order in which characters specified by character class expressions or
- 111133 equivalence class expressions is undefined, such expressions should only be used if the
- 111134 intent is to map several characters into one. An exception is case conversion, as described
- 111135 previously.

111136 When the **-d** option is specified:

- 111137 • Input characters found in the array specified by *string1* shall be deleted.
- 111138 • When the **-C** option is specified with **-d**, all characters except those specified by *string1*
- 111139 shall be deleted. The contents of *string2* are ignored, unless the **-s** option is also specified.
- 111140 • When the **-c** option is specified with **-d**, all values except those specified by *string1* shall
- 111141 be deleted. The contents of *string2* shall be ignored, unless the **-s** option is also specified.
- 111142 • The same string cannot be used for both the **-d** and the **-s** option; when both options are
- 111143 specified, both *string1* (used for deletion) and *string2* (used for squeezing) shall be
- 111144 required.

111145 When the **-s** option is specified, after any deletions or translations have taken place, repeated

111146 sequences of the same character shall be replaced by one occurrence of the same character, if the

111147 character is found in the array specified by the last operand. If the last operand contains a

111148 character class, such as the following example:

111149 `tr -s '[:space:]'`

111150 the last operand's array shall contain all of the characters in that character class. However, in a

111151 case conversion, as described previously, such as:

111152 `tr -s '[:upper:]' '[:lower:]'`

111153 the last operand's array shall contain only those characters defined as the second characters in

111154 each of the **toupper** or **tolower** character pairs, as appropriate.

111155 An empty string used for *string1* or *string2* produces undefined results.

## 111156 EXIT STATUS

111157 The following exit values shall be returned:

- 111158 0 All input was processed successfully.
- 111159 >0 An error occurred.

## 111160 CONSEQUENCES OF ERRORS

111161 Default.

## 111162 APPLICATION USAGE

111163 If necessary, *string1* and *string2* can be quoted to avoid pattern matching by the shell.

111164 If an ordinary digit (representing itself) is to follow an octal sequence, the octal sequence must

111165 use the full three digits to avoid ambiguity.

111166 When *string2* is shorter than *string1*, a difference results between historical System V and BSD

111167 systems. A BSD system pads *string2* with the last character found in *string2*. Thus, it is possible

111168 to do the following:

111169 `tr 0123456789 d`

111170 which would translate all digits to the letter 'd'. Since this area is specifically unspecified in

this volume of POSIX.1-2008, both the BSD and System V behaviors are allowed, but a conforming application cannot rely on the BSD behavior. It would have to code the example in the following way:

```
tr 0123456789 '[d*]'
```

It should be noted that, despite similarities in appearance, the string operands used by *tr* are not regular expressions.

Unlike some historical implementations, this definition of the *tr* utility correctly processes NUL characters in its input stream. NUL characters can be stripped by using:

```
tr -d '\000'
```

## EXAMPLES

1. The following example creates a list of all words in **file1** one per line in **file2**, where a word is taken to be a maximal string of letters.

```
tr -cs "[:alpha:]" "[\n*]" <file1 >file2
```

2. The next example translates all lowercase characters in **file1** to uppercase and writes the results to standard output.

```
tr "[:lower:]" "[:upper:]" <file1
```

3. This example uses an equivalence class to identify accented variants of the base character 'e' in **file1**, which are stripped of diacritical marks and written to **file2**.

```
tr "[=e]" "[e*]" <file1 >file2
```

## RATIONALE

In some early proposals, an explicit option **-n** was added to disable the historical behavior of stripping NUL characters from the input. It was considered that automatically stripping NUL characters from the input was not correct functionality. However, the removal of **-n** in a later proposal does not remove the requirement that *tr* correctly process NUL characters in its input stream. NUL characters can be stripped by using *tr -d '\000'*.

Historical implementations of *tr* differ widely in syntax and behavior. For example, the BSD version has not needed the bracket characters for the repetition sequence. The *tr* utility syntax is based more closely on the System V and XPG3 model while attempting to accommodate historical BSD implementations. In the case of the short *string2* padding, the decision was to unspecify the behavior and preserve System V and XPG3 scripts, which might find difficulty with the BSD method. The assumption was made that BSD users of *tr* have to make accommodations to meet the syntax defined here. Since it is possible to use the repetition sequence to duplicate the desired behavior, whereas there is no simple way to achieve the System V method, this was the correct, if not desirable, approach.

The use of octal values to specify control characters, while having historical precedents, is not portable. The introduction of escape sequences for control characters should provide the necessary portability. It is recognized that this may cause some historical scripts to break.

An early proposal included support for multi-character collating elements. It was pointed out that, while *tr* does employ some syntactical elements from REs, the aim of *tr* is quite different; ranges, for example, do not have a similar meaning ("any of the chars in the range matches", *versus* "translate each character in the range to the output counterpart"). As a result, the previously included support for multi-character collating elements has been removed. What remains are ranges in current collation order (to support, for example, accented characters), character classes, and equivalence classes.

In XPG3 the `[class:]` and `[equiv=]` conventions are shown with double brackets, as in RE syntax. However, *tr* does not implement RE principles; it just borrows part of the syntax. Consequently, `[class:]` and `[equiv=]` should be regarded as syntactical elements on a par with `[x*n]`, which is not an RE bracket expression.

The standard developers will consider changes to *tr* that allow it to translate characters between different character encodings, or they will consider providing a new utility to accomplish this.

On historical System V systems, a range expression requires enclosing square-brackets, such as:

```
tr '[a-z]' '[A-Z]'
```

However, BSD-based systems did not require the brackets, and this convention is used here to avoid breaking large numbers of BSD scripts:

```
tr a-z A-Z
```

The preceding System V script will continue to work because the brackets, treated as regular characters, are translated to themselves. However, any System V script that relied on "a-z" representing the three characters 'a', '-', and 'z' have to be rewritten as "az-".

The ISO POSIX-2:1993 standard had a `-c` option that behaved similarly to the `-C` option, but did not supply functionality equivalent to the `-c` option specified in POSIX.1-2008.

The earlier version also said that octal sequences referred to collating elements and could be placed adjacent to each other to specify multi-byte characters. However, it was noted that this caused ambiguities because *tr* would not be able to tell whether adjacent octal sequences were intending to specify multi-byte characters or multiple single byte characters. POSIX.1-2008 specifies that octal sequences always refer to single byte binary values when used to specify an endpoint of a range of collating elements.

Earlier versions of this standard allowed for implementations with bytes other than eight bits, but this has been modified in this version.

## FUTURE DIRECTIONS

None.

## SEE ALSO

*sed*

XBD [Table 5-1](#) (on page 121), [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

## CHANGE HISTORY

First released in Issue 2.

## Issue 6

The `-C` operand is added, and the description of the `-c` operand is changed to align with the IEEE P1003.2b draft standard.

The normative text is reworded to avoid use of the term "must" for application requirements.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/31 is applied, removing text describing behavior on systems with bytes consisting of more than eight bits.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/32 is applied, updating an example in the EXAMPLES section to avoid using unspecified behavior.

IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/33 is applied, making a correction to the RATIONALE.

111256 **Issue 7**

111257 SD5-XCU-ERN-30 is applied.

111258 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

111259 Austin Group Interpretation 1003.1-2001 #132 is applied, adding rationale to the *\character*  
111260 construct.

111261 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0145 [325] is applied.

111262 **NAME**

111263 true — return true value

111264 **SYNOPSIS**

111265 true

111266 **DESCRIPTION**111267 The *true* utility shall return with exit code zero.111268 **OPTIONS**

111269 None.

111270 **OPERANDS**

111271 None.

111272 **STDIN**

111273 Not used.

111274 **INPUT FILES**

111275 None.

111276 **ENVIRONMENT VARIABLES**

111277 None.

111278 **ASYNCHRONOUS EVENTS**

111279 Default.

111280 **STDOUT**

111281 Not used.

111282 **STDERR**

111283 Not used.

111284 **OUTPUT FILES**

111285 None.

111286 **EXTENDED DESCRIPTION**

111287 None.

111288 **EXIT STATUS**

111289 Zero.

111290 **CONSEQUENCES OF ERRORS**

111291 None.

111292 **APPLICATION USAGE**111293 This utility is typically used in shell scripts, as shown in the EXAMPLES section. The special  
111294 built-in utility `:` is sometimes more efficient than *true*.111295 **EXAMPLES**

111296 This command is executed forever:

111297 while true

111298 do

111299 command

111300 done

111301 **RATIONALE**

111302 The *true* utility has been retained in this volume of POSIX.1-2008, even though the shell special  
111303 built-in : provides similar functionality, because *true* is widely used in historical scripts and is  
111304 less cryptic to novice script readers.

111305 **FUTURE DIRECTIONS**

111306 None.

111307 **SEE ALSO**

111308 [Section 2.9](#) (on page 2355), *false*

111309 **CHANGE HISTORY**

111310 First released in Issue 2.

111311 **Issue 6**

111312 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/39 is applied, replacing the terms “None”  
111313 and “Default” from the STDERR and EXIT STATUS sections, respectively, with terms as defined  
111314 in [Section 1.4](#) (on page 2326).

111315 **NAME**

111316           tsort — topological sort

111317 **SYNOPSIS**111318           tsort [*file*]111319 **DESCRIPTION**111320           The *tsort* utility shall write to standard output a totally ordered list of items consistent with a  
111321           partial ordering of items contained in the input.111322           The application shall ensure that the input consists of pairs of items (non-empty strings)  
111323           separated by <blank> characters. Pairs of different items indicate ordering. Pairs of identical  
111324           items indicate presence, but not ordering.111325 **OPTIONS**

111326           None.

111327 **OPERANDS**

111328           The following operand shall be supported:

111329           *file*           A pathname of a text file to order. If no *file* operand is given, the standard input  
111330           shall be used.111331 **STDIN**111332           The standard input shall be used if no *file* operand is specified, and shall be used if the *file*  
111333           operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
111334           the standard input shall not be used. See the INPUT FILES section.111335 **INPUT FILES**

111336           The input file shall be a text file.

111337 **ENVIRONMENT VARIABLES**111338           The following environment variables shall affect the execution of *tsort*:111339           *LANG*           Provide a default value for the internationalization variables that are unset or null.  
111340           (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
111341           variables used to determine the values of locale categories.)111342           *LC\_ALL*       If set to a non-empty string value, override the values of all the other  
111343           internationalization variables.111344           *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
111345           characters (for example, single-byte as opposed to multi-byte characters in  
111346           arguments and input files).111347           *LC\_MESSAGES*111348           Determine the locale that should be used to affect the format and contents of  
111349           diagnostic messages written to standard error.111350 *XSI*           *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.111351 **ASYNCHRONOUS EVENTS**

111352           Default.

111353 **STDOUT**111354           The standard output shall be a text file consisting of the order list produced from the partially  
111355           ordered input.



**111356 STDERR**

111357 The standard error shall be used only for diagnostic messages.

**111358 OUTPUT FILES**

111359 None.

**111360 EXTENDED DESCRIPTION**

111361 None.

**111362 EXIT STATUS**

111363 The following exit values shall be returned:

111364 0 Successful completion.

111365 >0 An error occurred.

**111366 CONSEQUENCES OF ERRORS**

111367 Default.

**111368 APPLICATION USAGE**

111369 The *LC\_COLLATE* variable need not affect the actions of *tsort*. The output ordering is not  
111370 lexicographic, but depends on the pairs of items given as input.

**111371 EXAMPLES**

111372 The command:

```
111373 tsort <<EOF
111374 a b c c d e
111375 g g
111376 f g e f
111377 h h
111378 EOF
```

111379 produces the output:

```
111380 a
111381 b
111382 c
111383 d
111384 e
111385 f
111386 g
111387 h
```

**111388 RATIONALE**

111389 None.

**111390 FUTURE DIRECTIONS**

111391 None.

**111392 SEE ALSO**

111393 XBD [Chapter 8](#) (on page 173)

**111394 CHANGE HISTORY**

111395 First released in Issue 2.

111396 **Issue 6**

111397 The normative text is reworded to avoid use of the term “must” for application requirements.

111398 **Issue 7**

111399 Austin Group Interpretation 1003.1-2001 #092 is applied.

111400 The *tsort* utility is moved from the XSI option to the Base.

111401 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0146 [241] is applied.

111402 **NAME**111403        `tty` — return user's terminal name111404 **SYNOPSIS**111405        `tty`111406 **DESCRIPTION**

111407        The `tty` utility shall write to the standard output the name of the terminal that is open as  
 111408        standard input. The name that is used shall be equivalent to the string that would be returned by  
 111409        the `ttyname()` function defined in the System Interfaces volume of POSIX.1-2008.

111410 **OPTIONS**111411        The `tty` utility shall conform to XBD [Section 12.2](#) (on page 216).111412 **OPERANDS**

111413        None.

111414 **STDIN**

111415        While no input is read from standard input, standard input shall be examined to determine  
 111416        whether or not it is a terminal, and, if so, to determine the name of the terminal.

111417 **INPUT FILES**

111418        None.

111419 **ENVIRONMENT VARIABLES**111420        The following environment variables shall affect the execution of `tty`:

111421        `LANG`        Provide a default value for the internationalization variables that are unset or null.  
 111422                      (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 111423                      variables used to determine the values of locale categories.)

111424        `LC_ALL`      If set to a non-empty string value, override the values of all the other  
 111425                      internationalization variables.

111426        `LC_CTYPE`   Determine the locale for the interpretation of sequences of bytes of text data as  
 111427                      characters (for example, single-byte as opposed to multi-byte characters in  
 111428                      arguments).

111429        `LC_MESSAGES`

111430                      Determine the locale that should be used to affect the format and contents of  
 111431                      diagnostic messages written to standard error and informative messages written to  
 111432                      standard output.

111433        `XSI`        `NLSPATH`   Determine the location of message catalogs for the processing of `LC_MESSAGES`.

111434 **ASYNCHRONOUS EVENTS**

111435        Default.

111436 **STDOUT**

111437        If standard input is a terminal device, a pathname of the terminal as specified by the `ttyname()`  
 111438        function defined in the System Interfaces volume of POSIX.1-2008 shall be written in the  
 111439        following format:

111440        `"%s\n", <terminal name>`

111441        Otherwise, a message shall be written indicating that standard input is not connected to a  
 111442        terminal. In the POSIX locale, the `tty` utility shall use the format:

111443        `"not a tty\n"`

**111444 STDERR**

111445 The standard error shall be used only for diagnostic messages.

**111446 OUTPUT FILES**

111447 None.

**111448 EXTENDED DESCRIPTION**

111449 None.

**111450 EXIT STATUS**

111451 The following exit values shall be returned:

111452 0 Standard input is a terminal.

111453 1 Standard input is not a terminal.

111454 >1 An error occurred.

**111455 CONSEQUENCES OF ERRORS**

111456 Default.

**111457 APPLICATION USAGE**

111458 This utility checks the status of the file open as standard input against that of an  
111459 implementation-defined set of files. It is possible that no match can be found, or that the match  
111460 found need not be the same file as that which was opened for standard input (although they are  
111461 the same device).

**111462 EXAMPLES**

111463 None.

**111464 RATIONALE**

111465 None.

**111466 FUTURE DIRECTIONS**

111467 None.

**111468 SEE ALSO**

111469 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

111470 XSH [isatty\(\)](#), [ttyname\(\)](#)

**111471 CHANGE HISTORY**

111472 First released in Issue 2.

**111473 Issue 5**

111474 The SYNOPSIS is changed to indicate two forms of the command, with the second form marked  
111475 as obsolete. This is a clarification and does not change the functionality published in previous  
111476 issues.

**111477 Issue 6**

111478 The obsolescent `-s` option is removed.

111479 **NAME**

111480           type — write a description of command type

111481 **SYNOPSIS**

111482 XSI        type name...

111483 **DESCRIPTION**111484           The *type* utility shall indicate how each argument would be interpreted if used as a command  
111485           name.111486 **OPTIONS**

111487           None.

111488 **OPERANDS**

111489           The following operand shall be supported:

111490        name        A name to be interpreted.

111491 **STDIN**

111492           Not used.

111493 **INPUT FILES**

111494           None.

111495 **ENVIRONMENT VARIABLES**111496           The following environment variables shall affect the execution of *type*:111497        LANG        Provide a default value for the internationalization variables that are unset or null.  
111498                      (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
111499                      variables used to determine the values of locale categories.)111500        LC\_ALL       If set to a non-empty string value, override the values of all the other  
111501                      internationalization variables.111502        LC\_CTYPE     Determine the locale for the interpretation of sequences of bytes of text data as  
111503                      characters (for example, single-byte as opposed to multi-byte characters in  
111504                      arguments).111505        LC\_MESSAGES           Determine the locale that should be used to affect the format and contents of  
111506                                diagnostic messages written to standard error.  
111507111508        NLSPATH       Determine the location of message catalogs for the processing of *LC\_MESSAGES*.111509        PATH         Determine the location of *name*, as described in XBD [Chapter 8](#) (on page 173).111510 **ASYNCHRONOUS EVENTS**

111511           Default.

111512 **STDOUT**111513           The standard output of *type* contains information about each operand in an unspecified format.111514           The information provided typically identifies the operand as a shell built-in, function, alias, or  
111515           keyword, and where applicable, may display the operand's pathname.111516 **STDERR**

111517           The standard error shall be used only for diagnostic messages.

**111518 OUTPUT FILES**

111519       None.

**111520 EXTENDED DESCRIPTION**

111521       None.

**111522 EXIT STATUS**

111523       The following exit values shall be returned:

111524       0   Successful completion.

111525       >0   An error occurred.

**111526 CONSEQUENCES OF ERRORS**

111527       Default.

**111528 APPLICATION USAGE**

111529       Since *type* must be aware of the contents of the current shell execution environment (such as the  
111530       lists of commands, functions, and built-ins processed by *hash*), it is always provided as a shell  
111531       regular built-in. If it is called in a separate utility execution environment, such as one of the  
111532       following:

111533       nohup type writer

111534       find . -type f | xargs type

111535       it might not produce accurate results.

**111536 EXAMPLES**

111537       None.

**111538 RATIONALE**

111539       None.

**111540 FUTURE DIRECTIONS**

111541       None.

**111542 SEE ALSO**

111543       *command*, *hash*

111544       XBD Chapter 8 (on page 173)

**111545 CHANGE HISTORY**

111546       First released in Issue 2.

111547 **NAME**

111548           ulimit — set or report file size limit

111549 **SYNOPSIS**111550 XSI        ulimit [-f] [*blocks*]111551 **DESCRIPTION**

111552       The *ulimit* utility shall set or report the file-size writing limit imposed on files written by the  
 111553       shell and its child processes (files of any size may be read). Only a process with appropriate  
 111554       privileges can increase the limit.

111555 **OPTIONS**111556       The *ulimit* utility shall conform to XBD [Section 12.2](#) (on page 216).

111557       The following option shall be supported:

111558       -f           Set (or report, if no *blocks* operand is present), the file size limit in blocks. The -f  
 111559       option shall also be the default case.

111560 **OPERANDS**

111561       The following operand shall be supported:

111562       *blocks*       The number of 512-byte blocks to use as the new file size limit.111563 **STDIN**

111564       Not used.

111565 **INPUT FILES**

111566       None.

111567 **ENVIRONMENT VARIABLES**111568       The following environment variables shall affect the execution of *ulimit*:

111569       LANG        Provide a default value for the internationalization variables that are unset or null.  
 111570                   (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 111571                   variables used to determine the values of locale categories.)

111572       LC\_ALL      If set to a non-empty string value, override the values of all the other  
 111573                   internationalization variables.

111574       LC\_CTYPE    Determine the locale for the interpretation of sequences of bytes of text data as  
 111575                   characters (for example, single-byte as opposed to multi-byte characters in  
 111576                   arguments).

111577       LC\_MESSAGES

111578                   Determine the locale that should be used to affect the format and contents of  
 111579                   diagnostic messages written to standard error.

111580       NLSPATH     Determine the location of message catalogs for the processing of LC\_MESSAGES.

111581 **ASYNCHRONOUS EVENTS**

111582       Default.

111583 **STDOUT**

111584       The standard output shall be used when no *blocks* operand is present. If the current number of  
 111585       blocks is limited, the number of blocks in the current limit shall be written in the following  
 111586       format:

111587       "%d\n", &lt;number of 512-byte blocks&gt;

111588       If there is no current limit on the number of blocks, in the POSIX locale the following format

111589           shall be used:

111590           "unlimited\n"

#### 111591 **STDERR**

111592           The standard error shall be used only for diagnostic messages.

#### 111593 **OUTPUT FILES**

111594           None.

#### 111595 **EXTENDED DESCRIPTION**

111596           None.

#### 111597 **EXIT STATUS**

111598           The following exit values shall be returned:

111599           0   Successful completion.

111600           >0  A request for a higher limit was rejected or an error occurred.

#### 111601 **CONSEQUENCES OF ERRORS**

111602           Default.

#### 111603 **APPLICATION USAGE**

111604           Since *ulimit* affects the current shell execution environment, it is always provided as a shell regular built-in. If it is called in a separate utility execution environment, such as one of the following:

111607           nohup ulimit -f 10000

111608           env ulimit 10000

111609           it does not affect the file size limit of the caller's environment.

111610           Once a limit has been decreased by a process, it cannot be increased (unless appropriate privileges are involved), even back to the original system limit.

#### 111612 **EXAMPLES**

111613           Set the file size limit to 51 200 bytes:

111614           ulimit -f 100

#### 111615 **RATIONALE**

111616           None.

#### 111617 **FUTURE DIRECTIONS**

111618           None.

#### 111619 **SEE ALSO**

111620           XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

111621           XSH [ulimit\(\)](#)

#### 111622 **CHANGE HISTORY**

111623           First released in Issue 2.

#### 111624 **Issue 7**

111625           SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



111626 **NAME**

111627       umask — get or set the file mode creation mask

111628 **SYNOPSIS**111629       umask [-S] [*mask*]111630 **DESCRIPTION**

111631       The *umask* utility shall set the file mode creation mask of the current shell execution environment  
 111632       (see [Section 2.12](#), on page 2371) to the value specified by the *mask* operand. This mask shall affect  
 111633       the initial value of the file permission bits of subsequently created files. If *umask* is called in a  
 111634       subshell or separate utility execution environment, such as one of the following:

```
111635       (umask 002)
111636       nohup umask ...
111637       find . -exec umask ... \;
```

111638       it shall not affect the file mode creation mask of the caller's environment.

111639       If the *mask* operand is not specified, the *umask* utility shall write to standard output the value of  
 111640       the file mode creation mask of the invoking process.

111641 **OPTIONS**111642       The *umask* utility shall conform to XBD [Section 12.2](#) (on page 216).

111643       The following option shall be supported:

111644       -S       Produce symbolic output.

111645       The default output style is unspecified, but shall be recognized on a subsequent invocation of  
 111646       *umask* on the same system as a *mask* operand to restore the previous file mode creation mask.

111647 **OPERANDS**

111648       The following operand shall be supported:

111649       *mask*       A string specifying the new file mode creation mask. The string is treated in the  
 111650       same way as the *mode* operand described in the EXTENDED DESCRIPTION  
 111651       section for *chmod*.

111652       For a *symbolic\_mode* value, the new value of the file mode creation mask shall be  
 111653       the logical complement of the file permission bits portion of the file mode specified  
 111654       by the *symbolic\_mode* string.

111655       In a *symbolic\_mode* value, the permissions *op* characters '+' and '-' shall be  
 111656       interpreted relative to the current file mode creation mask; '+' shall cause the bits  
 111657       for the indicated permissions to be cleared in the mask; '-' shall cause the bits for  
 111658       the indicated permissions to be set in the mask.

111659       The interpretation of *mode* values that specify file mode bits other than the file  
 111660       permission bits is unspecified.

111661       In the octal integer form of *mode*, the specified bits are set in the file mode creation  
 111662       mask.

111663       The file mode creation mask shall be set to the resulting numeric value.

111664       The default output of a prior invocation of *umask* on the same system with no  
 111665       operand also shall be recognized as a *mask* operand.

111666 **STDIN**

111667 Not used.

111668 **INPUT FILES**

111669 None.

111670 **ENVIRONMENT VARIABLES**111671 The following environment variables shall affect the execution of *umask*:

111672 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 111673 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 111674 variables used to determine the values of locale categories.)

111675 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 111676 internationalization variables.

111677 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 111678 characters (for example, single-byte as opposed to multi-byte characters in  
 111679 arguments).

111680 *LC\_MESSAGES*

111681 Determine the locale that should be used to affect the format and contents of  
 111682 diagnostic messages written to standard error.

111683 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

111684 **ASYNCHRONOUS EVENTS**

111685 Default.

111686 **STDOUT**

111687 When the *mask* operand is not specified, the *umask* utility shall write a message to standard  
 111688 output that can later be used as a *umask mask* operand.

111689 If *-S* is specified, the message shall be in the following format:

111690 "u=%s,g=%s,o=%s\n", <owner permissions>, <group permissions>,  
 111691 <other permissions>

111692 where the three values shall be combinations of letters from the set {*r*, *w*, *x*}; the presence of a  
 111693 letter shall indicate that the corresponding bit is clear in the file mode creation mask.

111694 If a *mask* operand is specified, there shall be no output written to standard output.111695 **STDERR**

111696 The standard error shall be used only for diagnostic messages.

111697 **OUTPUT FILES**

111698 None.

111699 **EXTENDED DESCRIPTION**

111700 None.

111701 **EXIT STATUS**

111702 The following exit values shall be returned:

111703 0 The file mode creation mask was successfully changed, or no *mask* operand was supplied.

111704 &gt;0 An error occurred.

111705 **CONSEQUENCES OF ERRORS**

111706 Default.

111707 **APPLICATION USAGE**

111708 Since *umask* affects the current shell execution environment, it is generally provided as a shell  
 111709 regular built-in.

111710 In contrast to the negative permission logic provided by the file mode creation mask and the  
 111711 octal number form of the *mask* argument, the symbolic form of the *mask* argument specifies those  
 111712 permissions that are left alone.

111713 **EXAMPLES**

111714 Either of the commands:

111715 `umask a=rX,ug+w`111716 `umask 002`

111717 sets the mode mask so that subsequently created files have their S\_IWOTH bit cleared.

111718 After setting the mode mask with either of the above commands, the *umask* command can be  
 111719 used to write out the current value of the mode mask:

111720 `$ umask`111721 `0002`

111722 (The output format is unspecified, but historical implementations use the octal integer mode  
 111723 format.)

111724 `$ umask -S`111725 `u=rwx,g=rwx,o=rX`

111726 Either of these outputs can be used as the mask operand to a subsequent invocation of the *umask*  
 111727 utility.

111728 Assuming the mode mask is set as above, the command:

111729 `umask g-w`

111730 sets the mode mask so that subsequently created files have their S\_IWGRP and S\_IWOTH bits  
 111731 cleared.

111732 The command:

111733 `umask -- -w`

111734 sets the mode mask so that subsequently created files have all their write bits cleared. Note that  
 111735 *mask* operands `-r`, `-w`, `-x` or anything beginning with a <hyphen-minus>, must be preceded by  
 111736 `--` to keep it from being interpreted as an option.

111737 **RATIONALE**

111738 Since *umask* affects the current shell execution environment, it is generally provided as a shell  
 111739 regular built-in. If it is called in a subshell or separate utility execution environment, such as one  
 111740 of the following:

111741 `(umask 002)`111742 `nohup umask ...`111743 `find . -exec umask ... \;`

111744 it does not affect the file mode creation mask of the environment of the caller.

111745 The description of the historical utility was modified to allow it to use the symbolic modes of

111746 *chmod*. The `-s` option used in early proposals was changed to `-S` because `-s` could be confused  
111747 with a *symbolic\_mode* form of mask referring to the `S_ISUID` and `S_ISGID` bits.

111748 The default output style is unspecified to permit implementors to provide migration to the new  
111749 symbolic style at the time most appropriate to their users. A `-o` flag to force octal mode output  
111750 was omitted because the octal mode may not be sufficient to specify all of the information that  
111751 may be present in the file mode creation mask when more secure file access permission checks  
111752 are implemented.

111753 It has been suggested that trusted systems developers might appreciate ameliorating the  
111754 requirement that the mode mask “affects” the file access permissions, since it seems access  
111755 control lists might replace the mode mask to some degree. The wording has been changed to say  
111756 that it affects the file permission bits, and it leaves the details of the behavior of how they affect  
111757 the file access permissions to the description in the System Interfaces volume of POSIX.1-2008.

#### 111758 FUTURE DIRECTIONS

111759 None.

#### 111760 SEE ALSO

111761 [Chapter 2](#) (on page 2335), *chmod*

111762 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

111763 XSH *umask()*

#### 111764 CHANGE HISTORY

111765 First released in Issue 2.

#### 111766 Issue 6

111767 The following new requirements on POSIX implementations derive from alignment with the  
111768 Single UNIX Specification:

- 111769 • The octal mode is supported.

111770 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/34 is applied, making a correction to the  
111771 RATIONALE.

#### 111772 Issue 7

111773 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

111774 **NAME**

111775 unalias — remove alias definitions

111776 **SYNOPSIS**111777 unalias *alias-name*...

111778 unalias -a

111779 **DESCRIPTION**

111780 The *unalias* utility shall remove the definition for each alias name specified. See [Section 2.3.1](#) (on  
 111781 page 2338). The aliases shall be removed from the current shell execution environment; see  
 111782 [Section 2.12](#) (on page 2371).

111783 **OPTIONS**111784 The *unalias* utility shall conform to XBD [Section 12.2](#) (on page 216).

111785 The following option shall be supported:

111786 -a Remove all alias definitions from the current shell execution environment.

111787 **OPERANDS**

111788 The following operand shall be supported:

111789 *alias-name* The name of an alias to be removed.111790 **STDIN**

111791 Not used.

111792 **INPUT FILES**

111793 None.

111794 **ENVIRONMENT VARIABLES**111795 The following environment variables shall affect the execution of *unalias*:

111796 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 111797 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 111798 variables used to determine the values of locale categories.)

111799 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 111800 internationalization variables.

111801 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 111802 characters (for example, single-byte as opposed to multi-byte characters in  
 111803 arguments).

111804 *LC\_MESSAGES*

111805 Determine the locale that should be used to affect the format and contents of  
 111806 diagnostic messages written to standard error.

111807 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.111808 **ASYNCHRONOUS EVENTS**

111809 Default.

111810 **STDOUT**

111811 Not used.

111812 **STDERR**

111813 The standard error shall be used only for diagnostic messages.

**111814 OUTPUT FILES**

111815 None.

**111816 EXTENDED DESCRIPTION**

111817 None.

**111818 EXIT STATUS**

111819 The following exit values shall be returned:

111820 0 Successful completion.

111821 >0 One of the *alias-name* operands specified did not represent a valid alias definition, or an  
111822 error occurred.

**111823 CONSEQUENCES OF ERRORS**

111824 Default.

**111825 APPLICATION USAGE**

111826 Since *unalias* affects the current shell execution environment, it is generally provided as a shell  
111827 regular built-in.

**111828 EXAMPLES**

111829 None.

**111830 RATIONALE**

111831 The *unalias* description is based on that from historical KornShell implementations. Known  
111832 differences exist between that and the C shell. The KornShell version was adopted to be  
111833 consistent with all the other KornShell features in this volume of POSIX.1-2008, such as  
111834 command line editing.

111835 The *-a* option is the equivalent of the *unalias \** form of the C shell and is provided to address  
111836 security concerns about unknown aliases entering the environment of a user (or application)  
111837 through the allowable implementation-defined predefined alias route or as a result of an *ENV*  
111838 file. (Although *unalias* could be used to simplify the “secure” shell script shown in the *command*  
111839 rationale, it does not obviate the need to quote all command names. An initial call to *unalias -a*  
111840 would have to be quoted in case there was an alias for *unalias*.)

**111841 FUTURE DIRECTIONS**

111842 None.

**111843 SEE ALSO**

111844 [Chapter 2](#) (on page 2335), *alias*

111845 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

**111846 CHANGE HISTORY**

111847 First released in Issue 4.

**111848 Issue 6**

111849 This utility is marked as part of the User Portability Utilities option.

**111850 Issue 7**

111851 The *unalias* utility is moved from the User Portability Utilities option to the Base. User  
111852 Portability Utilities is now an option for interactive utilities.

**111853 NAME**

111854        `uname` — return system name

**111855 SYNOPSIS**

111856        `uname [-amnrsv]`

**111857 DESCRIPTION**

111858        By default, the *uname* utility shall write the operating system name to standard output. When  
111859        options are specified, symbols representing one or more system characteristics shall be written to  
111860        the standard output. The format and contents of the symbols are implementation-defined. On  
111861        systems conforming to the System Interfaces volume of POSIX.1-2008, the symbols written shall  
111862        be those supported by the *uname()* function as defined in the System Interfaces volume of  
111863        POSIX.1-2008.

**111864 OPTIONS**

111865        The *uname* utility shall conform to XBD [Section 12.2](#) (on page 216).

111866        The following options shall be supported:

- |        |           |                                                                                |
|--------|-----------|--------------------------------------------------------------------------------|
| 111867 | <b>-a</b> | Behave as though all of the options <b>-mnrsv</b> were specified.              |
| 111868 | <b>-m</b> | Write the name of the hardware type on which the system is running to standard |
| 111869 |           | output.                                                                        |
| 111870 | <b>-n</b> | Write the name of this node within an implementation-defined communications    |
| 111871 |           | network.                                                                       |
| 111872 | <b>-r</b> | Write the current release level of the operating system implementation.        |
| 111873 | <b>-s</b> | Write the name of the implementation of the operating system.                  |
| 111874 | <b>-v</b> | Write the current version level of this release of the operating system        |
| 111875 |           | implementation.                                                                |

111876        If no options are specified, the *uname* utility shall write the operating system name, as if the **-s**  
111877        option had been specified.

**111878 OPERANDS**

111879        None.

**111880 STDIN**

111881        Not used.

**111882 INPUT FILES**

111883        None.

**111884 ENVIRONMENT VARIABLES**

111885        The following environment variables shall affect the execution of *uname*:

- |        |                 |                                                                                                                                                                                                                                                                                      |
|--------|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 111886 | <b>LANG</b>     | Provide a default value for the internationalization variables that are unset or null.<br>111887        (See XBD <a href="#">Section 8.2</a> (on page 174) for the precedence of internationalization<br>111888        variables used to determine the values of locale categories.) |
| 111889 | <b>LC_ALL</b>   | If set to a non-empty string value, override the values of all the other<br>111890        internationalization variables.                                                                                                                                                            |
| 111891 | <b>LC_CTYPE</b> | Determine the locale for the interpretation of sequences of bytes of text data as<br>111892        characters (for example, single-byte as opposed to multi-byte characters in<br>111893        arguments).                                                                          |

111894 **LC\_MESSAGES**

111895 Determine the locale that should be used to affect the format and contents of  
111896 diagnostic messages written to standard error.

111897 XSI **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

111898 **ASYNCHRONOUS EVENTS**

111899 Default.

111900 **STDOUT**

111901 By default, the output shall be a single line of the following form:

111902 "%s\n", <sysname>

111903 If the **-a** option is specified, the output shall be a single line of the following form:

111904 "%s %s %s %s %s\n", <sysname>, <nodename>, <release>,  
111905 <version>, <machine>

111906 Additional implementation-defined symbols may be written; all such symbols shall be written at  
111907 the end of the line of output before the <newline>.

111908 If options are specified to select different combinations of the symbols, only those symbols shall  
111909 be written, in the order shown above for the **-a** option. If a symbol is not selected for writing, its  
111910 corresponding trailing <blank> characters also shall not be written.

111911 **STDERR**

111912 The standard error shall be used only for diagnostic messages.

111913 **OUTPUT FILES**

111914 None.

111915 **EXTENDED DESCRIPTION**

111916 None.

111917 **EXIT STATUS**

111918 The following exit values shall be returned:

111919 0 The requested information was successfully written.

111920 >0 An error occurred.

111921 **CONSEQUENCES OF ERRORS**

111922 Default.

111923 **APPLICATION USAGE**

111924 Note that any of the symbols could include embedded <space> characters, which may affect  
111925 parsing algorithms if multiple options are selected for output.

111926 The node name is typically a name that the system uses to identify itself for inter-system  
111927 communication addressing.

111928 **EXAMPLES**

111929 The following command:

111930 `uname -sr`

111931 writes the operating system name and release level, separated by one or more <blank>  
111932 characters.



111933 **RATIONALE**

111934 It was suggested that this utility cannot be used portably since the format of the symbols is  
111935 implementation-defined. The POSIX.1 working group could not achieve consensus on defining  
111936 these formats in the underlying *uname()* function, and there was no expectation that this volume  
111937 of POSIX.1-2008 would be any more successful. Some applications may still find this historical  
111938 utility of value. For example, the symbols could be used for system log entries or for comparison  
111939 with operator or user input.

111940 **FUTURE DIRECTIONS**

111941 None.

111942 **SEE ALSO**

111943 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

111944 XSH [uname\(\)](#)

111945 **CHANGE HISTORY**

111946 First released in Issue 2.

111947 **NAME**

111948 uncompress — expand compressed data

111949 **SYNOPSIS**111950 XSI `uncompress [-cfv] [file...]`111951 **DESCRIPTION**

111952 The *uncompress* utility shall restore files to their original state after they have been compressed  
 111953 using the *compress* utility. If no files are specified, the standard input shall be uncompressed to  
 111954 the standard output. If the invoking process has appropriate privileges, the ownership, modes,  
 111955 access time, and modification time of the original file shall be preserved.

111956 This utility shall support the uncompressing of any files produced by the *compress* utility on the  
 111957 same implementation. For files produced by *compress* on other systems, *uncompress* supports 9 to  
 111958 14-bit compression (see *compress*, **-b**); it is implementation-defined whether values of **-b** greater  
 111959 than 14 are supported.

111960 **OPTIONS**

111961 The *uncompress* utility shall conform to XBD [Section 12.2](#) (on page 216), except that Guideline 1  
 111962 does apply since the utility name has ten letters.

111963 The following options shall be supported:

- 111964 **-c** Write to standard output; no files are changed.
- 111965 **-f** Do not prompt for overwriting files. Except when run in the background, if **-f** is  
 111966 not given the user shall be prompted as to whether an existing file should be  
 111967 overwritten. If the standard input is not a terminal and **-f** is not given, *uncompress*  
 111968 shall write a diagnostic message to standard error and exit with a status greater  
 111969 than zero.
- 111970 **-v** Write messages to standard error concerning the expansion of each file.

111971 **OPERANDS**

111972 The following operand shall be supported:

- 111973 *file* A pathname of a file. If *file* already has the **.Z** suffix specified, it shall be used as the  
 111974 input file and the output file shall be named **file** with the **.Z** suffix removed.  
 111975 Otherwise, *file* shall be used as the name of the output file and **file** with the **.Z**  
 111976 suffix appended shall be used as the input file.

111977 **STDIN**

111978 The standard input shall be used only if no *file* operands are specified, or if a *file* operand is **'-'**.

111979 **INPUT FILES**

111980 Input files shall be in the format produced by the *compress* utility.

111981 **ENVIRONMENT VARIABLES**

111982 The following environment variables shall affect the execution of *uncompress*:

- 111983 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 111984 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 111985 variables used to determine the values of locale categories.)
- 111986 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 111987 internationalization variables.

- 111988 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 111989 characters (for example, single-byte as opposed to multi-byte characters in  
 111990 arguments).
- 111991 **LC\_MESSAGES**  
 111992 Determine the locale that should be used to affect the format and contents of  
 111993 diagnostic messages written to standard error.
- 111994 **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 111995 **ASYNCHRONOUS EVENTS**
- 111996 Default.
- 111997 **STDOUT**  
 111998 When there are no *file* operands or the *-c* option is specified, the uncompressed output is written  
 111999 to standard output.
- 112000 **STDERR**  
 112001 Prompts shall be written to the standard error output under the conditions specified in the  
 112002 DESCRIPTION and OPTIONS sections. The prompts shall contain the *file* pathname, but their  
 112003 format is otherwise unspecified. Otherwise, the standard error output shall be used only for  
 112004 diagnostic messages.
- 112005 **OUTPUT FILES**  
 112006 Output files are the same as the respective input files to *compress*.
- 112007 **EXTENDED DESCRIPTION**  
 112008 None.
- 112009 **EXIT STATUS**  
 112010 The following exit values shall be returned:  
 112011 0 Successful completion.  
 112012 >0 An error occurred.
- 112013 **CONSEQUENCES OF ERRORS**  
 112014 The input file remains unmodified.
- 112015 **APPLICATION USAGE**  
 112016 The limit of 14 on the *compress -b bits* argument is to achieve portability to all systems (within  
 112017 the restrictions imposed by the lack of an explicit published file format). Some implementations  
 112018 based on 16-bit architectures cannot support 15 or 16-bit uncompression.
- 112019 **EXAMPLES**  
 112020 None.
- 112021 **RATIONALE**  
 112022 None.
- 112023 **FUTURE DIRECTIONS**  
 112024 None.
- 112025 **SEE ALSO**  
 112026 *compress, zcat*  
 112027 XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

112028 **CHANGE HISTORY**

112029 First released in Issue 4.

112030 **Issue 6**

112031 The normative text is reworded to avoid use of the term “must” for application requirements.

112032 **Issue 7**112033 SD5-XCU-ERN-26 is applied, clarifying that this utility is allowed to break the Utility Syntax  
112034 Guidelines by having ten letters in its name.

112035 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

112036 **NAME**

112037 unexpand — convert spaces to tabs

112038 **SYNOPSIS**112039 unexpand [-a|-t *tablist*] [*file*...]112040 **DESCRIPTION**

112041 The *unexpand* utility shall copy files or standard input to standard output, converting <blank>  
 112042 characters at the beginning of each line into the maximum number of <tab> characters followed  
 112043 by the minimum number of <space> characters needed to fill the same column positions  
 112044 originally filled by the translated <blank> characters. By default, tabstops shall be set at every  
 112045 eighth column position. Each <backspace> shall be copied to the output, and shall cause the  
 112046 column position count for tab calculations to be decremented; the count shall never be  
 112047 decremented to a value less than one.

112048 **OPTIONS**112049 The *unexpand* utility shall conform to XBD [Section 12.2](#) (on page 216).

112050 The following options shall be supported:

112051 **-a** In addition to translating <blank> characters at the beginning of each line,  
 112052 translate all sequences of two or more <blank> characters immediately preceding a  
 112053 tab stop to the maximum number of <tab> characters followed by the minimum  
 112054 number of <space> characters needed to fill the same column positions originally  
 112055 filled by the translated <blank> characters.

112056 **-t *tablist*** Specify the tab stops. The application shall ensure that the *tablist* option-argument  
 112057 is a single argument consisting of a single positive decimal integer or multiple  
 112058 positive decimal integers, separated by <blank> or <comma> characters, in  
 112059 ascending order. If a single number is given, tabs shall be set *tablist* column  
 112060 positions apart instead of the default 8. If multiple numbers are given, the tabs  
 112061 shall be set at those specific column positions.

112062 The application shall ensure that each tab-stop position *N* is an integer value  
 112063 greater than zero, and the list shall be in strictly ascending order. This is taken to  
 112064 mean that, from the start of a line of output, tabbing to position *N* shall cause the  
 112065 next character output to be in the (*N*+1)th column position on that line. When the  
 112066 **-t** option is not specified, the default shall be the equivalent of specifying **-t 8**  
 112067 (except for the interaction with **-a**, described below).

112068 No <space>-to-<tab> conversions shall occur for characters at positions beyond  
 112069 the last of those specified in a multiple tab-stop list.

112070 When **-t** is specified, the presence or absence of the **-a** option shall be ignored;  
 112071 conversion shall not be limited to the processing of leading <blank> characters.

112072 **OPERANDS**

112073 The following operand shall be supported:

112074 *file* A pathname of a text file to be used as input.

112075 **STDIN**

112076 See the INPUT FILES section.

112077 **INPUT FILES**

112078 The input files shall be text files.

112079 **ENVIRONMENT VARIABLES**

112080 The following environment variables shall affect the execution of *unexpand*:

112081 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 112082 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 112083 variables used to determine the values of locale categories.)

112084 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 112085 internationalization variables.

112086 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 112087 characters (for example, single-byte as opposed to multi-byte characters in  
 112088 arguments and input files), the processing of <tab> and <space> characters, and  
 112089 for the determination of the width in column positions each character would  
 112090 occupy on an output device.

112091 *LC\_MESSAGES*

112092 Determine the locale that should be used to affect the format and contents of  
 112093 diagnostic messages written to standard error.

112094 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

112095 **ASYNCHRONOUS EVENTS**

112096 Default.

112097 **STDOUT**

112098 The standard output shall be equivalent to the input files with the specified <space>-to-<tab>  
 112099 conversions.

112100 **STDERR**

112101 The standard error shall be used only for diagnostic messages.

112102 **OUTPUT FILES**

112103 None.

112104 **EXTENDED DESCRIPTION**

112105 None.

112106 **EXIT STATUS**

112107 The following exit values shall be returned:

112108 0 Successful completion.

112109 >0 An error occurred.

112110 **CONSEQUENCES OF ERRORS**

112111 Default.

112112 **APPLICATION USAGE**

112113 One non-intuitive aspect of *unexpand* is its restriction to leading <space> characters when neither  
 112114 *-a* nor *-t* is specified. Users who always want to convert all <space> characters in a file can  
 112115 easily alias *unexpand* to use the *-a* or *-t 8* option.

112116 **EXAMPLES**

112117 None.

112118 **RATIONALE**

112119 On several occasions, consideration was given to adding a *-t* option to the *unexpand* utility to  
 112120 complement the *-t* in *expand* (see *expand*). The historical intent of *unexpand* was to translate  
 112121 multiple <blank> characters into tab stops, where tab stops were a multiple of eight column

112122 positions on most UNIX systems. An early proposal omitted `-t` because it seemed outside the  
112123 scope of the User Portability Utilities option; it was not described in any of the base documents  
112124 for IEEE Std 1003.2-1992. However, hard-coding tab stops every eight columns was not suitable  
112125 for the international community and broke historical precedents for some vendors in the  
112126 FORTRAN community, so `-t` was restored in conjunction with the list of valid extension  
112127 categories considered by the standard developers. Thus, *unexpand* is now the logical converse of  
112128 *expand*.

#### 112129 FUTURE DIRECTIONS

112130 None.

#### 112131 SEE ALSO

112132 *expand*, *tabs*

112133 XBD Chapter 8 (on page 173), Section 12.2 (on page 216)

#### 112134 CHANGE HISTORY

112135 First released in Issue 4.

#### 112136 Issue 6

112137 This utility is marked as part of the User Portability Utilities option.

112138 The definition of the `LC_CTYPE` environment variable is changed to align with the  
112139 IEEE P1003.2b draft standard.

112140 The normative text is reworded to avoid use of the term “must” for application requirements.

#### 112141 Issue 7

112142 The *unexpand* utility is moved from the User Portability Utilities option to the Base. User  
112143 Portability Utilities is now an option for interactive utilities.

112144 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

112145 **NAME**112146 unget — undo a previous get of an SCCS file (**DEVELOPMENT**)112147 **SYNOPSIS**112148 XSI unget [-ns] [-r *SID*] *file*...112149 **DESCRIPTION**112150 The *unget* utility shall reverse the effect of a *get* -e done prior to creating the intended new delta.112151 **OPTIONS**112152 The *unget* utility shall conform to XBD [Section 12.2](#) (on page 216).

112153 The following options shall be supported:

112154 -r *SID* Uniquely identify which delta is no longer intended. (This would have been  
 112155 specified by *get* as the new delta.) The use of this option is necessary only if two or  
 112156 more outstanding *get* commands for editing on the same SCCS file were done by  
 112157 the same person (login name).

112158 -s Suppress the writing to standard output of the intended delta's SID.

112159 -n Retain the file that was obtained by *get*, which would normally be removed from  
 112160 the current directory.

112161 **OPERANDS**

112162 The following operands shall be supported:

112163 *file* A pathname of an existing SCCS file or a directory. If *file* is a directory, the *unget*  
 112164 utility shall behave as though each file in the directory were specified as a named  
 112165 file, except that non-SCCS files (last component of the pathname does not begin  
 112166 with s.) and unreadable files shall be silently ignored.

112167 If exactly one *file* operand appears, and it is '-', the standard input shall be read;  
 112168 each line of the standard input shall be taken to be the name of an SCCS file to be  
 112169 processed. Non-SCCS files and unreadable files shall be silently ignored.

112170 **STDIN**

112171 The standard input shall be a text file used only when the *file* operand is specified as '-'. Each  
 112172 line of the text file shall be interpreted as an SCCS pathname.

112173 **INPUT FILES**

112174 Any SCCS files processed shall be files of an unspecified format.

112175 **ENVIRONMENT VARIABLES**112176 The following environment variables shall affect the execution of *unget*:

112177 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 112178 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 112179 variables used to determine the values of locale categories.)

112180 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 112181 internationalization variables.

112182 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 112183 characters (for example, single-byte as opposed to multi-byte characters in  
 112184 arguments and input files).

112185 *LC\_MESSAGES*

112186 Determine the locale that should be used to affect the format and contents of  
 112187 diagnostic messages written to standard error.



- 112188 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 112189 **ASYNCHRONOUS EVENTS**
- 112190 Default.
- 112191 **STDOUT**
- 112192 The standard output shall consist of a line for each file, in the following format:
- 112193 "%s\n", *<SID removed from file>*
- 112194 If there is more than one named file or if a directory or standard input is named, each pathname
- 112195 shall be written before each of the preceding lines:
- 112196 "\n%s:\n", *<pathname>*
- 112197 **STDERR**
- 112198 The standard error shall be used only for diagnostic messages.
- 112199 **OUTPUT FILES**
- 112200 Any SCCS files updated shall be files of an unspecified format. During processing of a *file*, a
- 112201 locking *z-file*, as described in *get*, and a *q-file* (a working copy of the *p-file*), may be created and
- 112202 deleted. The *p-file* and *g-file*, as described in *get*, shall be deleted.
- 112203 **EXTENDED DESCRIPTION**
- 112204 None.
- 112205 **EXIT STATUS**
- 112206 The following exit values shall be returned:
- 112207 0 Successful completion.
- 112208 >0 An error occurred.
- 112209 **CONSEQUENCES OF ERRORS**
- 112210 Default.
- 112211 **APPLICATION USAGE**
- 112212 None.
- 112213 **EXAMPLES**
- 112214 None.
- 112215 **RATIONALE**
- 112216 None.
- 112217 **FUTURE DIRECTIONS**
- 112218 None.
- 112219 **SEE ALSO**
- 112220 *delta*, *get*, *sact*
- 112221 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)
- 112222 **CHANGE HISTORY**
- 112223 First released in Issue 2.
- 112224 **Issue 6**
- 112225 The normative text is reworded to avoid use of the term “must” for application requirements.

112226 **Issue 7**  
112227

SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

112228 **NAME**112229        `uniq` — report or filter out repeated lines in a file112230 **SYNOPSIS**112231        `uniq [-c|-d|-u] [-f fields] [-s char] [input_file [output_file]]`112232 **DESCRIPTION**

112233        The *uniq* utility shall read an input file comparing adjacent lines, and write one copy of each  
 112234        input line on the output. The second and succeeding copies of repeated adjacent input lines shall  
 112235        not be written. The trailing <newline> of each line in the input shall be ignored when doing  
 112236        comparisons.

112237        Repeated lines in the input shall not be detected if they are not adjacent.

112238 **OPTIONS**

112239        The *uniq* utility shall conform to XBD [Section 12.2](#) (on page 216), except that '+' may be  
 112240        recognized as an option delimiter as well as '-'.

112241        The following options shall be supported:

112242        **-c**        Precede each output line with a count of the number of times the line occurred in  
 112243        the input.

112244        **-d**        Suppress the writing of lines that are not repeated in the input.

112245        **-f *fields***    Ignore the first *fields* fields on each input line when doing comparisons, where  
 112246        *fields* is a positive decimal integer. A field is the maximal string matched by the  
 112247        basic regular expression:

112248        `[[[:blank:]]*[^[:blank:]]*]`

112249        If the *fields* option-argument specifies more fields than appear on an input line, a  
 112250        null string shall be used for comparison.

112251        **-s *chars***    Ignore the first *chars* characters when doing comparisons, where *chars* shall be a  
 112252        positive decimal integer. If specified in conjunction with the **-f** option, the first  
 112253        *chars* characters after the first *fields* fields shall be ignored. If the *chars* option-  
 112254        argument specifies more characters than remain on an input line, a null string shall  
 112255        be used for comparison.

112256        **-u**        Suppress the writing of lines that are repeated in the input.

112257 **OPERANDS**

112258        The following operands shall be supported:

112259        *input\_file*    A pathname of the input file. If the *input\_file* operand is not specified, or if the  
 112260        *input\_file* is '-', the standard input shall be used.

112261        *output\_file*    A pathname of the output file. If the *output\_file* operand is not specified, the  
 112262        standard output shall be used. The results are unspecified if the file named by  
 112263        *output\_file* is the file named by *input\_file*.

112264 **STDIN**112265        The standard input shall be used only if no *input\_file* operand is specified or if *input\_file* is '-'.

112266        See the INPUT FILES section.

112267 **INPUT FILES**

112268        The input file shall be a text file.

## 112269 ENVIRONMENT VARIABLES

112270 The following environment variables shall affect the execution of *uniq*:

112271 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 112272 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 112273 variables used to determine the values of locale categories.)

112274 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 112275 internationalization variables.

112276 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 112277 characters (for example, single-byte as opposed to multi-byte characters in  
 112278 arguments and input files) and which characters constitute a <blank> in the  
 112279 current locale.

112280 *LC\_MESSAGES*

112281 Determine the locale that should be used to affect the format and contents of  
 112282 diagnostic messages written to standard error.

112283 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 112284 ASYNCHRONOUS EVENTS

112285 Default.

## 112286 STDOUT

112287 The standard output shall be used if no *output\_file* operand is specified, and shall be used if the  
 112288 *output\_file* operand is '-' and the implementation treats the '-' as meaning standard output.  
 112289 Otherwise, the standard output shall not be used. See the OUTPUT FILES section.

## 112290 STDERR

112291 The standard error shall be used only for diagnostic messages.

## 112292 OUTPUT FILES

112293 If the -c option is specified, the output file shall be empty or each line shall be of the form:

112294 "%d %s", <number of duplicates>, <line>

112295 otherwise, the output file shall be empty or each line shall be of the form:

112296 "%s", <line>

## 112297 EXTENDED DESCRIPTION

112298 None.

## 112299 EXIT STATUS

112300 The following exit values shall be returned:

112301 0 The utility executed successfully.

112302 >0 An error occurred.

## 112303 CONSEQUENCES OF ERRORS

112304 Default.

**APPLICATION USAGE**

If the collating sequence of the current locale has a total ordering of all characters, the *sort* utility can be used to cause repeated lines to be adjacent in the input file. If the collating sequence does not have a total ordering of all characters, the *sort* utility should still do this but it might not. To ensure that all duplicate lines are eliminated, and have the output sorted according the collating sequence of the current locale, applications should use:

```
LC_ALL=C sort -u | sort
```

instead of:

```
sort | uniq
```

To remove duplicate lines based on whether they collate equally instead of whether they are identical, applications should use:

```
sort -u
```

instead of:

```
sort | uniq
```

When using *uniq* to process pathnames, it is recommended that *LC\_ALL*, or at least *LC\_CTYPE* and *LC\_COLLATE*, are set to POSIX or C in the environment, since pathnames can contain byte sequences that do not form valid characters in some locales, in which case the utility's behavior would be undefined. In the POSIX locale each byte is a valid single-byte character, and therefore this problem is avoided.

**EXAMPLES**

The following input file data (but flushed left) was used for a test series on *uniq*:

```
#01 foo0 bar0 fool bar1
#02 bar0 fool bar1 fool
#03 foo0 bar0 fool bar1
#04
#05 foo0 bar0 fool bar1
#06 foo0 bar0 fool bar1
#07 bar0 fool bar1 foo0
```

What follows is a series of test invocations of the *uniq* utility that use a mixture of *uniq* options against the input file data. These tests verify the meaning of *adjacent*. The *uniq* utility views the input data as a sequence of strings delimited by '\n'. Accordingly, for the *fields* member of the sequence, *uniq* interprets unique or repeated adjacent lines strictly relative to the *fields*+1th member.

1. This first example tests the line counting option, comparing each line of the input file data starting from the second field:

```
uniq -c -f 1 uniq_0I.t
1 #01 foo0 bar0 fool bar1
1 #02 bar0 fool bar1 fool
1 #03 foo0 bar0 fool bar1
1 #04
2 #05 foo0 bar0 fool bar1
1 #07 bar0 fool bar1 foo0
```

The number '2', prefixing the fifth line of output, signifies that the *uniq* utility detected a pair of repeated lines. Given the input data, this can only be true when *uniq* is run using the *-f 1* option (which shall cause *uniq* to ignore the first field on each input line).

- 112350 2. The second example tests the option to suppress unique lines, comparing each line of the  
112351 input file data starting from the second field:
- ```
112352      uniq -d -f 1 uniq_0I.t
112353      #05 foo0 bar0 fool bar1
```
- 112354 3. This test suppresses repeated lines, comparing each line of the input file data starting  
112355 from the second field:
- ```
112356      uniq -u -f 1 uniq_0I.t
112357      #01 foo0 bar0 fool bar1
112358      #02 bar0 fool bar1 fool
112359      #03 foo0 bar0 fool bar1
112360      #04
112361      #07 bar0 fool bar1 foo0
```
- 112362 4. This suppresses unique lines, comparing each line of the input file data starting from the  
112363 third character:
- ```
112364      uniq -d -s 2 uniq_0I.t
```
- 112365 In the last example, the *uniq* utility found no input matching the above criteria.

#### 112366 RATIONALE

112367 Some historical implementations have limited lines to be 1 080 bytes in length, which does not  
112368 meet the implied {LINE\_MAX} limit.

112369 Earlier versions of this standard allowed the *-number* and *+number* options. These options are no  
112370 longer specified by POSIX.1-2008 but may be present in some implementations.

#### 112371 FUTURE DIRECTIONS

112372 None.

#### 112373 SEE ALSO

112374 *comm*, *sort*

112375 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

#### 112376 CHANGE HISTORY

112377 First released in Issue 2.

#### 112378 Issue 6

112379 The obsolescent SYNOPSIS and associated text are removed.

112380 The normative text is reworded to avoid use of the term “must” for application requirements.

112381 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/40 is applied, adding *LC\_COLLATE* to the  
112382 ENVIRONMENT VARIABLES section, and changing “the application shall ensure that” in the  
112383 OUTPUT FILES section.

#### 112384 Issue 7

112385 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying that ‘+’ may be recognized  
112386 as an option delimiter in the OPTIONS section.

112387 Austin Group Interpretation 1003.1-2001 #092 is applied.

112388 Austin Group Interpretation 1003.1-2001 #133 is applied, clarifying the behavior of the trailing  
112389 <newline>.

112390 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

112391 SD5-XCU-ERN-141 is applied, updating the EXAMPLES section.



112392 **NAME**

112393 unlink — call the *unlink()* function

112394 **SYNOPSIS**

112395 XSI `unlink file`

112396 **DESCRIPTION**

112397 The *unlink* utility shall perform the function call:

112398 `unlink(file);`

112399 A user may need appropriate privileges to invoke the *unlink* utility.

112400 **OPTIONS**

112401 None.

112402 **OPERANDS**

112403 The following operands shall be supported:

112404 *file* The pathname of an existing file.

112405 **STDIN**

112406 Not used.

112407 **INPUT FILES**

112408 Not used.

112409 **ENVIRONMENT VARIABLES**

112410 The following environment variables shall affect the execution of *unlink*:

112411 *LANG* Provide a default value for the internationalization variables that are unset or null.  
112412 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
112413 variables used to determine the values of locale categories.)

112414 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
112415 internationalization variables.

112416 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
112417 characters (for example, single-byte as opposed to multi-byte characters in  
112418 arguments).

112419 *LC\_MESSAGES*  
112420 Determine the locale that should be used to affect the format and contents of  
112421 diagnostic messages written to standard error.

112422 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

112423 **ASYNCHRONOUS EVENTS**

112424 Default.

112425 **STDOUT**

112426 None.

112427 **STDERR**

112428 The standard error shall be used only for diagnostic messages.

112429 **OUTPUT FILES**

112430 None.



112431 **EXTENDED DESCRIPTION**

112432       None.

112433 **EXIT STATUS**

112434       The following exit values shall be returned:

112435       0   Successful completion.

112436       &gt;0   An error occurred.

112437 **CONSEQUENCES OF ERRORS**

112438       Default.

112439 **APPLICATION USAGE**

112440       None.

112441 **EXAMPLES**

112442       None.

112443 **RATIONALE**

112444       None.

112445 **FUTURE DIRECTIONS**

112446       None.

112447 **SEE ALSO**112448       [link](#), [rm](#)112449       XBD [Chapter 8](#) (on page 173)112450       XSH [unlink\(\)](#)112451 **CHANGE HISTORY**

112452       First released in Issue 5.

112453 **NAME**

112454       uucp — system-to-system copy

112455 **SYNOPSIS**112456 UU       uucp [-cCdfjmr] [-n *user*] *source-file*... *destination-file*112457 **DESCRIPTION**

112458       The *uucp* utility shall copy files named by the *source-file* argument to the *destination-file* argument.  
 112459       The files named can be on local or remote systems.

112460       The *uucp* utility cannot guarantee support for all character encodings in all circumstances. For  
 112461       example, transmission data may be restricted to 7 bits by the underlying network, 8-bit data and  
 112462       filenames need not be portable to non-internationalized systems, and so on. Under these  
 112463       circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991  
 112464       standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used,  
 112465       and that only characters defined in the portable filename character set be used for naming files.  
 112466       The protocol for transfer of files is unspecified by POSIX.1-2008.

112467       Typical implementations of this utility require a communications line configured to use XBD  
 112468       Chapter 11 (on page 199), but other communications means may be used. On systems where  
 112469       there are no available communications means (either temporarily or permanently), this utility  
 112470       shall write an error message describing the problem and exit with a non-zero exit status.

112471 **OPTIONS**112472       The *uucp* utility shall conform to XBD Section 12.2 (on page 216).

112473       The following options shall be supported:

- 112474       -c       Do not copy local file to the spool directory for transfer to the remote machine
- 112475       (default).
- 112476       -C       Force the copy of local files to the spool directory for transfer.
- 112477       -d       Make all necessary directories for the file copy (default).
- 112478       -f       Do not make intermediate directories for the file copy.
- 112479       -j       Write the job identification string to standard output. This job identification can be
- 112480       used by *uustat* to obtain the status or terminate a job.
- 112481       -m       Send mail to the requester when the copy is completed.
- 112482       -n *user*   Notify *user* on the remote system that a file was sent.
- 112483       -r       Do not start the file transfer; just queue the job.

112484 **OPERANDS**

112485       The following operands shall be supported:

112486       *destination-file*, *source-file*

112487       A pathname of a file to be copied to, or from, respectively. Either name can be a  
 112488       pathname on the local machine, or can have the form:

112489       *system-name*!*pathname*

112490       where *system-name* is taken from a list of system names that *uucp* knows about.  
 112491       The destination *system-name* can also be a list of names such as:

112492       *system-name*!*system-name*! . . . !*system-name*!*pathname*

112493       in which case, an attempt is made to send the file via the specified route to the

112494 destination. Care should be taken to ensure that intermediate nodes in the route  
112495 are willing to forward information.

112496 The shell pattern matching notation characters '?', '\*', and "[...]" appearing  
112497 in *pathname* shall be expanded on the appropriate system.

112498 Pathnames can be one of:

- 112499 1. An absolute pathname.
- 112500 2. A pathname preceded by *~user* where *user* is a login name on the specified  
112501 system and is replaced by that user's login directory. Note that if an invalid  
112502 login is specified, the default is to the public directory (called *PUBDIR*; the  
112503 actual location of *PUBDIR* is implementation-defined).
- 112504 3. A pathname preceded by *~/destination* where *destination* is appended to  
112505 *PUBDIR*.

112506 **Note:** This destination is treated as a filename unless more than one file is being  
112507 transferred by this request or the destination is already a directory. To  
112508 ensure that it is a directory, follow the destination with a '/'. For  
112509 example, *~/dan/* as the destination makes the directory **PUBDIR/dan** if it  
112510 does not exist and puts the requested files in that directory.

- 112511 4. Anything else shall be prefixed by the current directory.

112512 If the result is an erroneous pathname for the remote system, the copy shall fail. If  
112513 the *destination-file* is a directory, the last part of the *source-file* name shall be used.

112514 The read, write, and execute permissions given by *uucp* are implementation-  
112515 defined.

## 112516 STDIN

112517 Not used.

## 112518 INPUT FILES

112519 The files to be copied are regular files.

## 112520 ENVIRONMENT VARIABLES

112521 The following environment variables shall affect the execution of *uucp*:

112522 *LANG* Provide a default value for the internationalization variables that are unset or null.  
112523 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
112524 variables used to determine the values of locale categories.)

112525 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
112526 internationalization variables.

112527 *LC\_COLLATE*  
112528 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
112529 character collating elements within bracketed filename patterns.

112530 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
112531 characters (for example, single-byte as opposed to multi-byte characters in  
112532 arguments and input files) and the behavior of character classes within bracketed  
112533 filename patterns (for example, "[[:lower:]]\*").

112534 *LC\_MESSAGES*  
112535 Determine the locale that should be used to affect the format and contents of  
112536 diagnostic messages written to standard error, and informative messages written  
112537 to standard output.

- 112538 *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 112539 **ASYNCHRONOUS EVENTS**
- 112540 Default.
- 112541 **STDOUT**
- 112542 Not used.
- 112543 **STDERR**
- 112544 The standard error shall be used only for diagnostic messages.
- 112545 **OUTPUT FILES**
- 112546 The output files (which may be on other systems) are copies of the input files.
- 112547 If *-m* is used, mail files are modified.
- 112548 **EXTENDED DESCRIPTION**
- 112549 None.
- 112550 **EXIT STATUS**
- 112551 The following exit values shall be returned:
- 112552 0 Successful completion.
- 112553 >0 An error occurred.
- 112554 **CONSEQUENCES OF ERRORS**
- 112555 Default.
- 112556 **APPLICATION USAGE**
- 112557 This utility is part of the UUCP Utilities option and need not be supported by all implementations.
- 112558
- 112559 The domain of remotely accessible files can (and for obvious security reasons usually should) be severely restricted.
- 112560
- 112561 Note that the *'!'* character in addresses has to be escaped when using *cs**h* as a command interpreter because of its history substitution syntax. For *ksh* and *sh* the escape is not necessary, but may be used.
- 112562
- 112563
- 112564 As noted above, shell metacharacters appearing in pathnames are expanded on the appropriate system. On an internationalized system, this is done under the control of local settings of *LC\_COLLATE* and *LC\_CTYPE*. Thus, care should be taken when using bracketed filename patterns, as collation and typing rules may vary from one system to another. Also be aware that certain types of expression (that is, equivalence classes, character classes, and collating symbols) need not be supported on non-internationalized systems.
- 112565
- 112566
- 112567
- 112568
- 112569
- 112570 **EXAMPLES**
- 112571 None.
- 112572 **RATIONALE**
- 112573 None.
- 112574 **FUTURE DIRECTIONS**
- 112575 None.
- 112576 **SEE ALSO**
- 112577 *mailx*, *uuencode*, *uustat*, *uux*
- 112578 XBD [Chapter 8](#) (on page 173), [Chapter 11](#) (on page 199), [Section 12.2](#) (on page 216)

112579 **CHANGE HISTORY**

112580 First released in Issue 2.

112581 **Issue 6**112582 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.112583 The UN margin codes and associated shading are removed from the *-C*, *-f*, *-j*, *-n*, and *-r*  
112584 options in response to The Open Group Base Resolution bwg2001-003.112585 **Issue 7**

112586 SD5-XCU-ERN-46 is applied, moving this utility to the UUCP Utilities Option Group.

112587 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

112588 **NAME**

112589       uudecode — decode a binary file

112590 **SYNOPSIS**112591       uudecode [-o *outfile*] [*file*]112592 **DESCRIPTION**

112593       The *uudecode* utility shall read a file, or standard input if no file is specified, that includes data  
 112594       created by the *uuencode* utility. The *uudecode* utility shall scan the input file, searching for data  
 112595       compatible with one of the formats specified in *uuencode*, and attempt to create or overwrite the  
 112596       file described by the data (or overridden by the **-o** option). The pathname shall be contained in  
 112597       the data or specified by the **-o** option. The file access permission bits and contents for the file to  
 112598       be produced shall be contained in that data. The mode bits of the created file (other than  
 112599       standard output) shall be set from the file access permission bits contained in the data; that is,  
 112600       other attributes of the mode, including the file mode creation mask (see *umask*), shall not affect  
 112601       the file being produced. If either of the *op* characters '+' and '-' (see *chmod*) are specified in  
 112602       symbolic mode, the initial mode on which those operations are based is unspecified.

112603       If the pathname of the file resolves to an existing file and the user does not have write  
 112604       permission on that file, *uudecode* shall terminate with an error. If the pathname of the file resolves  
 112605       to an existing file and the user has write permission on that file, the existing file shall be  
 112606       overwritten and, if possible, the mode bits of the file (other than standard output) shall be set as  
 112607       described above; if the mode bits cannot be set, *uudecode* shall not treat this as an error.

112608       If the input data was produced by *uuencode* on a system with a different number of bits per byte  
 112609       than on the target system, the results of *uudecode* are unspecified.

112610 **OPTIONS**112611       The *uudecode* utility shall conform to XBD [Section 12.2](#) (on page 216).

112612       The following option shall be supported by the implementation:

112613       **-o *outfile***   A pathname of a file that shall be used instead of any pathname contained in the  
 112614       input data. Specifying an *outfile* option-argument of **/dev/stdout** shall indicate  
 112615       standard output.

112616 **OPERANDS**

112617       The following operand shall be supported:

112618       *file*           The pathname of a file containing the output of *uuencode*.112619 **STDIN**

112620       See the INPUT FILES section.

112621 **INPUT FILES**112622       The input files shall be files containing the output of *uuencode*.112623 **ENVIRONMENT VARIABLES**112624       The following environment variables shall affect the execution of *uudecode*:

112625       **LANG**           Provide a default value for the internationalization variables that are unset or null.  
 112626       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 112627       variables used to determine the values of locale categories.)

112628       **LC\_ALL**          If set to a non-empty string value, override the values of all the other  
 112629       internationalization variables.

112630       **LC\_CTYPE**       Determine the locale for the interpretation of sequences of bytes of text data as  
 112631       characters (for example, single-byte as opposed to multi-byte characters in  
 112632       arguments and input files).

112633 **LC\_MESSAGES**

112634 Determine the locale that should be used to affect the format and contents of  
 112635 diagnostic messages written to standard error.

112636 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

112637 **ASYNCHRONOUS EVENTS**

112638 Default.

112639 **STDOUT**

112640 If the file data header encoded by *uuencode* is `-` or `/dev/stdout`, or the `-o /dev/stdout` option  
 112641 overrides the file data, the standard output shall be in the same format as the file originally  
 112642 encoded by *uuencode*. Otherwise, the standard output shall not be used.

112643 **STDERR**

112644 The standard error shall be used only for diagnostic messages.

112645 **OUTPUT FILES**

112646 The output file shall be in the same format as the file originally encoded by *uuencode*.

112647 **EXTENDED DESCRIPTION**

112648 None.

112649 **EXIT STATUS**

112650 The following exit values shall be returned:

112651 0 Successful completion.

112652 >0 An error occurred.

112653 **CONSEQUENCES OF ERRORS**

112654 Default.

112655 **APPLICATION USAGE**

112656 The user who is invoking *uudecode* must have write permission on any file being created.

112657 The output of *uuencode* is essentially an encoded bit stream that is not cognizant of byte  
 112658 boundaries. It is possible that a 9-bit byte target machine can process input from an 8-bit source,  
 112659 if it is aware of the requirement, but the reverse is unlikely to be satisfying. Of course, the only  
 112660 data that is meaningful for such a transfer between architectures is generally character data.

112661 **EXAMPLES**

112662 None.

112663 **RATIONALE**

112664 Input files are not necessarily text files, as stated by an early proposal. Although the *uuencode*  
 112665 output is a text file, that output could have been wrapped within another file or mail message  
 112666 that is not a text file.

112667 The `-o` option is not historical practice, but was added at the request of WG15 so that the user  
 112668 could override the target pathname without having to edit the input data itself.

112669 In early drafts, the `[-o outfile]` option-argument allowed the use of `-` to mean standard output.  
 112670 The symbol `-` has only been used previously in POSIX.1-2008 as a standard input indicator. The  
 112671 standard developers did not wish to overload the meaning of `-` in this manner. The `/dev/stdout`  
 112672 concept exists on most modern systems. The `/dev/stdout` syntax does not refer to a new special  
 112673 file. It is just a magic cookie to specify standard output.

112674 **FUTURE DIRECTIONS**

112675 None.

112676 **SEE ALSO**112677 *chmod, umask, uuencode*112678 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)112679 **CHANGE HISTORY**

112680 First released in Issue 4.

112681 **Issue 6**

112682 This utility is marked as part of the User Portability Utilities option.

112683 The `-o outfile` option is added, as specified in the IEEE P1003.2b draft standard.

112684 The normative text is reworded to avoid use of the term “must” for application requirements.

112685 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/35 is applied, clarifying in the  
112686 DESCRIPTION that the initial mode used if either of the *op* characters is '+' or '-' is  
112687 unspecified.

112688 **Issue 7**

112689 The *uudecode* utility is moved from the User Portability Utilities option to the Base. User  
112690 Portability Utilities is now an option for interactive utilities.

112691 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



112692 **NAME**

112693 uuencode — encode a binary file

112694 **SYNOPSIS**112695 uuencode [-m] [*file*] *decode\_pathname*112696 **DESCRIPTION**

112697 The *uuencode* utility shall write an encoded version of the named input file, or standard input if  
 112698 no *file* is specified, to standard output. The output shall be encoded using one of the algorithms  
 112699 described in the STDOUT section and shall include the file access permission bits (in *chmod* octal  
 112700 or symbolic notation) of the input file and the *decode\_pathname*, for re-creation of the file on  
 112701 another system that conforms to this volume of POSIX.1-2008.

112702 **OPTIONS**112703 The *uuencode* utility shall conform to XBD [Section 12.2](#) (on page 216).

112704 The following option shall be supported by the implementation:

112705 **-m** Encode the output using the MIME Base64 algorithm described in STDOUT. If **-m**  
 112706 is not specified, the historical algorithm described in STDOUT shall be used.

112707 **OPERANDS**

112708 The following operands shall be supported:

112709 *decode\_pathname*

112710 The pathname of the file into which the *uudecode* utility shall place the decoded  
 112711 file. Specifying a *decode\_pathname* operand of **/dev/stdout** shall indicate that  
 112712 *uudecode* is to use standard output. If there are characters in *decode\_pathname* that  
 112713 are not in the portable filename character set the results are unspecified.

112714 *file* A pathname of the file to be encoded.

112715 **STDIN**

112716 See the INPUT FILES section.

112717 **INPUT FILES**

112718 Input files can be files of any type.

112719 **ENVIRONMENT VARIABLES**112720 The following environment variables shall affect the execution of *uuencode*:

112721 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 112722 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 112723 variables used to determine the values of locale categories.)

112724 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 112725 internationalization variables.

112726 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 112727 characters (for example, single-byte as opposed to multi-byte characters in  
 112728 arguments and input files).

112729 **LC\_MESSAGES**

112730 Determine the locale that should be used to affect the format and contents of  
 112731 diagnostic messages written to standard error.

112732 **XS** **NLSPATH** Determine the location of message catalogs for the processing of **LC\_MESSAGES**.

112733 **ASYNCHRONOUS EVENTS**

112734 Default.

112735 **STDOUT**112736 **uuencode Base64 Algorithm**112737 The standard output shall be a text file (encoded in the character set of the current locale) that  
112738 begins with the line:

112739 "begin-base64Δ%sΔs\n", &lt;mode&gt;, &lt;decode\_pathname&gt;

112740 and ends with the line:

112741 "====\n"

112742 In both cases, the lines shall have no preceding or trailing &lt;blank&gt; characters.

112743 The encoding process represents 24-bit groups of input bits as output strings of four encoded  
 112744 characters. Proceeding from left to right, a 24-bit input group shall be formed by concatenating  
 112745 three 8-bit input groups. Each 24-bit input group then shall be treated as four concatenated 6-bit  
 112746 groups, each of which shall be translated into a single digit in the Base64 alphabet. When  
 112747 encoding a bit stream via the Base64 encoding, the bit stream shall be presumed to be ordered  
 112748 with the most-significant bit first. That is, the first bit in the stream shall be the high-order bit in  
 112749 the first byte, and the eighth bit shall be the low-order bit in the first byte, and so on. Each 6-bit  
 112750 group is used as an index into an array of 64 printable characters, as shown in [Table 4-22](#).

112751 **Table 4-22** uuencode Base64 Values

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v	(pad)	=
14	O	31	f	48	w		
15	P	32	g	49	x		
16	Q	33	h	50	y		

112770 The character referenced by the index shall be placed in the output string.

112771 The output stream (encoded bytes) shall be represented in lines of no more than 76 characters  
 112772 each. All line breaks or other characters not found in the table shall be ignored by decoding  
 112773 software (see [uudecode](#)).

112774 Special processing shall be performed if fewer than 24 bits are available at the end of a message  
 112775 or encapsulated part of a message. A full encoding quantum shall always be completed at the

end of a message. When fewer than 24 input bits are available in an input group, zero bits shall be added (on the right) to form an integral number of 6-bit groups. Output character positions that are not required to represent actual input data shall be set to the character '='. Since all Base64 input is an integral number of octets, only the following cases can arise:

1. The final quantum of encoding input is an integral multiple of 24 bits; here, the final unit of encoded output shall be an integral multiple of 4 characters with no '=' padding.
2. The final quantum of encoding input is exactly 16 bits; here, the final unit of encoded output shall be three characters followed by one '=' padding character.
3. The final quantum of encoding input is exactly 8 bits; here, the final unit of encoded output shall be two characters followed by two '=' padding characters.

A terminating "====" evaluates to nothing and denotes the end of the encoded data.

### uuencode Historical Algorithm

The standard output shall be a text file (encoded in the character set of the current locale) that begins with the line:

```
"beginΔ%sΔ%s\n" <mode>, <decode_pathname>
```

and ends with the line:

```
"end\n"
```

In both cases, the lines shall have no preceding or trailing <blank> characters.

The algorithm that shall be used for lines in between **begin** and **end** takes three octets as input and writes four characters of output by splitting the input at six-bit intervals into four octets, containing data in the lower six bits only. These octets shall be converted to characters by adding a value of 0x20 to each octet, so that each octet is in the range [0x20,0x5f], and then it shall be assumed to represent a printable character in the ISO/IEC 646:1991 standard encoded character set. It then shall be translated into the corresponding character codes for the codeset in use in the current locale. (For example, the octet 0x41, representing 'A', would be translated to 'A' in the current codeset, such as 0xc1 if it were EBCDIC.)

Where the bits of two octets are combined, the least significant bits of the first octet shall be shifted left and combined with the most significant bits of the second octet shifted right. Thus the three octets *A*, *B*, *C* shall be converted into the four octets:

```
0x20 + (( A >> 2 ) & 0x3F)
0x20 + ((( A << 4 ) | (( B >> 4 ) & 0xF ) ) & 0x3F)
0x20 + ((( B << 2 ) | (( C >> 6 ) & 0x3 ) ) & 0x3F)
0x20 + (( C ) & 0x3F)
```

These octets then shall be translated into the local character set.

Each encoded line contains a length character, equal to the number of characters to be decoded plus 0x20 translated to the local character set as described above, followed by the encoded characters. The maximum number of octets to be encoded on each line shall be 45.

### STDERR

The standard error shall be used only for diagnostic messages.

**112815 OUTPUT FILES**

112816 None.

**112817 EXTENDED DESCRIPTION**

112818 None.

**112819 EXIT STATUS**

112820 The following exit values shall be returned:

112821 0 Successful completion.

112822 >0 An error occurred.

**112823 CONSEQUENCES OF ERRORS**

112824 Default.

**112825 APPLICATION USAGE**

112826 The file is expanded by 35 percent (each three octets become four, plus control information)  
112827 causing it to take longer to transmit.

112828 Since this utility is intended to create files to be used for data interchange between systems with  
112829 possibly different codesets, and to represent binary data as a text file, the ISO/IEC 646:1991  
112830 standard was chosen for a midpoint in the algorithm as a known reference point. The output  
112831 from *uuencode* is a text file on the local system. If the output were in the ISO/IEC 646:1991  
112832 standard codeset, it might not be a text file (at least because the <newline> characters might not  
112833 match), and the goal of creating a text file would be defeated. If this text file was then carried to  
112834 another machine with the same codeset, it would be perfectly compatible with that system's  
112835 *uudecode*. If it was transmitted over a mail system or sent to a machine with a different codeset,  
112836 it is assumed that, as for every other text file, some translation mechanism would convert it (by  
112837 the time it reached a user on the other system) into an appropriate codeset. This translation only  
112838 makes sense from the local codeset, not if the file has been put into a ISO/IEC 646:1991 standard  
112839 representation first. Similarly, files processed by *uuencode* can be placed in *pax* archives,  
112840 intermixed with other text files in the same codeset.

**112841 EXAMPLES**

112842 None.

**112843 RATIONALE**

112844 A new algorithm was added at the request of the international community to parallel work in  
112845 RFC 2045 (MIME). As with the historical *uuencode* format, the Base64 Content-Transfer-Encoding  
112846 is designed to represent arbitrary sequences of octets in a form that is not humanly readable. A  
112847 65-character subset of the ISO/IEC 646:1991 standard is used, enabling 6 bits to be represented  
112848 per printable character. (The extra 65th character, '=', is used to signify a special processing  
112849 function.)

112850 This subset has the important property that it is represented identically in all versions of the  
112851 ISO/IEC 646:1991 standard, including US ASCII, and all characters in the subset are also  
112852 represented identically in all versions of EBCDIC. The historical *uuencode* algorithm does not  
112853 share this property, which is the reason that a second algorithm was added to the ISO POSIX-2  
112854 standard.

112855 The string "====" was used for the termination instead of the end used in the original format  
112856 because the latter is a string that could be valid encoded input.

112857 In an early draft, the *-m* option was named *-b* (for Base64), but it was renamed to reflect its  
112858 relationship to the RFC 2045. A *-u* was also present to invoke the default algorithm, but since  
112859 this was not historical practice, it was omitted as being unnecessary.

112860 See the RATIONALE section in *uuencode* for the derivation of the **/dev/stdout** symbol.

112861 **FUTURE DIRECTIONS**

112862 None.

112863 **SEE ALSO**

112864 *chmod*, *mailx*, *uuencode*

112865 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

112866 **CHANGE HISTORY**

112867 First released in Issue 4.

112868 **Issue 6**

112869 This utility is marked as part of the User Portability Utilities option.

112870 The Base64 algorithm and the ability to output to **/dev/stdout** are added as specified in the  
112871 IEEE P1003.2b draft standard.

112872 **Issue 7**

112873 The *uuencode* utility is moved from the User Portability Utilities option to the Base. User  
112874 Portability Utilities is now an option for interactive utilities.

112875 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

112876 **NAME**

112877 uustat — uucp status enquiry and job control

112878 **SYNOPSIS**112879 UU uustat [-q|-k *jobid*|-r *jobid*]112880 uustat [-s *system*] [-u *user*]112881 **DESCRIPTION**

112882 The *uustat* utility shall display the status of, or cancel, previously specified *uucp* requests, or  
 112883 provide general status on *uucp* connections to other systems.

112884 When no options are given, *uustat* shall write to standard output the status of all *uucp* requests  
 112885 issued by the current user.

112886 Typical implementations of this utility require a communications line configured to use XBD  
 112887 [Chapter 11](#) (on page 199), but other communications means may be used. On systems where  
 112888 there are no available communications means (either temporarily or permanently), this utility  
 112889 shall write an error message describing the problem and exit with a non-zero exit status.

112890 **OPTIONS**112891 The *uustat* utility shall conform to XBD [Section 12.2](#) (on page 216).

112892 The following options shall be supported:

112893 **-q** Write the jobs queued for each machine.

112894 **-k *jobid*** Kill the *uucp* request whose job identification is *jobid*. The application shall ensure  
 112895 that the killed *uucp* request belongs to the person invoking *uustat* unless that user  
 112896 has appropriate privileges.

112897 **-r *jobid*** Rejuvenate *jobid*. The files associated with *jobid* are touched so that their  
 112898 modification time is set to the current time. This prevents the cleanup program  
 112899 from deleting the job until the jobs modification time reaches the limit imposed by  
 112900 the program.

112901 **-s *system*** Write the status of all *uucp* requests for remote system *system*.112902 **-u *user*** Write the status of all *uucp* requests issued by *user*.112903 **OPERANDS**

112904 None.

112905 **STDIN**

112906 Not used.

112907 **INPUT FILES**

112908 None.

112909 **ENVIRONMENT VARIABLES**112910 The following environment variables shall affect the execution of *uustat*:

112911 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 112912 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 112913 variables used to determine the values of locale categories.)

112914 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 112915 internationalization variables.

- 112916 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
112917 characters (for example, single-byte as opposed to multi-byte characters in  
112918 arguments).
- 112919 **LC\_MESSAGES**  
112920 Determine the locale that should be used to affect the format and contents of  
112921 diagnostic messages written to standard error, and informative messages written  
112922 to standard output.
- 112923 **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 112924 **ASYNCHRONOUS EVENTS**  
112925 Default.
- 112926 **STDOUT**  
112927 The standard output shall consist of information about each job selected, in an unspecified  
112928 format. The information shall include at least the job ID, the user ID or name, and the remote  
112929 system name.
- 112930 **STDERR**  
112931 The standard error shall be used only for diagnostic messages.
- 112932 **OUTPUT FILES**  
112933 None.
- 112934 **EXTENDED DESCRIPTION**  
112935 None.
- 112936 **EXIT STATUS**  
112937 The following exit values shall be returned:  
112938 0 Successful completion.  
112939 >0 An error occurred.
- 112940 **CONSEQUENCES OF ERRORS**  
112941 Default.
- 112942 **APPLICATION USAGE**  
112943 This utility is part of the UUCP Utilities option and need not be supported by all  
112944 implementations.
- 112945 **EXAMPLES**  
112946 None.
- 112947 **RATIONALE**  
112948 None.
- 112949 **FUTURE DIRECTIONS**  
112950 None.
- 112951 **SEE ALSO**  
112952 *uucp*  
112953 XBD [Chapter 8](#) (on page 173), [Chapter 11](#) (on page 199), [Section 12.2](#) (on page 216)
- 112954 **CHANGE HISTORY**  
112955 First released in Issue 2.

112956 **Issue 6**

112957 The normative text is reworded to avoid use of the term “must” for application requirements.

112958 The *LC\_TIME* and *TZ* entries are removed from the ENVIRONMENT VARIABLES section.

112959 The UN margin code and associated shading are removed from the **-q** option in response to The  
112960 Open Group Base Resolution bwg2001-003.

112961 **Issue 7**

112962 SD5-XCU-ERN-46 is applied, moving this utility to the UUCP Utilities Option Group.

112963 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.



112964 **NAME**

112965 uux — remote command execution

112966 **SYNOPSIS**112967 UU `uux [-jnp] command-string`112968 **DESCRIPTION**

112969 The *uux* utility shall gather zero or more files from various systems, execute a shell pipeline (see  
 112970 [Section 2.9](#), on page 2355) on a specified system, and then send the standard output of the  
 112971 command to a file on a specified system. Only the first command of a pipeline can have a *system-*  
 112972 *name!* prefix. All other commands in the pipeline shall be executed on the system of the first  
 112973 command.

112974 The following restrictions are applicable to the shell pipeline processed by *uux*:

- 112975 • In gathering files from different systems, pathname expansion shall not be performed by  
 112976 *uux*. Thus, a request such as:  
 112977 `uux "c99 remsys!~/*.c"`  
 112978 would attempt to copy the file named literally *\*.c* to the local system.
- 112979 • The redirection operators "*>>*", "*<<*", "*>|*", and "*>&*" shall not be accepted. Any use of  
 112980 these redirection operators shall cause this utility to write an error message describing the  
 112981 problem and exit with a non-zero exit status.
- 112982 • The reserved word *!* cannot be used at the head of the pipeline to modify the exit status.  
 112983 (See the *command-string* operand description below.)
- 112984 • Alias substitution shall not be performed.

112985 A filename can be specified as for *uucp*; it can be an absolute pathname, a pathname preceded by  
 112986 *~name* (which is replaced by the corresponding login directory), a pathname specified as *~/dest*  
 112987 (*dest* is prefixed by the public directory called *PUBDIR*; the actual location of *PUBDIR* is  
 112988 implementation-defined), or a simple filename (which is prefixed by *uux* with the current  
 112989 directory). See *uucp* for the details.

112990 The execution of commands on remote systems shall take place in an execution directory known  
 112991 to the *uucp* system. All files required for the execution shall be put into this directory unless they  
 112992 already reside on that machine. Therefore, the application shall ensure that non-local filenames  
 112993 (without path or machine reference) are unique within the *uux* request.

112994 The *uux* utility shall attempt to get all files to the execution system. For files that are output files,  
 112995 the application shall ensure that the filename is escaped using parentheses.

112996 The remote system shall notify the user by mail if the requested command on the remote system  
 112997 was disallowed or the files were not accessible. This notification can be turned off by the *-n*  
 112998 option.

112999 Typical implementations of this utility require a communications line configured to use XBD  
 113000 [Chapter 11](#) (on page 199), but other communications means may be used. On systems where  
 113001 there are no available communications means (either temporarily or permanently), this utility  
 113002 shall write an error message describing the problem and exit with a non-zero exit status.

113003 The *uux* utility cannot guarantee support for all character encodings in all circumstances. For  
 113004 example, transmission data may be restricted to 7 bits by the underlying network, 8-bit data and  
 113005 filenames need not be portable to non-internationalized systems, and so on. Under these  
 113006 circumstances, it is recommended that only characters defined in the ISO/IEC 646:1991  
 113007 standard International Reference Version (equivalent to ASCII) 7-bit range of characters be used

113008 and that only characters defined in the portable filename character set be used for naming files.

#### 113009 OPTIONS

113010 The *uux* utility shall conform to XBD [Section 12.2](#) (on page 216).

113011 The following options shall be supported:

113012 **-j** Write the job identification string to standard output. This job identification can be  
113013 used by *uustat* to obtain the status or terminate a job.

113014 **-n** Do not notify the user if the command fails.

113015 **-p** Make the standard input to *uux* the standard input to the *command-string*.

#### 113016 OPERANDS

113017 The following operand shall be supported:

113018 *command-string*

113019 A string made up of one or more arguments that are similar to normal command  
113020 arguments, except that the command and any filenames can be prefixed by *system-*  
113021 *name!*. A null *system-name* shall be interpreted as the local system.

#### 113022 STDIN

113023 The standard input shall not be used unless the **'-'** or **-p** option is specified; in those cases, the  
113024 standard input shall be made the standard input of the *command-string*.

#### 113025 INPUT FILES

113026 Input files shall be selected according to the contents of *command-string*.

#### 113027 ENVIRONMENT VARIABLES

113028 The following environment variables shall affect the execution of *uux*:

113029 **LANG** Provide a default value for the internationalization variables that are unset or null.  
113030 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
113031 variables used to determine the values of locale categories.)

113032 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
113033 internationalization variables.

113034 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
113035 characters (for example, single-byte as opposed to multi-byte characters in  
113036 arguments).

113037 **LC\_MESSAGES**

113038 Determine the locale that should be used to affect the format and contents of  
113039 diagnostic messages written to standard error.

113040 **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

#### 113041 ASYNCHRONOUS EVENTS

113042 Default.

#### 113043 STDOUT

113044 The standard output shall not be used unless the **-j** option is specified; in that case, the job  
113045 identification string shall be written to standard output in the following format:

113046 "%s\n", <*jobid*>

113047 **STDERR**

113048 The standard error shall be used only for diagnostic messages.

113049 **OUTPUT FILES**

113050 Output files shall be created or written, or both, according to the contents of *command-string*.

113051 If **-n** is not used, mail files shall be modified following any command or file-access failures on  
113052 the remote system.

113053 **EXTENDED DESCRIPTION**

113054 None.

113055 **EXIT STATUS**

113056 The following exit values shall be returned:

113057 0 Successful completion.

113058 >0 An error occurred.

113059 **CONSEQUENCES OF ERRORS**

113060 Default.

113061 **APPLICATION USAGE**

113062 This utility is part of the UUCP Utilities option and need not be supported by all  
113063 implementations.

113064 Note that, for security reasons, many installations limit the list of commands executable on  
113065 behalf of an incoming request from *uux*. Many sites permit little more than the receipt of mail  
113066 via *uux*.

113067 Any characters special to the command interpreter should be quoted either by quoting the entire  
113068 *command-string* or quoting the special characters as individual arguments.

113069 As noted in *uucp*, shell pattern matching notation characters appearing in pathnames are  
113070 expanded on the appropriate local system. This is done under the control of local settings of  
113071 *LC\_COLLATE* and *LC\_CTYPE*. Thus, care should be taken when using bracketed filename  
113072 patterns, as collation and typing rules may vary from one system to another. Also be aware that  
113073 certain types of expression (that is, equivalence classes, character classes, and collating symbols)  
113074 need not be supported on non-internationalized systems.

113075 **EXAMPLES**

113076 1. The following command gets **file1** from system **a** and **file2** from system **b**, executes *diff* on  
113077 the local system, and puts the results in **file.diff** in the local *PUBDIR* directory. (*PUBDIR*  
113078 is the *uucp* public directory on the local system.)

113079 `uux "!diff a!/usr/file1 b!/a4/file2 >!~/file.diff"`

113080 2. The following command fails because *uux* places all files copied to a system in the same  
113081 working directory. Although the files **xyz** are from two different systems, their filenames  
113082 are the same and conflict.

113083 `uux "!diff a!/usr1/xyz b!/usr2/xyz >!~/xyz.diff"`

113084 3. The following command succeeds (assuming *diff* is permitted on system **a**) because the  
113085 file local to system **a** is not copied to the working directory, and hence does not conflict  
113086 with the file from system **c**.

113087 `uux "a!diff a!/usr/xyz c!/usr/xyz >!~/xyz.diff"`

113088 **RATIONALE**

113089 None.

113090 **FUTURE DIRECTIONS**

113091 None.

113092 **SEE ALSO**113093 [Chapter 2](#) (on page 2335), [uucp](#), [uuencode](#), [uustat](#)113094 XBD [Chapter 8](#) (on page 173), [Chapter 11](#) (on page 199), [Section 12.2](#) (on page 216)113095 **CHANGE HISTORY**

113096 First released in Issue 2.

113097 **Issue 6**

113098 The obsolescent SYNOPSIS is removed.

113099 The normative text is reworded to avoid use of the term “must” for application requirements.

113100 The UN margin code and associated shading are removed from the `-j` option in response to The  
113101 Open Group Base Resolution bwg2001-003.113102 **Issue 7**

113103 SD5-XCU-ERN-46 is applied, moving this utility to the UUCP Utilities Option Group.

113104 **NAME**113105 val — validate SCCS files (**DEVELOPMENT**)113106 **SYNOPSIS**

```
113107 XSI    val -
113108        val [-s] [-m name] [-r SID] [-y type] file...
```

113109 **DESCRIPTION**

113110 The *val* utility shall determine whether the specified *file* is an SCCS file meeting the  
 113111 characteristics specified by the options.

113112 **OPTIONS**

113113 The *val* utility shall conform to XBD [Section 12.2](#) (on page 216), except that the usage of the ‘-’  
 113114 operand is not strictly as intended by the guidelines (that is, reading options and operands from  
 113115 standard input).

113116 The following options shall be supported:

- 113117 **-m name** Specify a *name*, which is compared with the SCCS %M% keyword in *file*; see [get](#).
- 113118 **-r SID** Specify a *SID* (SCCS Identification String), an SCCS delta number. A check shall be  
 113119 made to determine whether the *SID* is ambiguous (for example, **-r 1** is ambiguous  
 113120 because it physically does not exist but implies 1.1, 1.2, and so on, which may exist)  
 113121 or invalid (for example, **-r 1.0** or **-r 1.1.0** are invalid because neither case can exist  
 113122 as a valid delta number). If the *SID* is valid and not ambiguous, a check shall be  
 113123 made to determine whether it actually exists.
- 113124 **-s** Silence the diagnostic message normally written to standard output for any error  
 113125 that is detected while processing each named file on a given command line.
- 113126 **-y type** Specify a *type*, which shall be compared with the SCCS %Y% keyword in *file*; see  
 113127 [get](#).

113128 **OPERANDS**

113129 The following operands shall be supported:

- 113130 *file* A pathname of an existing SCCS file. If exactly one *file* operand appears, and it is  
 113131 ‘-’, the standard input shall be read: each line shall be independently processed  
 113132 as if it were a command line argument list. (However, the line is not subjected to  
 113133 any of the shell word expansions, such as parameter expansion or quote removal.)

113134 **STDIN**

113135 The standard input shall be a text file used only when the *file* operand is specified as ‘-’.

113136 **INPUT FILES**

113137 Any SCCS files processed shall be files of an unspecified format.

113138 **ENVIRONMENT VARIABLES**

113139 The following environment variables shall affect the execution of *val*:

- 113140 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 113141 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 113142 variables used to determine the values of locale categories.)
- 113143 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 113144 internationalization variables.

113145 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 113146 characters (for example, single-byte as opposed to multi-byte characters in  
 113147 arguments and input files).

113148 **LC\_MESSAGES**  
 113149 Determine the locale that should be used to affect the format and contents of  
 113150 diagnostic messages written to standard error, and informative messages written  
 113151 to standard output.

113152 **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

## 113153 ASYNCHRONOUS EVENTS

113154 Default.

## 113155 STDOUT

113156 The standard output shall consist of informative messages about either:

- 113157 1. Each file processed
- 113158 2. Each command line read from standard input

113159 If the standard input is not used, for each *file* operand yielding a discrepancy, the output line  
 113160 shall have the following format:

113161 "%s: %s\n", <pathname>, <unspecified string>

113162 If the standard input is used, for each input line yielding a discrepancy, the output shall have the  
 113163 following format:

113164 "%s\n\n %s: %s\n", <input>, <pathname>, <unspecified string>

113165 where <input> is the input line minus its terminating <newline>.

## 113166 STDERR

113167 Not used.

## 113168 OUTPUT FILES

113169 None.

## 113170 EXTENDED DESCRIPTION

113171 None.

## 113172 EXIT STATUS

113173 The 8-bit code returned by *val* shall be a disjunction of the possible errors; that is, it can be  
 113174 interpreted as a bit string where set bits are interpreted as follows:

113175	0x80	=	Missing file argument.
113176	0x40	=	Unknown or duplicate option.
113177	0x20	=	Corrupted SCCS file.
113178	0x10	=	Cannot open file or file not SCCS.
113179	0x08	=	<i>SID</i> is invalid or ambiguous.
113180	0x04	=	<i>SID</i> does not exist.
113181	0x02	=	%Y%, -y mismatch.
113182	0x01	=	%M%, -m mismatch.

113183 Note that *val* can process two or more files on a given command line and can process multiple  
 113184 command lines (when reading the standard input). In these cases an aggregate code shall be  
 113185 returned: a logical OR of the codes generated for each command line and file processed.

**113186 CONSEQUENCES OF ERRORS**

113187 Default.

**113188 APPLICATION USAGE**

113189 Since the *val* exit status sets the 0x80 bit, shell applications checking "\$?" cannot tell if it  
113190 terminated due to a missing file argument or receipt of a signal.

**113191 EXAMPLES**

113192 In a directory with three SCCS files—*s.x* (of *t* type “text”), *s.y*, and *s.z* (a corrupted file)—the  
113193 following command could produce the output shown:

```
113194 val - <<EOF
113195 -y source s.x
113196 -m y s.y
113197 s.z
113198 EOF
113199 -y source s.x

113200 s.x: %Y%, -y mismatch
113201 s.z
113202 s.z: corrupted SCCS file
```

**113203 RATIONALE**

113204 None.

**113205 FUTURE DIRECTIONS**

113206 None.

**113207 SEE ALSO**

113208 *admin, delta, get, prs*

113209 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

**113210 CHANGE HISTORY**

113211 First released in Issue 2.

**113212 Issue 6**

113213 The Open Group Corrigendum U025/4 is applied, correcting a typographical error in the EXIT  
113214 STATUS.

**113215 Issue 7**

113216 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

113217 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0147 [416] and XCU/TC1-2008/0148  
113218 [416] are applied.

113219 **NAME**

113220 vi — screen-oriented (visual) display editor

113221 **SYNOPSIS**113222 UP `vi [-rR] [-c command] [-t tagstring] [-w size] [file...]`113223 **DESCRIPTION**113224 This utility shall be provided on systems that both support the User Portability Utilities option  
113225 and define the POSIX2\_CHAR\_TERM symbol. On other systems it is optional.113226 The *vi* (visual) utility is a screen-oriented text editor. Only the open and visual modes of the  
113227 editor are described in POSIX.1-2008; see the line editor *ex* for additional editing capabilities  
113228 used in *vi*. The user can switch back and forth between *vi* and *ex* and execute *ex* commands from  
113229 within *vi*.113230 This reference page uses the term *edit buffer* to describe the current working text. No specific  
113231 implementation is implied by this term. All editing changes are performed on the edit buffer,  
113232 and no changes to it shall affect any file until an editor command writes the file.113233 When using *vi*, the terminal screen acts as a window into the editing buffer. Changes made to  
113234 the editing buffer shall be reflected in the screen display; the position of the cursor on the screen  
113235 shall indicate the position within the editing buffer.113236 Certain terminals do not have all the capabilities necessary to support the complete *vi* definition.  
113237 When these commands cannot be supported on such terminals, this condition shall not produce  
113238 an error message such as “not an editor command” or report a syntax error. The implementation  
113239 may either accept the commands and produce results on the screen that are the result of an  
113240 unsuccessful attempt to meet the requirements of this volume of POSIX.1-2008 or report an error  
113241 describing the terminal-related deficiency.113242 **OPTIONS**113243 The *vi* utility shall conform to XBD [Section 12.2](#) (on page 216), except that ‘+’ may be  
113244 recognized as an option delimiter as well as ‘-’.

113245 The following options shall be supported:

113246 **-c *command*** See the *ex* command description of the **-c** option.113247 **-r** See the *ex* command description of the **-r** option.113248 **-R** See the *ex* command description of the **-R** option.113249 **-t *tagstring*** See the *ex* command description of the **-t** option.113250 **-w *size*** See the *ex* command description of the **-w** option.113251 **OPERANDS**113252 See the OPERANDS section of the *ex* command for a description of the operands supported by  
113253 the *vi* command.113254 **STDIN**113255 If standard input is not a terminal device, the results are undefined. The standard input consists  
113256 of a series of commands and input text, as described in the EXTENDED DESCRIPTION section.113257 If a read from the standard input returns an error, or if the editor detects an end-of-file condition  
113258 from the standard input, it shall be equivalent to a SIGHUP asynchronous event.



**113259 INPUT FILES**

113260 See the INPUT FILES section of the *ex* command for a description of the input files supported by  
 113261 the *vi* command.

**113262 ENVIRONMENT VARIABLES**

113263 See the ENVIRONMENT VARIABLES section of the *ex* command for the environment variables  
 113264 that affect the execution of the *vi* command.

**113265 ASYNCHRONOUS EVENTS**

113266 See the ASYNCHRONOUS EVENTS section of the *ex* for the asynchronous events that affect the  
 113267 execution of the *vi* command.

**113268 STDOUT**

113269 If standard output is not a terminal device, undefined results occur.

113270 Standard output may be used for writing prompts to the user, for informational messages, and  
 113271 for writing lines from the file.

**113272 STDERR**

113273 If standard output is not a terminal device, undefined results occur.

113274 The standard error shall be used only for diagnostic messages.

**113275 OUTPUT FILES**

113276 See the OUTPUT FILES section of the *ex* command for a description of the output files  
 113277 supported by the *vi* command.

**113278 EXTENDED DESCRIPTION**

113279 If the terminal does not have the capabilities necessary to support an unspecified portion of the  
 113280 *vi* definition, implementations shall start initially in *ex* mode or open mode. Otherwise, after  
 113281 initialization, *vi* shall be in command mode; text input mode can be entered by one of several  
 113282 commands used to insert or change text. In text input mode, <ESC> can be used to return to  
 113283 command mode; other uses of <ESC> are described later in this section; see [Terminate](#)  
 113284 [Command or Input Mode](#) (on page 3374).

**113285 Initialization in *ex* and *vi***

113286 See [Initialization in \*ex\* and \*vi\*](#) (on page 2690) for a description of *ex* and *vi* initialization for the *vi*  
 113287 utility.

**113288 Command Descriptions in *vi***

113289 The following symbols are used in this reference page to represent arguments to commands.

113290 *buffer* See the description of *buffer* in the EXTENDED DESCRIPTION section of the *ex* utility;  
 113291 see [Command Descriptions in \*ex\*](#) (on page 2699).

113292 In open and visual mode, when a command synopsis shows both [*buffer*] and [*count*]  
 113293 preceding the command name, they can be specified in either order.

113294 *count* A positive integer used as an optional argument to most commands, either to give a  
 113295 repeat count or as a size. This argument is optional and shall default to 1 unless  
 113296 otherwise specified.

113297 The Synopsis lines for the *vi* commands <control>-G, <control>-L, <control>-R,  
 113298 <control>-], %, &, ^, D, m, M, Q, u, U, and ZZ do not have *count* as an optional  
 113299 argument. Regardless, it shall not be an error to specify a *count* to these commands, and  
 113300 any specified *count* shall be ignored.

*motion* An optional trailing argument used by the **!**, **<**, **>**, **c**, **d**, and **y** commands, which is used to indicate the region of text that shall be affected by the command. The motion can be either one of the command characters repeated or one of several other *vi* commands (listed in the following table). Each of the applicable commands specifies the region of text matched by repeating the command; each command that can be used as a motion command specifies the region of text it affects.

Commands that take *motion* arguments operate on either lines or characters, depending on the circumstances. When operating on lines, all lines that fall partially or wholly within the text region specified for the command shall be affected. When operating on characters, only the exact characters in the specified text region shall be affected. Each motion command specifies this individually.

When commands that may be motion commands are not used as motion commands, they shall set the current position to the current line and column as specified.

The following commands shall be valid cursor motion commands:

<apostrophe>	(	-	j	H
<carriage-return>	)	\$	k	L
<comma>	[	%	l	M
<control>-H	]	_	n	N
<control>-N	{	;	t	T
<control>-P	}	?	w	W
<grave-accent>	^	b	B	
<newline>	+	e	E	
<space>		f	F	
<zero>	/	h	G	

Any *count* that is specified to a command that has an associated motion command shall be applied to the motion command. If a *count* is applied to both the command and its associated motion command, the effect shall be multiplicative.

The following symbols are used in this section to specify locations in the edit buffer:

*current character*

The character that is currently indicated by the cursor.

*end of a line*

The point located between the last non-<newline> (if any) and the terminating <newline> of a line. For an empty line, this location coincides with the beginning of the line.

*end of the edit buffer*

The location corresponding to the end of the last line in the edit buffer.

The following symbols are used in this section to specify command actions:

*bigword* In the POSIX locale, *vi* shall recognize four kinds of *bigwords*:

1. A maximal sequence of non-<blank> characters preceded and followed by <blank> characters or the beginning or end of a line or the edit buffer
2. One or more sequential blank lines
3. The first character in the edit buffer

- 113343 4. The last non-`<newline>` in the edit buffer
- 113344 *word* In the POSIX locale, *vi* shall recognize five kinds of words:
- 113345 1. A maximal sequence of letters, digits, and underscores, delimited at both ends
- 113346 by:
- 113347 — Characters other than letters, digits, or underscores
- 113348 — The beginning or end of a line
- 113349 — The beginning or end of the edit buffer
- 113350 2. A maximal sequence of characters other than letters, digits, underscores, or
- 113351 `<blank>` characters, delimited at both ends by:
- 113352 — A letter, digit, underscore
- 113353 — `<blank>` characters
- 113354 — The beginning or end of a line
- 113355 — The beginning or end of the edit buffer
- 113356 3. One or more sequential blank lines
- 113357 4. The first character in the edit buffer
- 113358 5. The last non-`<newline>` in the edit buffer
- 113359 *section boundary*
- 113360 A *section boundary* is one of the following:
- 113361 1. A line whose first character is a `<form-feed>`
- 113362 2. A line whose first character is an open curly brace ( `' { '` )
- 113363 3. A line whose first character is a `<period>` and whose second and third characters
- 113364 match a two-character pair in the **sections** edit option (see *ex*)
- 113365 4. A line whose first character is a `<period>` and whose only other character
- 113366 matches the first character of a two-character pair in the **sections** edit option,
- 113367 where the second character of the two-character pair is a `<space>`
- 113368 5. The first line of the edit buffer
- 113369 6. The last line of the edit buffer if the last line of the edit buffer is empty or if it is a
- 113370 `]]` or `}` command; otherwise, the last non-`<newline>` of the last line of the edit
- 113371 buffer
- 113372 *paragraph boundary*
- 113373 A *paragraph boundary* is one of the following:
- 113374 1. A section boundary
- 113375 2. A line whose first character is a `<period>` and whose second and third characters
- 113376 match a two-character pair in the **paragraphs** edit option (see *ex*)
- 113377 3. A line whose first character is a `<period>` and whose only other character
- 113378 matches the first character of a two-character pair in the *paragraphs* edit option,
- 113379 where the second character of the two-character pair is a `<space>`

#### 4. One or more sequential blank lines

*remembered search direction*

See the description of *remembered search direction* in *ex*.

*sentence boundary*

A *sentence boundary* is one of the following:

1. A paragraph boundary
2. The first non-`<blank>` that occurs after a paragraph boundary
3. The first non-`<blank>` that occurs after a `<period>` (`' . '`), `<exclamation-mark>` (`' ! '`), or `<question-mark>` (`' ? '`), followed by two `<space>` characters or the end of a line; any number of closing parenthesis (`' ) '`), closing brackets (`' ] '`), double-quote (`' " '`), or single-quote (`<apostrophe>`) characters can appear between the punctuation mark and the two `<space>` characters or end-of-line

In the remainder of the description of the *vi* utility, the term “buffer line” refers to a line in the edit buffer and the term “display line” refers to the line or lines on the display screen used to display one buffer line. The term “current line” refers to a specific “buffer line”.

If there are display lines on the screen for which there are no corresponding buffer lines because they correspond to lines that would be after the end of the file, they shall be displayed as a single `<tilde>` (`' ~ '`) character, plus the terminating `<newline>`.

The last line of the screen shall be used to report errors or display informational messages. It shall also be used to display the input for “line-oriented commands” (`/`, `?`, `:`, and `!`). When a line-oriented command is executed, the editor shall enter text input mode on the last line on the screen, using the respective command characters as prompt characters. (In the case of the `!` command, the associated motion shall be entered by the user before the editor enters text input mode.) The line entered by the user shall be terminated by a `<newline>`, a non-`<control>`-V-escaped `<carriage-return>`, or unescaped `<ESC>`. It is unspecified if more characters than require a display width minus one column number of screen columns can be entered.

If any command is executed that overwrites a portion of the screen other than the last line of the screen (for example, the *ex* **suspend** or `!` commands), other than the *ex* **shell** command, the user shall be prompted for a character before the screen is refreshed and the edit session continued.

`<tab>` characters shall take up the number of columns on the screen set by the **tabstop** edit option (see *ex*), unless there are less than that number of columns before the display margin that will cause the displayed line to be folded; in this case, they shall only take up the number of columns up to that boundary.

The cursor shall be placed on the current line and relative to the current column as specified by each command described in the following sections.

In open mode, if the current line is not already displayed, then it shall be displayed.

In visual mode, if the current line is not displayed, then the lines that are displayed shall be expanded, scrolled, or redrawn to cause an unspecified portion of the current line to be displayed. If the screen is redrawn, no more than the number of display lines specified by the value of the **window** edit option shall be displayed (unless the current line cannot be completely displayed in the number of display lines specified by the **window** edit option) and the current line shall be positioned as close to the center of the displayed lines as possible (within the constraints imposed by the distance of the line from the beginning or end of the edit buffer). If the current line is before the first line in the display and the screen is scrolled, an unspecified

portion of the current line shall be placed on the first line of the display. If the current line is after the last line in the display and the screen is scrolled, an unspecified portion of the current line shall be placed on the last line of the display.

In visual mode, if a line from the edit buffer (other than the current line) does not entirely fit into the lines at the bottom of the display that are available for its presentation, the editor may choose not to display any portion of the line. The lines of the display that do not contain text from the edit buffer for this reason shall each consist of a single '@' character.

In visual mode, the editor may choose for unspecified reasons to not update lines in the display to correspond to the underlying edit buffer text. The lines of the display that do not correctly correspond to text from the edit buffer for this reason shall consist of a single '@' character (plus the terminating <newline>), and the <control>-R command shall cause the editor to update the screen to correctly represent the edit buffer.

Open and visual mode commands that set the current column set it to a column position in the display, and not a character position in the line. In this case, however, the column position in the display shall be calculated for an infinite width display; for example, the column related to a character that is part of a line that has been folded onto additional screen lines will be offset from the display line column where the buffer line begins, not from the beginning of a particular display line.

The display cursor column in the display is based on the value of the current column, as follows, with each rule applied in turn:

1. If the current column is after the last display line column used by the displayed line, the display cursor column shall be set to the last display line column occupied by the last non-<newline> in the current line; otherwise, the display cursor column shall be set to the current column.
2. If the character of which some portion is displayed in the display line column specified by the display cursor column requires more than a single display line column:
  - a. If in text input mode, the display cursor column shall be adjusted to the first display line column in which any portion of that character is displayed.
  - b. Otherwise, the display cursor column shall be adjusted to the last display line column in which any portion of that character is displayed.

The current column shall not be changed by these adjustments to the display cursor column.

If an error occurs during the parsing or execution of a *vi* command:

- The terminal shall be alerted. Execution of the *vi* command shall stop, and the cursor (for example, the current line and column) shall not be further modified.
- Unless otherwise specified by the following command sections, it is unspecified whether an informational message shall be displayed.
- Any partially entered *vi* command shall be discarded.
- If the *vi* command resulted from a **map** expansion, all characters from that **map** expansion shall be discarded, except as otherwise specified by the **map** command (see *ex*).
- If the *vi* command resulted from the execution of a buffer, no further commands caused by the execution of the buffer shall be executed.

**Page Backwards**

*Synopsis:*     [*count*] <control>-B

If in open mode, the <control>-B command shall behave identically to the **z** command. Otherwise, if the current line is the first line of the edit buffer, it shall be an error.

If the **window** edit option is less than 3, display a screen where the last line of the display shall be some portion of:

(*current first line*) -1

otherwise, display a screen where the first line of the display shall be some portion of:

(*current first line*) - *count* x ((*window edit option*) -2)

If this calculation would result in a line that is before the first line of the edit buffer, the first line of the display shall display some portion of the first line of the edit buffer.

*Current line:* If no lines from the previous display remain on the screen, set to the last line of the display; otherwise, set to (*line* - the number of new lines displayed on this screen).

*Current column:* Set to non-<blank>.

**Scroll Forward**

*Synopsis:*     [*count*] <control>-D

If the current line is the last line of the edit buffer, it shall be an error.

If no *count* is specified, *count* shall default to the *count* associated with the previous <control>-D or <control>-U command. If there was no previous <control>-D or <control>-U command, *count* shall default to the value of the **scroll** edit option.

If in open mode, write lines starting with the line after the current line, until *count* lines or the last line of the file have been written.

*Current line:* If the current line + *count* is past the last line of the edit buffer, set to the last line of the edit buffer; otherwise, set to the current line + *count*.

*Current column:* Set to non-<blank>.

**Scroll Forward by Line**

*Synopsis:*     [*count*] <control>-E

Display the line *count* lines after the last line currently displayed.

If the last line of the edit buffer is displayed, it shall be an error. If there is no line *count* lines after the last line currently displayed, the last line of the display shall display some portion of the last line of the edit buffer.

*Current line:* Unchanged if the previous current character is displayed; otherwise, set to the first line displayed.

*Current column:* Unchanged.

**Page Forward**

*Synopsis:*     [*count*] <control>-F

If in open mode, the <control>-F command shall behave identically to the **z** command. Otherwise, if the current line is the last line of the edit buffer, it shall be an error.

If the **window** edit option is less than 3, display a screen where the first line of the display shall be some portion of:

(*current last line*) +1

otherwise, display a screen where the first line of the display shall be some portion of:

(*current first line*) + *count* x ((*window edit option*) -2)

If this calculation would result in a line that is after the last line of the edit buffer, the last line of the display shall display some portion of the last line of the edit buffer.

*Current line:* If no lines from the previous display remain on the screen, set to the first line of the display; otherwise, set to (*line* + the number of new lines displayed on this screen).

*Current column:* Set to non-<blank>.

**Display Information**

*Synopsis:*     <control>-G

This command shall be equivalent to the **ex file** command.

**Move Cursor Backwards**

*Synopsis:*     [*count*] <control>-H

                  [*count*] h

                  the current erase character (see **stty**)

If there are no characters before the current character on the current line, it shall be an error. If there are less than *count* previous characters on the current line, *count* shall be adjusted to the number of previous characters on the line.

If used as a motion command:

1. The text region shall be from the character before the starting cursor up to and including the *count*th character before the starting cursor.
2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Unchanged.

*Current column:* Set to (*column* - the number of columns occupied by *count* characters ending with the previous current column).



**Move Down**

*Synopsis:*    [*count*] <newline>  
                   [*count*] <control>-J  
                   [*count*] <control>-M  
                   [*count*] <control>-N  
                   [*count*] j  
                   [*count*] <carriage-return>  
                   [*count*] +

If there are less than *count* lines after the current line in the edit buffer, it shall be an error.

If used as a motion command:

1. The text region shall include the starting line and the next *count* – 1 lines.
2. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

*Current line:* Set to *current line* + *count*.

*Current column:* Set to non-<blank> for the <carriage-return>, <control>-M, and + commands; otherwise, unchanged.

**Clear and Redisplay**

*Synopsis:*    <control>-L

If in open mode, clear the screen and redisplay the current line. Otherwise, clear and redisplay the screen.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Move Up**

*Synopsis:*    [*count*] <control>-P  
                   [*count*] k  
                   [*count*] –

If there are less than *count* lines before the current line in the edit buffer, it shall be an error.

If used as a motion command:

1. The text region shall include the starting line and the previous *count* lines.
2. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

*Current line:* Set to *current line* – *count*.

*Current column:* Set to non-<blank> for the – command; otherwise, unchanged.



**113565 Redraw Screen**

113566 *Synopsis:*      <control>-R

113567 If any lines have been deleted from the display screen and flagged as deleted on the terminal  
113568 using the @ convention (see the beginning of the EXTENDED DESCRIPTION section), they shall  
113569 be redisplayed to match the contents of the edit buffer.

113570 It is unspecified whether lines flagged with @ because they do not fit on the terminal display  
113571 shall be affected.

113572 *Current line:* Unchanged.

113573 *Current column:* Unchanged.

**113574 Scroll Backward**

113575 *Synopsis:*      [count] <control>-U

113576 If the current line is the first line of the edit buffer, it shall be an error.

113577 If no *count* is specified, *count* shall default to the *count* associated with the previous <control>-D  
113578 or <control>-U command. If there was no previous <control>-D or <control>-U command, *count*  
113579 shall default to the value of the **scroll** edit option.

113580 *Current line:* If *count* is greater than the current line, set to 1; otherwise, set to the current line –  
113581 *count*.

113582 *Current column:* Set to non-<blank>.

**113583 Scroll Backward by Line**

113584 *Synopsis:*      [count] <control>-Y

113585 Display the line *count* lines before the first line currently displayed.

113586 If the current line is the first line of the edit buffer, it shall be an error. If this calculation would  
113587 result in a line that is before the first line of the edit buffer, the first line of the display shall  
113588 display some portion of the first line of the edit buffer.

113589 *Current line:* Unchanged if the previous current character is displayed; otherwise, set to the first  
113590 line displayed.

113591 *Current column:* Unchanged.

**113592 Edit the Alternate File**

113593 *Synopsis:*      <control>-^

113594 This command shall be equivalent to the *ex* **edit** command, with the alternate pathname as its  
113595 argument.

**Terminate Command or Input Mode**

*Synopsis:*      <ESC>

If a partial *vi* command (as defined by at least one, non-*count* character) has been entered, discard the *count* and the command character(s).

Otherwise, if no command characters have been entered, and the <ESC> was the result of a map expansion, the terminal shall be alerted and the <ESC> character shall be discarded, but it shall not be an error.

Otherwise, it shall be an error.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Search for tagstring**

*Synopsis:*      <control>-]

If the current character is not a word or <blank>, it shall be an error.

This command shall be equivalent to the *ex tag* command, with the argument to that command defined as follows.

If the current character is a <blank>:

1. Skip all <blank> characters after the cursor up to the end of the line.
2. If the end of the line is reached, it shall be an error.

Then, the argument to the *ex tag* command shall be the current character and all subsequent characters, up to the first non-word character or the end of the line.

**Move Cursor Forward**

*Synopsis:*      [*count*] <space>  
                 [*count*] 1 (ell)

If there are less than *count* non-<newline> characters after the cursor on the current line, *count* shall be adjusted to the number of non-<newline> characters after the cursor on the line.

If used as a motion command:

1. If the current or *count*th character after the cursor is the last non-<newline> in the line, the text region shall be comprised of the current character up to and including the last non-<newline> in the line. Otherwise, the text region shall be from the current character up to, but not including, the *count*th character after the cursor.
2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

If there are no non-<newline> characters after the current character on the current line, it shall be an error.

*Current line:* Unchanged.

*Current column:* Set to the last column that displays any portion of the *count*th character after the current character.

**Replace Text with Results from Shell Command**

*Synopsis:*     [*count*] ! *motion shell-commands* <newline>

If the motion command is the ! command repeated:

1. If the edit buffer is empty and no *count* was supplied, the command shall be the equivalent of the *ex :read !* command, with the text input, and no text shall be copied to any buffer.
2. Otherwise:
  - a. If there are less than *count* –1 lines after the current line in the edit buffer, it shall be an error.
  - b. The text region shall be from the current line up to and including the next *count* –1 lines.

Otherwise, the text region shall be the lines in which any character of the text region specified by the motion command appear.

Any text copied to a buffer shall be in line mode.

This command shall be equivalent to the *ex !* command for the specified lines.

**Move Cursor to End-of-Line**

*Synopsis:*     [*count*] \$

It shall be an error if there are less than (*count* –1) lines after the current line in the edit buffer.

If used as a motion command:

1. If *count* is 1:
  - a. It shall be an error if the line is empty.
  - b. Otherwise, the text region shall consist of all characters from the starting cursor to the last non-<newline> in the line, inclusive, and any text copied to a buffer shall be in character mode.
2. Otherwise, if the starting cursor position is at or before the first non-<blank> in the line, the text region shall consist of the current and the next *count* –1 lines, and any text saved to a buffer shall be in line mode.
3. Otherwise, the text region shall consist of all characters from the starting cursor to the last non-<newline> in the line that is *count* –1 lines forward from the current line, and any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Set to the *current line* + *count* –1.

*Current column:* The current column is set to the last display line column of the last non-<newline> in the line, or column position 1 if the line is empty.

The current column shall be adjusted to be on the last display line column of the last non-<newline> of the current line as subsequent commands change the current line, until a command changes the current column.

**Move to Matching Character***Synopsis:*     %

If the character at the current position is not a parenthesis, bracket, or curly brace, search forward in the line to the first one of those characters. If no such character is found, it shall be an error.

The matching character shall be the parenthesis, bracket, or curly brace matching the parenthesis, bracket, or curly brace, respectively, that was at the current position or that was found on the current line.

Matching shall be determined as follows, for an open parenthesis:

1. Set a counter to 1.
2. Search forwards until a parenthesis is found or the end of the edit buffer is reached.
3. If the end of the edit buffer is reached, it shall be an error.
4. If an open parenthesis is found, increment the counter by 1.
5. If a close parenthesis is found, decrement the counter by 1.
6. If the counter is zero, the current character is the matching character.

Matching for a close parenthesis shall be equivalent, except that the search shall be backwards, from the starting character to the beginning of the buffer, a close parenthesis shall increment the counter by 1, and an open parenthesis shall decrement the counter by 1.

Matching for brackets and curly braces shall be equivalent, except that searching shall be done for open and close brackets or open and close curly braces. It is implementation-defined whether other characters are searched for and matched as well.

If used as a motion command:

1. If the matching cursor was after the starting cursor in the edit buffer, and the starting cursor position was at or before the first non-`<blank>` non-`<newline>` in the starting line, and the matching cursor position was at or after the last non-`<blank>` non-`<newline>` in the matching line, the text region shall consist of the current line to the matching line, inclusive, and any text copied to a buffer shall be in line mode.
2. If the matching cursor was before the starting cursor in the edit buffer, and the starting cursor position was at or after the last non-`<blank>` non-`<newline>` in the starting line, and the matching cursor position was at or before the first non-`<blank>` non-`<newline>` in the matching line, the text region shall consist of the current line to the matching line, inclusive, and any text copied to a buffer shall be in line mode.
3. Otherwise, the text region shall consist of the starting character to the matching character, inclusive, and any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Set to the line where the matching character is located.

*Current column:* Set to the last column where any portion of the matching character is displayed.

**Repeat Substitution***Synopsis:*     &

Repeat the previous substitution command. This command shall be equivalent to the *ex &* command with the current line as its addresses, and without *options*, *count*, or *flags*.

**Return to Previous Context at Beginning of Line***Synopsis:*     ' *character*

It shall be an error if there is no line in the edit buffer marked by *character*.

If used as a motion command:

1. If the starting cursor is after the marked cursor, then the locations of the starting cursor and the marked cursor in the edit buffer shall be logically swapped.
2. The text region shall consist of the starting line up to and including the marked line, and any text copied to a buffer shall be in line mode.

If not used as a motion command:

*Current line:* Set to the line referenced by the mark.

*Current column:* Set to non-<blank>.

**Return to Previous Context***Synopsis:*     ' *character*

It shall be an error if the marked line is no longer in the edit buffer. If the marked line no longer contains a character in the saved numbered character position, it shall be as if the marked position is the first non-<blank>.

If used as a motion command:

1. It shall be an error if the marked cursor references the same character in the edit buffer as the starting cursor.
2. If the starting cursor is after the marked cursor, then the locations of the starting cursor and the marked cursor in the edit buffer shall be logically swapped.
3. If the starting line is empty or the starting cursor is at or before the first non-<blank> non-<newline> of the starting line, and the marked cursor line is empty or the marked cursor references the first character of the marked cursor line, the text region shall consist of all lines containing characters from the starting cursor to the line before the marked cursor line, inclusive, and any text copied to a buffer shall be in line mode.
4. Otherwise, if the marked cursor line is empty or the marked cursor references a character at or before the first non-<blank> non-<newline> of the marked cursor line, the region of text shall be from the starting cursor to the last non-<newline> of the line before the marked cursor line, inclusive, and any text copied to a buffer shall be in character mode.
5. Otherwise, the region of text shall be from the starting cursor (inclusive), to the marked cursor (exclusive), and any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Set to the line referenced by the mark.

*Current column:* Set to the last column in which any portion of the character referenced by the

mark is displayed.

## Return to Previous Section

*Synopsis:*     [*count*] [[

Move the cursor backward through the edit buffer to the first character of the previous section boundary, *count* times.

If used as a motion command:

1. If the starting cursor was at the first character of the starting line or the starting line was empty, and the first character of the boundary was the first character of the boundary line, the text region shall consist of the current line up to and including the line where the *count*th next boundary starts, and any text copied to a buffer shall be in line mode.
2. If the boundary was the last line of the edit buffer or the last non-<newline> of the last line of the edit buffer, the text region shall consist of the last character in the edit buffer up to and including the starting character, and any text saved to a buffer shall be in character mode.
3. Otherwise, the text region shall consist of the starting character up to but not including the first character in the *count*th next boundary, and any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Set to the line where the *count*th next boundary in the edit buffer starts.

*Current column:* Set to the last column in which any portion of the first character of the *count*th next boundary is displayed, or column position 1 if the line is empty.

## Move to Next Section

*Synopsis:*     [*count*] ]]

Move the cursor forward through the edit buffer to the first character of the next section boundary, *count* times.

If used as a motion command:

1. If the starting cursor was at the first character of the starting line or the starting line was empty, and the first character of the boundary was the first character of the boundary line, the text region shall consist of the current line up to and including the line where the *count*th previous boundary starts, and any text copied to a buffer shall be in line mode.
2. If the boundary was the first line of the edit buffer, the text region shall consist of the first character in the edit buffer up to but not including the starting character, and any text copied to a buffer shall be in character mode.
3. Otherwise, the text region shall consist of the first character in the *count*th previous section boundary up to but not including the starting character, and any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Set to the line where the *count*th previous boundary in the edit buffer starts.

*Current column:* Set to the last column in which any portion of the first character of the *count*th previous boundary is displayed, or column position 1 if the line is empty.

**Move to First Non-<blank> Position on Current Line***Synopsis:*     `^`

If used as a motion command:

1. If the line has no non-<blank> non-<newline> characters, or if the cursor is at the first non-<blank> non-<newline> of the line, it shall be an error.
2. If the cursor is before the first non-<blank> non-<newline> of the line, the text region shall be comprised of the current character, up to, but not including, the first non-<blank> non-<newline> of the line.
3. If the cursor is after the first non-<blank> non-<newline> of the line, the text region shall be from the character before the starting cursor up to and including the first non-<blank> non-<newline> of the line.
4. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Unchanged.*Current column:* Set to non-<blank>.**Current and Line Above***Synopsis:*     `[count] _`If there are less than *count* -1 lines after the current line in the edit buffer, it shall be an error.

If used as a motion command:

1. If *count* is less than 2, the text region shall be the current line.
2. Otherwise, the text region shall include the starting line and the next *count* -1 lines.
3. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

*Current line:* Set to current line + *count* -1.*Current column:* Set to non-<blank>.**Move Back to Beginning of Sentence***Synopsis:*     `[count] (`

Move backward to the beginning of a sentence. This command shall be equivalent to the `[]` command, with the exception that sentence boundaries shall be used instead of section boundaries.

**Move Forward to Beginning of Sentence***Synopsis:*     `[count] )`

Move forward to the beginning of a sentence. This command shall be equivalent to the `][]` command, with the exception that sentence boundaries shall be used instead of section boundaries.

**Move Back to Preceding Paragraph**

*Synopsis:*     [*count*] {

Move back to the beginning of the preceding paragraph. This command shall be equivalent to the `[[` command, with the exception that paragraph boundaries shall be used instead of section boundaries.

**Move Forward to Next Paragraph**

*Synopsis:*     [*count*] }

Move forward to the beginning of the next paragraph. This command shall be equivalent to the `]]` command, with the exception that paragraph boundaries shall be used instead of section boundaries.

**Move to Specific Column Position**

*Synopsis:*     [*count*] |

For the purposes of this command, lines that are too long for the current display and that have been folded shall be treated as having a single, 1-based, number of columns.

If there are less than *count* columns in which characters from the current line are displayed on the screen, *count* shall be adjusted to be the last column in which any portion of the line is displayed on the screen.

If used as a motion command:

1. If the line is empty, or the cursor character is the same as the character on the *count*th column of the line, it shall be an error.
2. If the cursor is before the *count*th column of the line, the text region shall be comprised of the current character, up to but not including the character on the *count*th column of the line.
3. If the cursor is after the *count*th column of the line, the text region shall be from the character before the starting cursor up to and including the character on the *count*th column of the line.
4. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Unchanged.

*Current column:* Set to the last column in which any portion of the character that is displayed in the *count* column of the line is displayed.

**Reverse Find Character**

*Synopsis:*     [*count*] ,

If the last **F**, **f**, **T**, or **t** command was **F**, **f**, **T**, or **t**, this command shall be equivalent to an **f**, **F**, **t**, or **T** command, respectively, with the specified *count* and the same search character.

If there was no previous **F**, **f**, **T**, or **t** command, it shall be an error.



**Repeat**

*Synopsis:*      [*count*] .

Repeat the last **!**, **<**, **>**, **A**, **C**, **D**, **I**, **J**, **O**, **P**, **R**, **S**, **X**, **Y**, **a**, **c**, **d**, **i**, **o**, **p**, **r**, **s**, **x**, **y**, or **~** command. It shall be an error if none of these commands have been executed. Commands (other than commands that enter text input mode) executed as a result of map expansions, shall not change the value of the last repeatable command.

Repeated commands with associated motion commands shall repeat the motion command as well; however, any specified *count* shall replace the *count*(s) that were originally specified to the repeated command or its associated motion command.

If the motion component of the repeated command is **f**, **F**, **t**, or **T**, the repeated command shall not set the remembered search character for the **;** and **,** commands.

If the repeated command is **p** or **P**, and the buffer associated with that command was a numeric buffer named with a number less than 9, the buffer associated with the repeated command shall be set to be the buffer named by the name of the previous buffer logically incremented by 1.

If the repeated character is a text input command, the input text associated with that command is repeated literally:

- Input characters are neither macro or abbreviation-expanded.
- Input characters are not interpreted in any special way with the exception that <newline>, <carriage-return>, and <control>-T behave as described in [Input Mode Commands in vi](#) (on page 3399).

*Current line:* Set as described for the repeated command.

*Current column:* Set as described for the repeated command.

**Find Regular Expression**

*Synopsis:*      /

If the input line contains no non-<newline> characters, it shall be equivalent to a line containing only the last regular expression encountered. The enhanced regular expressions supported by *vi* are described in [Regular Expressions in ex](#) (on page 2723).

Otherwise, the line shall be interpreted as one or more regular expressions, optionally followed by an address offset or a *vi* **z** command.

If the regular expression is not the last regular expression on the line, or if a line offset or **z** command is specified, the regular expression shall be terminated by an unescaped **'/'** character, which shall not be used as part of the regular expression. If the regular expression is not the first regular expression on the line, it shall be preceded by zero or more <blank> characters, a <semicolon>, zero or more <blank> characters, and a leading **'/'** character, which shall not be interpreted as part of the regular expression. It shall be an error to precede any regular expression with any characters other than these.

Each search shall begin from the character after the first character of the last match (or, if it is the first search, after the cursor). If the **wraps**can edit option is set, the search shall continue to the character before the starting cursor character; otherwise, to the end of the edit buffer. It shall be an error if any search fails to find a match, and an informational message to this effect shall be displayed.

An optional address offset (see [Addressing in ex](#), on page 2692) can be specified after the last regular expression by including a trailing **'/'** character after the regular expression and

specifying the address offset. This offset will be from the line containing the match for the last regular expression specified. It shall be an error if the line offset would indicate a line address less than 1 or greater than the last line in the edit buffer. An address offset of zero shall be supported. It shall be an error to follow the address offset with any other characters than <blank> characters.

If not used as a motion command, an optional **z** command (see [Redraw Window](#), on page 3398) can be specified after the last regular expression by including a trailing ' / ' character after the regular expression, zero or more <blank> characters, a ' z ', zero or more <blank> characters, an optional new **window** edit option value, zero or more <blank> characters, and a location character. The effect shall be as if the **z** command was executed after the / command. It shall be an error to follow the **z** command with any other characters than <blank> characters.

The remembered search direction shall be set to forward.

If used as a motion command:

1. It shall be an error if the last match references the same character in the edit buffer as the starting cursor.
2. If any address offset is specified, the last match shall be adjusted by the specified offset as described previously.
3. If the starting cursor is after the last match, then the locations of the starting cursor and the last match in the edit buffer shall be logically swapped.
4. If any address offset is specified, the text region shall consist of all lines containing characters from the starting cursor to the last match line, inclusive, and any text copied to a buffer shall be in line mode.
5. Otherwise, if the starting line is empty or the starting cursor is at or before the first non-<blank> non-<newline> of the starting line, and the last match line is empty or the last match starts at the first character of the last match line, the text region shall consist of all lines containing characters from the starting cursor to the line before the last match line, inclusive, and any text copied to a buffer shall be in line mode.
6. Otherwise, if the last match line is empty or the last match begins at a character at or before the first non-<blank> non-<newline> of the last match line, the region of text shall be from the current cursor to the last non-<newline> of the line before the last match line, inclusive, and any text copied to a buffer shall be in character mode.
7. Otherwise, the region of text shall be from the current cursor (inclusive), to the first character of the last match (exclusive), and any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* If a match is found, set to the last matched line plus the address offset, if any; otherwise, unchanged.

*Current column:* Set to the last column on which any portion of the first character in the last matched string is displayed, if a match is found; otherwise, unchanged.

**Move to First Character in Line**

*Synopsis:*     0   (zero)

Move to the first character on the current line. The character '0' shall not be interpreted as a command if it is immediately preceded by a digit.

If used as a motion command:

1. If the cursor character is the first character in the line, it shall be an error.
2. The text region shall be from the character before the cursor character up to and including the first character in the line.
3. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Unchanged.

*Current column:* The last column in which any portion of the first character in the line is displayed, or if the line is empty, unchanged.

**Execute an ex Command**

*Synopsis:*     :

Execute one or more *ex* commands.

If any portion of the screen other than the last line of the screen was overwritten by any *ex* command (except **shell**), *vi* shall display a message indicating that it is waiting for an input from the user, and shall then read a character. This action may also be taken for other, unspecified reasons.

If the next character entered is a ':', another *ex* command shall be accepted and executed. Any other character shall cause the screen to be refreshed and *vi* shall return to command mode.

*Current line:* As specified for the *ex* command.

*Current column:* As specified for the *ex* command.

**Repeat Find**

*Synopsis:*     [count] ;

This command shall be equivalent to the last **F**, **f**, **T**, or **t** command, with the specified *count*, and with the same search character used for the last **F**, **f**, **T**, or **t** command. If there was no previous **F**, **f**, **T**, or **t** command, it shall be an error.

**Shift Left**

*Synopsis:*     [count] < *motion*

If the motion command is the < command repeated:

1. If there are less than *count* - 1 lines after the current line in the edit buffer, it shall be an error.
2. The text region shall be from the current line, up to and including the next *count* - 1 lines.

Shift any line in the text region specified by the *count* and motion command one shiftwidth (see the *ex* **shiftwidth** option) toward the start of the line, as described by the *ex* < command. The unshifted lines shall be copied to the unnamed buffer in line mode.

113977 *Current line*: If the motion was from the current cursor position toward the end of the edit buffer,  
 113978 unchanged. Otherwise, set to the first line in the edit buffer that is part of the text region  
 113979 specified by the motion command.

113980 *Current column*: Set to non-<blank>.

## 113981 **Shift Right**

113982 *Synopsis:*     `[count] > motion`

113983 If the motion command is the `>` command repeated:

- 113984     1. If there are less than *count* - 1 lines after the current line in the edit buffer, it shall be an  
 113985       error.
- 113986     2. The text region shall be from the current line, up to and including the next *count* - 1 lines.

113987 Shift any line with characters in the text region specified by the *count* and motion command one  
 113988 shiftwidth (see the *ex* **shiftwidth** option) away from the start of the line, as described by the *ex* `>`  
 113989 command. The unshifted lines shall be copied into the unnamed buffer in line mode.

113990 *Current line*: If the motion was from the current cursor position toward the end of the edit buffer,  
 113991 unchanged. Otherwise, set to the first line in the edit buffer that is part of the text region  
 113992 specified by the motion command.

113993 *Current column*: Set to non-<blank>.

## 113994 **Scan Backwards for Regular Expression**

113995 *Synopsis:*     `?`

113996 Scan backwards; the `?` command shall be equivalent to the `/` command (see [Find Regular](#)  
 113997 [Expression](#), on page 3381) with the following exceptions:

- 113998     1. The input prompt shall be a `' ? '`.
- 113999     2. Each search shall begin from the character before the first character of the last match (or, if  
 114000       it is the first search, the character before the cursor character).
- 114001     3. The search direction shall be from the cursor toward the beginning of the edit buffer, and  
 114002       the **wrapscan** edit option shall affect whether the search wraps to the end of the edit  
 114003       buffer and continues.
- 114004     4. The remembered search direction shall be set to backward.

## 114005 **Execute**

114006 *Synopsis:*     `@buffer`

114007 If the *buffer* is specified as `@`, the last buffer executed shall be used. If no previous buffer has been  
 114008 executed, it shall be an error.

114009 Behave as if the contents of the named buffer were entered as standard input. After each line of a  
 114010 line-mode buffer, and all but the last line of a character mode buffer, behave as if a `<newline>`  
 114011 were entered as standard input.

114012 If an error occurs during this process, an error message shall be written, and no more characters  
 114013 resulting from the execution of this command shall be processed.

114014 If a *count* is specified, behave as if that count were entered as user input before the characters  
 114015 from the `@` buffer were entered.

114016 *Current line*: As specified for the individual commands.

114017 *Current column*: As specified for the individual commands.

## 114018 **Reverse Case**

114019 *Synopsis*:     [*count*] ~

114020 Reverse the case of the current character and the next *count* -1 characters, such that lowercase  
114021 characters that have uppercase counterparts shall be changed to uppercase characters, and  
114022 uppercase characters that have lowercase counterparts shall be changed to lowercase characters,  
114023 as prescribed by the current locale. No other characters shall be affected by this command.

114024 If there are less than *count* -1 characters after the cursor in the edit buffer, *count* shall be adjusted  
114025 to the number of characters after the cursor in the edit buffer minus 1.

114026 For the purposes of this command, the next character after the last non-<newline> on the line  
114027 shall be the next character in the edit buffer.

114028 *Current line*: Set to the line including the (*count*-1)th character after the cursor.

114029 *Current column*: Set to the last column in which any portion of the (*count*-1)th character after the  
114030 cursor is displayed.

## 114031 **Append**

114032 *Synopsis*:     [*count*] a

114033 Enter text input mode after the current cursor position. No characters already in the edit buffer  
114034 shall be affected by this command. A *count* shall cause the input text to be appended *count* -1  
114035 more times to the end of the input.

114036 *Current line/column*: As specified for the text input commands (see [Input Mode Commands in vi](#),  
114037 on page 3399).

## 114038 **Append at End-of-Line**

114039 *Synopsis*:     [*count*] A

114040 This command shall be equivalent to the *vi* command:

114041 \$ [ *count* ] a

114042 (see [Append](#)).

## 114043 **Move Backward to Preceding Word**

114044 *Synopsis*:     [*count*] b

114045 With the exception that words are used as the delimiter instead of bigwords, this command shall  
114046 be equivalent to the **B** command.

**Move Backward to Preceding Bigword**

*Synopsis:*     [*count*] B

If the edit buffer is empty or the cursor is on the first character of the edit buffer, it shall be an error. If less than *count* bigwords begin between the cursor and the start of the edit buffer, *count* shall be adjusted to the number of bigword beginnings between the cursor and the start of the edit buffer.

If used as a motion command:

1. The text region shall be from the first character of the *count*th previous bigword beginning up to but not including the cursor character.
2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Set to the line containing the *current column*.

*Current column:* Set to the last column upon which any part of the first character of the *count*th previous bigword is displayed.

**Change**

*Synopsis:*     [*buffer*][*count*] c *motion*

If the motion command is the **c** command repeated:

1. The buffer text shall be in line mode.
2. If there are less than *count* - 1 lines after the current line in the edit buffer, it shall be an error.
3. The text region shall be from the current line up to and including the next *count* - 1 lines.

Otherwise, the buffer text mode and text region shall be as specified by the motion command.

The replaced text shall be copied into *buffer*, if specified, and into the unnamed buffer. If the text to be replaced contains characters from more than a single line, or the buffer text is in line mode, the replaced text shall be copied into the numeric buffers as well.

If the buffer text is in line mode:

1. Any lines that contain characters in the region shall be deleted, and the editor shall enter text input mode at the beginning of a new line which shall replace the first line deleted.
2. If the **autoindent** edit option is set, **autoindent** characters equal to the **autoindent** characters on the first line deleted shall be inserted as if entered by the user.

Otherwise, if characters from more than one line are in the region of text:

1. The text shall be deleted.
2. Any text remaining in the last line in the text region shall be appended to the first line in the region, and the last line in the region shall be deleted.
3. The editor shall enter text input mode after the last character not deleted from the first line in the text region, if any; otherwise, on the first column of the first line in the region.

Otherwise:

1. If the glyph for ' \$ ' is smaller than the region, the end of the region shall be marked with a ' \$ '.
2. The editor shall enter text input mode, overwriting the region of text.

*Current line/column:* As specified for the text input commands (see [Input Mode Commands in vi](#), on page 3399).

### Change to End-of-Line

*Synopsis:*     [buffer][count] C

This command shall be equivalent to the *vi* command:

[buffer][count] c\$

See the **c** command.

### Delete

*Synopsis:*     [buffer][count] d motion

If the motion command is the **d** command repeated:

1. The buffer text shall be in line mode.
2. If there are less than *count* -1 lines after the current line in the edit buffer, it shall be an error.
3. The text region shall be from the current line up to and including the next *count* -1 lines.

Otherwise, the buffer text mode and text region shall be as specified by the motion command.

If in open mode, and the current line is deleted, and the line remains on the display, an '@' character shall be displayed as the first glyph of that line.

Delete the region of text into *buffer*, if specified, and into the unnamed buffer. If the text to be deleted contains characters from more than a single line, or the buffer text is in line mode, the deleted text shall be copied into the numeric buffers, as well.

*Current line:* Set to the first text region line that appears in the edit buffer, unless that line has been deleted, in which case it shall be set to the last line in the edit buffer, or line 1 if the edit buffer is empty.

*Current column:*

1. If the line is empty, set to column position 1.
2. Otherwise, if the buffer text is in line mode or the motion was from the cursor toward the end of the edit buffer:
  - a. If a character from the current line is displayed in the current column, set to the last column that displays any portion of that character.
  - b. Otherwise, set to the last column in which any portion of any character in the line is displayed.
3. Otherwise, if a character is displayed in the column that began the text region, set to the last column that displays any portion of that character.



4. Otherwise, set to the last column in which any portion of any character in the line is displayed.

### Delete to End-of-Line

*Synopsis:*     [*buffer*] D

Delete the text from the current position to the end of the current line; equivalent to the *vi* command:

[*buffer*] d\$

### Move to End-of-Word

*Synopsis:*     [*count*] e

With the exception that words are used instead of bigwords as the delimiter, this command shall be equivalent to the **E** command.

### Move to End-of-Bigword

*Synopsis:*     [*count*] E

If the edit buffer is empty it shall be an error. If less than *count* bigwords end between the cursor and the end of the edit buffer, *count* shall be adjusted to the number of bigword endings between the cursor and the end of the edit buffer.

If used as a motion command:

1. The text region shall be from the last character of the *count*th next bigword up to and including the cursor character.
2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Set to the line containing the current column.

*Current column:* Set to the last column upon which any part of the last character of the *count*th next bigword is displayed.

### Find Character in Current Line (Forward)

*Synopsis:*     [*count*] f *character*

It shall be an error if *count* occurrences of the character do not occur after the cursor in the line.

If used as a motion command:

1. The text range shall be from the cursor character up to and including the *count*th occurrence of the specified character after the cursor.
2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Unchanged.

*Current column:* Set to the last column in which any portion of the *count*th occurrence of the specified character after the cursor appears in the line.



**Find Character in Current Line (Reverse)**

*Synopsis:*     [*count*] F *character*

It shall be an error if *count* occurrences of the character do not occur before the cursor in the line.

If used as a motion command:

1. The text region shall be from the *count*th occurrence of the specified character before the cursor, up to, but not including the cursor character.
2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Unchanged.

*Current column:* Set to the last column in which any portion of the *count*th occurrence of the specified character before the cursor appears in the line.

**Move to Line**

*Synopsis:*     [*count*] G

If *count* is not specified, it shall default to the last line of the edit buffer. If *count* is greater than the last line of the edit buffer, it shall be an error.

If used as a motion command:

1. The text region shall be from the cursor line up to and including the specified line.
2. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

*Current line:* Set to *count* if *count* is specified; otherwise, the last line.

*Current column:* Set to non-<blank>.

**Move to Top of Screen**

*Synopsis:*     [*count*] H

If the beginning of the line *count* greater than the first line of which any portion appears on the display does not exist, it shall be an error.

If used as a motion command:

1. If in open mode, the text region shall be the current line.
2. Otherwise, the text region shall be from the starting line up to and including (the first line of the display + *count* - 1).
3. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

If in open mode, this command shall set the current column to non-<blank> and do nothing else.

Otherwise, it shall set the current line and current column as follows.

*Current line:* Set to (the first line of the display + *count* - 1).

*Current column:* Set to non-<blank>.

**Insert Before Cursor**

*Synopsis:*     [*count*] i

Enter text input mode before the current cursor position. No characters already in the edit buffer shall be affected by this command. A *count* shall cause the input text to be appended *count* -1 more times to the end of the input.

*Current line/column:* As specified for the text input commands (see [Input Mode Commands in vi](#), on page 3399).

**Insert at Beginning of Line**

*Synopsis:*     [*count*] I

This command shall be equivalent to the *vi* command `^[count]i`.

**Join**

*Synopsis:*     [*count*] J

If the current line is the last line in the edit buffer, it shall be an error.

This command shall be equivalent to the *ex* **join** command with no addresses, and an *ex* command *count* value of 1 if *count* was not specified or if a *count* of 1 was specified, and an *ex* command *count* value of *count* -1 for any other value of *count*, except that the current line and column shall be set as follows.

*Current line:* Unchanged.

*Current column:* The last column in which any portion of the character following the last character in the initial line is displayed, or the last non-<newline> in the line if no characters were appended.

**Move to Bottom of Screen**

*Synopsis:*     [*count*] L

If the beginning of the line *count* less than the last line of which any portion appears on the display does not exist, it shall be an error.

If used as a motion command:

1. If in open mode, the text region shall be the current line.
2. Otherwise, the text region shall include all lines from the starting cursor line to (the last line of the display -(*count* -1)).
3. Any text copied to a buffer shall be in line mode.

If not used as a motion command:

1. If in open mode, this command shall set the current column to non-<blank> and do nothing else.
2. Otherwise, it shall set the current line and current column as follows.

*Current line:* Set to (the last line of the display -(*count* -1)).

*Current column:* Set to non-<blank>.

114226 **Mark Position**114227 *Synopsis:*    m letter

114228 This command shall be equivalent to the *ex mark* command with the specified character as an  
 114229 argument.

114230 **Move to Middle of Screen**114231 *Synopsis:*    M

114232 The middle line of the display shall be calculated as follows:

114233 (the top line of the display) + (((number of lines displayed) + 1) / 2) - 1

114234 If used as a motion command:

- 114235       1. If in open mode, the text region shall be the current line.
- 114236       2. Otherwise, the text region shall include all lines from the starting cursor line up to and  
 114237       including the middle line of the display.
- 114238       3. Any text copied to a buffer shall be in line mode.

114239 If not used as a motion command:

114240 If in open mode, this command shall set the current column to non-&lt;blank&gt; and do nothing else.

114241 Otherwise, it shall set the current line and current column as follows.

114242 *Current line:* Set to the middle line of the display.114243 *Current column:* Set to non-<blank>.114244 **Repeat Regular Expression Find (Forward)**114245 *Synopsis:*    n

114246 If the remembered search direction was forward, the *n* command shall be equivalent to the *vi /*  
 114247 command with no characters entered by the user. Otherwise, it shall be equivalent to the *vi ?*  
 114248 command with no characters entered by the user.

114249 If the *n* command is used as a motion command for the *!* command, the editor shall not enter  
 114250 text input mode on the last line on the screen, and shall behave as if the user entered a single  
 114251 ' ! ' character as the text input.

114252 **Repeat Regular Expression Find (Reverse)**114253 *Synopsis:*    N

114254 Scan for the next match of the last pattern given to */* or *?*, but in the reverse direction; this is the  
 114255 reverse of *n*.

114256 If the remembered search direction was forward, the *N* command shall be equivalent to the *vi ?*  
 114257 command with no characters entered by the user. Otherwise, it shall be equivalent to the *vi /*  
 114258 command with no characters entered by the user. If the *N* command is used as a motion  
 114259 command for the *!* command, the editor shall not enter text input mode on the last line on the  
 114260 screen, and shall behave as if the user entered a single *!* character as the text input.

**Insert Empty Line Below**

*Synopsis:*      ○

Enter text input mode in a new line appended after the current line. A *count* shall cause the input text to be appended *count* –1 more times to the end of the already added text, each time starting on a new, appended line.

*Current line/column:* As specified for the text input commands (see [Input Mode Commands in vi](#), on page 3399).

**Insert Empty Line Above**

*Synopsis:*      ○

Enter text input mode in a new line inserted before the current line. A *count* shall cause the input text to be appended *count* –1 more times to the end of the already added text, each time starting on a new, appended line.

*Current line/column:* As specified for the text input commands (see [Input Mode Commands in vi](#), on page 3399).

**Put from Buffer Following**

*Synopsis:*      [*buffer*] p

If no *buffer* is specified, the unnamed buffer shall be used.

If the buffer text is in line mode, the text shall be appended below the current line, and each line of the buffer shall become a new line in the edit buffer. A *count* shall cause the buffer text to be appended *count* –1 more times to the end of the already added text, each time starting on a new, appended line.

If the buffer text is in character mode, the text shall be appended into the current line after the cursor, and each line of the buffer other than the first and last shall become a new line in the edit buffer. A *count* shall cause the buffer text to be appended *count* –1 more times to the end of the already added text, each time starting after the last added character.

*Current line:* If the buffer text is in line mode, set the line to line +1; otherwise, unchanged.

*Current column:* If the buffer text is in line mode:

1. If there is a non-<blank> in the first line of the buffer, set to the last column on which any portion of the first non-<blank> in the line is displayed.
2. If there is no non-<blank> in the first line of the buffer, set to the last column on which any portion of the last non-<newline> in the first line of the buffer is displayed.

If the buffer text is in character mode:

1. If the text in the buffer is from more than a single line, then set to the last column on which any portion of the first character from the buffer is displayed.
2. Otherwise, if the buffer is the unnamed buffer, set to the last column on which any portion of the last character from the buffer is displayed.
3. Otherwise, set to the first column on which any portion of the first character from the buffer is displayed.

**Put from Buffer Before**

*Synopsis:*     [*buffer*] P

If no *buffer* is specified, the unnamed buffer shall be used.

If the buffer text is in line mode, the text shall be inserted above the current line, and each line of the buffer shall become a new line in the edit buffer. A *count* shall cause the buffer text to be appended *count* -1 more times to the end of the already added text, each time starting on a new, appended line.

If the buffer text is in character mode, the text shall be inserted into the current line before the cursor, and each line of the buffer other than the first and last shall become a new line in the edit buffer. A *count* shall cause the buffer text to be appended *count* -1 more times to the end of the already added text, each time starting after the last added character.

*Current line:* Unchanged.

*Current column:* If the buffer text is in line mode:

1. If there is a non-<blank> in the first line of the buffer, set to the last column on which any portion of that character is displayed.
2. If there is no non-<blank> in the first line of the buffer, set to the last column on which any portion of the last non-<newline> in the first line of the buffer is displayed.

If the buffer text is in character mode:

1. If the text in the buffer is from more than a single line, then set to the last column on which any portion of the first character from the buffer is displayed.
2. Otherwise, if the buffer is the unnamed buffer, set to the last column on which any portion of the last character from the buffer is displayed.
3. Otherwise, set to the first column on which any portion of the first character from the buffer is displayed.

**Enter ex Mode**

*Synopsis:*     Q

Leave visual or open mode and enter *ex* command mode.

*Current line:* Unchanged.

*Current column:* Unchanged.

**Replace Character**

*Synopsis:*     [*count*] r *character*

Replace the *count* characters at and after the cursor with the specified character. If there are less than *count* non-<newline> characters at and after the cursor on the line, it shall be an error.

If character is <control>-V, any next character other than the <newline> shall be stripped of any special meaning and used as a literal character.

If character is <ESC>, no replacement shall be made and the current line and current column shall be unchanged.

If character is <carriage-return> or <newline>, *count* new lines shall be appended to the current line. All but the last of these lines shall be empty. *count* characters at and after the cursor shall be

discarded, and any remaining characters after the cursor in the current line shall be moved to the last of the new lines. If the **autoindent** edit option is set, they shall be preceded by the same number of **autoindent** characters found on the line from which the command was executed.

*Current line*: Unchanged unless the replacement character is a <carriage-return> or <newline>, in which case it shall be set to line + *count*.

*Current column*: Set to the last column position on which a portion of the last replaced character is displayed, or if the replacement character caused new lines to be created, set to non-<blank>.

### Replace Characters

*Synopsis*:        R

Enter text input mode at the current cursor position possibly replacing text on the current line. A *count* shall cause the input text to be appended *count* - 1 more times to the end of the input.

*Current line/column*: As specified for the text input commands (see [Input Mode Commands in vi](#), on page 3399).

### Substitute Character

*Synopsis*:        [*buffer*][*count*] s

This command shall be equivalent to the *vi* command:

[*buffer*][*count*] c<space>

### Substitute Lines

*Synopsis*:        [*buffer*][*count*] S

This command shall be equivalent to the *vi* command:

[*buffer*][*count*] c\_

### Move Cursor to Before Character (Forward)

*Synopsis*:        [*count*] t *character*

It shall be an error if *count* occurrences of the character do not occur after the cursor in the line.

If used as a motion command:

1. The text region shall be from the cursor up to but not including the *count*th occurrence of the specified character after the cursor.
2. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line*: Unchanged.

*Current column*: Set to the last column in which any portion of the character before the *count*th occurrence of the specified character after the cursor appears in the line.

**Move Cursor to After Character (Reverse)**

*Synopsis:*     [*count*] T *character*

It shall be an error if *count* occurrences of the character do not occur before the cursor in the line.

If used as a motion command:

1. If the character before the cursor is the specified character, it shall be an error.
2. The text region shall be from the character before the cursor up to but not including the *count*th occurrence of the specified character before the cursor.
3. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

*Current line:* Unchanged.

*Current column:* Set to the last column in which any portion of the character after the *count*th occurrence of the specified character before the cursor appears in the line.

**Undo**

*Synopsis:*     u

This command shall be equivalent to the *ex* **undo** command except that the current line and current column shall be set as follows:

*Current line:* Set to the first line added or changed if any; otherwise, move to the line preceding any deleted text if one exists; otherwise, move to line 1.

*Current column:* If undoing an *ex* command, set to the first non-<blank>.

Otherwise, if undoing a text input command:

1. If the command was a **C**, **c**, **O**, **o**, **R**, **S**, or **s** command, the current column shall be set to the value it held when the text input command was entered.
2. Otherwise, set to the last column in which any portion of the first character after the deleted text is displayed, or, if no non-<newline> characters follow the text deleted from this line, set to the last column in which any portion of the last non-<newline> in the line is displayed, or 1 if the line is empty.

Otherwise, if a single line was modified (that is, not added or deleted) by the **u** command:

1. If text was added or changed, set to the last column in which any portion of the first character added or changed is displayed.
2. If text was deleted, set to the last column in which any portion of the first character after the deleted text is displayed, or, if no non-<newline> characters follow the deleted text, set to the last column in which any portion of the last non-<newline> in the line is displayed, or 1 if the line is empty.

Otherwise, set to non-<blank>.

**Undo Current Line**

*Synopsis:*     U

Restore the current line to its state immediately before the most recent time that it became the current line.

*Current line:* Unchanged.

*Current column:* Set to the first column in the line in which any portion of the first character in the line is displayed.

**Move to Beginning of Word**

*Synopsis:*     [count] w

With the exception that words are used as the delimiter instead of bigwords, this command shall be equivalent to the **W** command.

**Move to Beginning of Bigword**

*Synopsis:*     [count] W

If the edit buffer is empty, it shall be an error. If there are less than *count* bigwords between the cursor and the end of the edit buffer, *count* shall be adjusted to move the cursor to the last bigword in the edit buffer.

If used as a motion command:

1. If the associated command is **c**, *count* is 1, and the cursor is on a <blank>, the region of text shall be the current character and no further action shall be taken.
2. If there are less than *count* bigwords between the cursor and the end of the edit buffer, then the command shall succeed, and the region of text shall include the last character of the edit buffer.
3. If there are <blank> characters or an end-of-line that precede the *count*th bigword, and the associated command is **c**, the region of text shall be up to and including the last character before the preceding <blank> characters or end-of-line.
4. If there are <blank> characters or an end-of-line that precede the bigword, and the associated command is **d** or **y**, the region of text shall be up to and including the last <blank> before the start of the bigword or end-of-line.
5. Any text copied to a buffer shall be in character mode.

If not used as a motion command:

1. If the cursor is on the last character of the edit buffer, it shall be an error.

*Current line:* Set to the line containing the current column.

*Current column:* Set to the last column in which any part of the first character of the *count*th next bigword is displayed.



**Delete Character at Cursor**

*Synopsis:*     [*buffer*][*count*] x

Delete the *count* characters at and after the current character into *buffer*, if specified, and into the unnamed buffer.

If the line is empty, it shall be an error. If there are less than *count* non-<newline> characters at and after the cursor on the current line, *count* shall be adjusted to the number of non-<newline> characters at and after the cursor.

*Current line:* Unchanged.

*Current column:* If the line is empty, set to column position 1. Otherwise, if there were *count* or less non-<newline> characters at and after the cursor on the current line, set to the last column that displays any part of the last non-<newline> of the line. Otherwise, unchanged.

**Delete Character Before Cursor**

*Synopsis:*     [*buffer*][*count*] X

Delete the *count* characters before the current character into *buffer*, if specified, and into the unnamed buffer.

If there are no characters before the current character on the current line, it shall be an error. If there are less than *count* previous characters on the current line, *count* shall be adjusted to the number of previous characters on the line.

*Current line:* Unchanged.

*Current column:* Set to (current column – the width of the deleted characters).

**Yank**

*Synopsis:*     [*buffer*][*count*] y *motion*

Copy (yank) the region of text into *buffer*, if specified, and into the unnamed buffer.

If the motion command is the y command repeated:

1. The buffer shall be in line mode.
2. If there are less than *count* – 1 lines after the current line in the edit buffer, it shall be an error.
3. The text region shall be from the current line up to and including the next *count* – 1 lines.

Otherwise, the buffer text mode and text region shall be as specified by the motion command.

*Current line:* If the motion was from the current cursor position toward the end of the edit buffer, unchanged. Otherwise, set to the first line in the edit buffer that is part of the text region specified by the motion command.

*Current column:*

1. If the motion was from the current cursor position toward the end of the edit buffer, unchanged.
2. Otherwise, if the current line is empty, set to column position 1.
3. Otherwise, set to the last column that displays any part of the first character in the file that is part of the text region specified by the motion command.

**Yank Current Line**

*Synopsis:*     `[buffer][count] Y`

This command shall be equivalent to the *vi* command:

`[buffer][count] y_`

**Redraw Window**

If in open mode, the **z** command shall have the Synopsis:

*Synopsis:*     `[count] z`

If *count* is not specified, it shall default to the **window** edit option `-1`. The **z** command shall be equivalent to the *ex z* command, with a type character of `=` and a *count* of *count* `-2`, except that the current line and current column shall be set as follows, and the **window** edit option shall not be affected. If the calculation for the *count* argument would result in a negative number, the *count* argument to the *ex z* command shall be zero. A blank line shall be written after the last line is written.

*Current line:* Unchanged.

*Current column:* Unchanged.

If not in open mode, the **z** command shall have the following Synopsis:

*Synopsis:*     `[line] z [count] character`

If *line* is not specified, it shall default to the current line. If *line* is specified, but is greater than the number of lines in the edit buffer, it shall default to the number of lines in the edit buffer.

If *count* is specified, the value of the **window** edit option shall be set to *count* (as described in the *ex window* command), and the screen shall be redrawn.

*line* shall be placed as specified by the following characters:

<newline>, <carriage-return>

Place the beginning of the line on the first line of the display.

. Place the beginning of the line in the center of the display. The middle line of the display shall be calculated as described for the **M** command.

– Place an unspecified portion of the line on the last line of the display.

+ If *line* was specified, equivalent to the <newline> case. If *line* was not specified, display a screen where the first line of the display shall be (current last line) `+1`. If there are no lines after the last line in the display, it shall be an error.

^ If *line* was specified, display a screen where the last line of the display shall contain an unspecified portion of the first line of a display that had an unspecified portion of the specified line on the last line of the display. If this calculation results in a line before the beginning of the edit buffer, display the first screen of the edit buffer.

Otherwise, display a screen where the last line of the display shall contain an unspecified portion of (current first line `-1`). If this calculation results in a line before the beginning of the edit buffer, it shall be an error.

114513 *Current line*: If *line* and the ' ^ ' character were specified:

114514 1. If the first screen was displayed as a result of the command attempting to display lines  
114515 before the beginning of the edit buffer: if the first screen was already displayed,  
114516 unchanged; otherwise, set to (current first line -1).

114517 2. Otherwise, set to the last line of the display.

114518 If *line* and the ' + ' character were specified, set to the first line of the display.

114519 Otherwise, if *line* was specified, set to *line*.

114520 Otherwise, unchanged.

114521 *Current column*: Set to non-<blank>.

## 114522 **Exit**

114523 *Synopsis*:      ZZ

114524 This command shall be equivalent to the *ex xit* command with no addresses, trailing !, or  
114525 filename (see the *ex xit* command).

## 114526 **Input Mode Commands in vi**

114527 In text input mode, the current line shall consist of zero or more of the following categories, plus  
114528 the terminating <newline>:

114529 1. Characters preceding the text input entry point

114530 Characters in this category shall not be modified during text input mode.

114531 2. **autoindent** characters

114532 **autoindent** characters shall be automatically inserted into each line that is created in text  
114533 input mode, either as a result of entering a <newline> or <carriage-return> while in text  
114534 input mode, or as an effect of the command itself; for example, **O** or **o** (see the *ex*  
114535 **autoindent** command), as if entered by the user.

114536 It shall be possible to erase **autoindent** characters with the <control>-D command; it is  
114537 unspecified whether they can be erased by <control>-H, <control>-U, and <control>-W  
114538 characters. Erasing any **autoindent** character turns the glyph into erase-columns and  
114539 deletes the character from the edit buffer, but does not change its representation on the  
114540 screen.

114541 3. Text input characters

114542 Text input characters are the characters entered by the user. Erasing any text input  
114543 character turns the glyph into erase-columns and deletes the character from the edit  
114544 buffer, but does not change its representation on the screen.

114545 Each text input character entered by the user (that does not have a special meaning) shall  
114546 be treated as follows:

114547 a. The text input character shall be appended to the last character in the edit buffer  
114548 from the first, second, or third categories.

114549 b. If there are no erase-columns on the screen, the text input command was the **R**  
114550 command, and characters in the fifth category from the original line follow the  
114551 cursor, the next such character shall be deleted from the edit buffer. If the  
114552 **slowopen** edit option is not set, the corresponding glyph on the screen shall

- 114553 become erase-columns.
- 114554 c. If there are erase-columns on the screen, as many columns as they occupy, or as are  
114555 necessary, shall be overwritten to display the text input character. (If only part of a  
114556 multi-column glyph is overwritten, the remainder shall be left on the screen, and  
114557 continue to be treated as erase-columns; it is unspecified whether the remainder of  
114558 the glyph is modified in any way.)
- 114559 d. If additional display line columns are needed to display the text input character:
- 114560 i. If the **slowopen** edit option is set, the text input characters shall be  
114561 displayed on subsequent display line columns, overwriting any characters  
114562 displayed in those columns.
- 114563 ii. Otherwise, any characters currently displayed on or after the column on the  
114564 display line where the text input character is to be displayed shall be  
114565 pushed ahead the number of display line columns necessary to display the  
114566 rest of the text input character.
- 114567 4. Erase-columns
- 114568 Erase-columns are not logically part of the edit buffer, appearing only on the screen, and  
114569 may be overwritten on the screen by subsequent text input characters. When text input  
114570 mode ends, all erase-columns shall no longer appear on the screen.
- 114571 Erase-columns are initially the region of text specified by the **c** command (see [Change](#), on  
114572 page 3386); however, erasing **autoindent** or text input characters causes the glyphs of the  
114573 erased characters to be treated as erase-columns.
- 114574 5. Characters following the text region for the **c** command, or the text input entry point for  
114575 all other commands
- 114576 Characters in this category shall not be modified during text input mode, except as  
114577 specified in category 3.b. for the **R** text input command, or as <blank> characters deleted  
114578 when a <newline> or <carriage-return> is entered.
- 114579 It is unspecified whether it is an error to attempt to erase past the beginning of a line that was  
114580 created by the entry of a <newline> or <carriage-return> during text input mode. If it is not an  
114581 error, the editor shall behave as if the erasing character was entered immediately after the last  
114582 text input character entered on the previous line, and all of the non-<newline> characters on the  
114583 current line shall be treated as erase-columns.
- 114584 When text input mode is entered, or after a text input mode character is entered (except as  
114585 specified for the special characters below), the cursor shall be positioned as follows:
- 114586 1. On the first column that displays any part of the first erase-column, if one exists
- 114587 2. Otherwise, if the **slowopen** edit option is set, on the first display line column after the last  
114588 character in the first, second, or third categories, if one exists
- 114589 3. Otherwise, the first column that displays any part of the first character in the fifth  
114590 category, if one exists
- 114591 4. Otherwise, the display line column after the last character in the first, second, or third  
114592 categories, if one exists
- 114593 5. Otherwise, on column position 1
- 114594 The characters that are updated on the screen during text input mode are unspecified, other than  
114595 that the last text input character shall always be updated, and, if the **slowopen** edit option is not

114596 set, the current cursor character shall always be updated.

114597 The following specifications are for command characters entered during text input mode.

114598 **NUL**

114599 *Synopsis:* NUL

114600 If the first character of the text input is a NUL, the most recently input text shall be input as if  
 114601 entered by the user, and then text input mode shall be exited. The text shall be input literally;  
 114602 that is, characters are neither macro or abbreviation expanded, nor are any characters interpreted  
 114603 in any special manner. It is unspecified whether implementations shall support more than 256  
 114604 bytes of remembered input text.

114605 **<control>-D**

114606 *Synopsis:* <control>-D

114607 The <control>-D character shall have no special meaning when in text input mode for a line-  
 114608 oriented command (see [Command Descriptions in vi](#), on page 3365).

114609 This command need not be supported on block-mode terminals.

114610 If the cursor does not follow an **autoindent** character, or an **autoindent** character and a '0' or  
 114611 '^' character:

- 114612 1. If the cursor is in column position 1, the <control>-D character shall be discarded and no  
 114613 further action taken.
- 114614 2. Otherwise, the <control>-D character shall have no special meaning.

114615 If the last input character was a '0', the cursor shall be moved to column position 1.

114616 Otherwise, if the last input character was a '^', the cursor shall be moved to column position 1.  
 114617 In addition, the **autoindent** level for the next input line shall be derived from the same line from  
 114618 which the **autoindent** level for the current input line was derived.

114619 Otherwise, the cursor shall be moved back to the column after the previous shiftwidth (see the  
 114620 *ex* **shiftwidth** command) boundary.

114621 All of the glyphs on columns between the starting cursor position and (inclusively) the ending  
 114622 cursor position shall become erase-columns as described in [Input Mode Commands in vi](#) (on  
 114623 page 3399).

114624 *Current line:* Unchanged.

114625 *Current column:* Set to 1 if the <control>-D was preceded by a '^' or '0'; otherwise, set to  
 114626 (column -1) - ((column -2) % **shiftwidth**).

114627 **<control>-H**

114628 *Synopsis:* <control>-H

114629 If in text input mode for a line-oriented command, and there are no characters to erase, text  
 114630 input mode shall be terminated, no further action shall be done for this command, and the  
 114631 current line and column shall be unchanged.

114632 If there are characters other than **autoindent** characters that have been input on the current line  
 114633 before the cursor, the cursor shall move back one character.

114634 Otherwise, if there are **autoindent** characters on the current line before the cursor, it is

implementation-defined whether the <control>-H command is an error or if the cursor moves back one **autoindent** character.

Otherwise, if the cursor is in column position 1 and there are previous lines that have been input, it is implementation-defined whether the <control>-H command is an error or if it is equivalent to entering <control>-H after the last input character on the previous input line.

Otherwise, it shall be an error.

All of the glyphs on columns between the starting cursor position and (inclusively) the ending cursor position shall become erase-columns as described in [Input Mode Commands in vi](#) (on page 3399).

The current erase character (see *stty*) shall cause an equivalent action to the <control>-H command, unless the previously inserted character was a <backslash>, in which case it shall be as if the literal current erase character had been inserted instead of the <backslash>.

*Current line*: Unchanged, unless previously input lines are erased, in which case it shall be set to line -1.

*Current column*: Set to the first column that displays any portion of the character backed up over.

#### <newline>

*Synopsis*:     <newline>  
                  <carriage-return>  
                  <control>-J  
                  <control>-M

If input was part of a line-oriented command, text input mode shall be terminated and the command shall continue execution with the input provided.

Otherwise, terminate the current line. If there are no characters other than **autoindent** characters on the line, all characters on the line shall be discarded. Otherwise, it is unspecified whether the **autoindent** characters in the line are modified by entering these characters.

Continue text input mode on a new line appended after the current line. If the **slowopen** edit option is set, the lines on the screen below the current line shall not be pushed down, but the first of them shall be cleared and shall appear to be overwritten. Otherwise, the lines of the screen below the current line shall be pushed down.

If the **autoindent** edit option is set, an appropriate number of **autoindent** characters shall be added as a prefix to the line as described by the *ex* **autoindent** edit option.

All columns after the cursor that are erase-columns (as described in [Input Mode Commands in vi](#), on page 3399) shall be discarded.

If the **autoindent** edit option is set, all <blank> characters immediately following the cursor shall be discarded.

All remaining characters after the cursor shall be transferred to the new line, positioned after any **autoindent** characters.

*Current line*: Set to current line +1.

*Current column*: Set to the first column that displays any portion of the first character after the **autoindent** characters on the new line, if any, or the first column position after the last **autoindent** character, if any, or column position 1.

114676 **<control>-T**114677 *Synopsis:*      <control>-T114678 The <control>-T character shall have no special meaning when in text input mode for a line-  
114679 oriented command (see [Command Descriptions in vi](#), on page 3365).

114680 This command need not be supported on block-mode terminals.

114681 Behave as if the user entered the minimum number of <blank> characters necessary to move the  
114682 cursor forward to the column position after the next **shiftwidth** (see the *ex* **shiftwidth**  
114683 command) boundary.114684 *Current line:* Unchanged.114685 *Current column:* Set to  $column + \text{shiftwidth} - ((column - 1) \% \text{shiftwidth})$ .114686 **<control>-U**114687 *Synopsis:*      <control>-U114688 If there are characters other than **autoindent** characters that have been input on the current line  
114689 before the cursor, the cursor shall move to the first character input after the **autoindent**  
114690 characters.114691 Otherwise, if there are **autoindent** characters on the current line before the cursor, it is  
114692 implementation-defined whether the <control>-U command is an error or if the cursor moves to  
114693 the first column position on the line.114694 Otherwise, if the cursor is in column position 1 and there are previous lines that have been  
114695 input, it is implementation-defined whether the <control>-U command is an error or if it is  
114696 equivalent to entering <control>-U after the last input character on the previous input line.

114697 Otherwise, it shall be an error.

114698 All of the glyphs on columns between the starting cursor position and (inclusively) the ending  
114699 cursor position shall become erase-columns as described in [Input Mode Commands in vi](#) (on  
114700 page 3399).114701 The current *kill* character (see *stty*) shall cause an equivalent action to the <control>-U command,  
114702 unless the previously inserted character was a <backslash>, in which case it shall be as if the  
114703 literal current *kill* character had been inserted instead of the <backslash>.114704 *Current line:* Unchanged, unless previously input lines are erased, in which case it shall be set to  
114705 line -1.114706 *Current column:* Set to the first column that displays any portion of the last character backed up  
114707 over.114708 **<control>-V**114709 *Synopsis:*      <control>-V  
114710                      <control>-Q114711 Allow the entry of any subsequent character, other than <control>-J or the <newline>, as a literal  
114712 character, removing any special meaning that it may have to the editor in text input mode. If a  
114713 <control>-V or <control>-Q is entered before a <control>-J or <newline>, the <control>-V or  
114714 <control>-Q character shall be discarded, and the <control>-J or <newline> shall behave as  
114715 described in the <newline> command character during input mode.

114716 For purposes of the display only, the editor shall behave as if a ' ^ ' character was entered, and



the cursor shall be positioned as if overwriting the ' ^ ' character. When a subsequent character is entered, the editor shall behave as if that character was entered instead of the original <control>-V or <control>-Q character.

*Current line:* Unchanged.

*Current column:* Unchanged.

#### <control>-W

*Synopsis:*      <control>-W

If there are characters other than **autoindent** characters that have been input on the current line before the cursor, the cursor shall move back over the last word preceding the cursor (including any <blank> characters between the end of the last word and the current cursor); the cursor shall not move to before the first character after the end of any **autoindent** characters.

Otherwise, if there are **autoindent** characters on the current line before the cursor, it is implementation-defined whether the <control>-W command is an error or if the cursor moves to the first column position on the line.

Otherwise, if the cursor is in column position 1 and there are previous lines that have been input, it is implementation-defined whether the <control>-W command is an error or if it is equivalent to entering <control>-W after the last input character on the previous input line.

Otherwise, it shall be an error.

All of the glyphs on columns between the starting cursor position and (inclusively) the ending cursor position shall become erase-columns as described in [Input Mode Commands in vi](#) (on page 3399).

*Current line:* Unchanged, unless previously input lines are erased, in which case it shall be set to line -1.

*Current column:* Set to the first column that displays any portion of the last character backed up over.

#### <ESC>

*Synopsis:*      <ESC>

If input was part of a line-oriented command:

1. If *interrupt* was entered, text input mode shall be terminated and the editor shall return to command mode. The terminal shall be alerted.
2. If <ESC> was entered, text input mode shall be terminated and the command shall continue execution with the input provided.

Otherwise, terminate text input mode and return to command mode.

Any **autoindent** characters entered on newly created lines that have no other non-<newline> characters shall be deleted.

Any leading **autoindent** and <blank> characters on newly created lines shall be rewritten to be the minimum number of <blank> characters possible.

The screen shall be redisplayed as necessary to match the contents of the edit buffer.

*Current line:* Unchanged.



114756 *Current column:*

- 114757 1. If there are text input characters on the current line, the column shall be set to the last
- 114758 column where any portion of the last text input character is displayed.
- 114759 2. Otherwise, if a character is displayed in the current column, unchanged.
- 114760 3. Otherwise, set to column position 1.

#### 114761 **EXIT STATUS**

114762 The following exit values shall be returned:

- 114763 0 Successful completion.
- 114764 >0 An error occurred.

#### 114765 **CONSEQUENCES OF ERRORS**

114766 When any error is encountered and the standard input is not a terminal device file, *vi* shall not

114767 write the file or return to command or text input mode, and shall terminate with a non-zero exit

114768 status.

114769 Otherwise, when an unrecoverable error is encountered it shall be equivalent to a SIGHUP

114770 asynchronous event.

114771 Otherwise, when an error is encountered, the editor shall behave as specified in [Command](#)

114772 [Descriptions in vi](#) (on page 3365).

#### 114773 **APPLICATION USAGE**

114774 None.

#### 114775 **EXAMPLES**

114776 None.

#### 114777 **RATIONALE**

114778 See the RATIONALE for [ex](#) for more information on *vi*. Major portions of the *vi* utility

114779 specification point to *ex* to avoid inadvertent divergence. While *ex* and *vi* have historically been

114780 implemented as a single utility, this is not required by POSIX.1-2008.

114781 It is recognized that portions of *vi* would be difficult, if not impossible, to implement

114782 satisfactorily on a block-mode terminal, or a terminal without any form of cursor addressing,

114783 thus it is not a mandatory requirement that such features should work on all terminals. It is the

114784 intention, however, that a *vi* implementation should provide the full set of capabilities on all

114785 terminals capable of supporting them.

114786 Historically, *vi* exited immediately if the standard input was not a terminal. POSIX.1-2008

114787 permits, but does not require, this behavior. An end-of-file condition is not equivalent to an end-

114788 of-file character. A common end-of-file character, <control>-D, is historically a *vi* command.

114789 The text in the STDOUT section reflects the usage of the verb *display* in this section; some

114790 implementations of *vi* use standard output to write to the terminal, but POSIX.1-2008 does not

114791 require that to be the case.

114792 Historically, implementations reverted to open mode if the terminal was incapable of supporting

114793 full visual mode. POSIX.1-2008 requires this behavior. Historically, the open mode of *vi* behaved

114794 roughly equivalently to the visual mode, with the exception that only a single line from the edit

114795 buffer (one “buffer line”) was kept current at any time. This line was normally displayed on the

114796 next-to-last line of a terminal with cursor addressing (and the last line performed its normal

114797 visual functions for line-oriented commands and messages). In addition, some few commands

114798 behaved differently in open mode than in visual mode. POSIX.1-2008 requires conformance to

114799 historical practice.

Historically, *ex* and *vi* implementations have expected text to proceed in the usual European/Latin order of left to right, top to bottom. There is no requirement in POSIX.1-2008 that this be the case. The specification was deliberately written using words like “before”, “after”, “first”, and “last” in order to permit implementations to support the natural text order of the language.

Historically, lines past the end of the edit buffer were marked with single <tilde> ('~') characters; that is, if the one-based display was 20 lines in length, and the last line of the file was on line one, then lines 2-20 would contain only a single '~' character.

Historically, the *vi* editor attempted to display only complete lines at the bottom of the screen (it did display partial lines at the top of the screen). If a line was too long to fit in its entirety at the bottom of the screen, the screen lines where the line would have been displayed were displayed as single '@' characters, instead of displaying part of the line. POSIX.1-2008 permits, but does not require, this behavior. Implementations are encouraged to attempt always to display a complete line at the bottom of the screen when doing scrolling or screen positioning by buffer lines.

Historically, lines marked with '@' were also used to minimize output to dumb terminals over slow lines; that is, changes local to the cursor were updated, but changes to lines on the screen that were not close to the cursor were simply marked with an '@' sign instead of being updated to match the current text. POSIX.1-2008 permits, but does not require this feature because it is used ever less frequently as terminals become smarter and connections are faster.

### Initialization in *ex* and *vi*

Historically, *vi* always had a line in the edit buffer, even if the edit buffer was “empty”. For example:

1. The *ex* command = executed from visual mode wrote “1” when the buffer was empty.
2. Writes from visual mode of an empty edit buffer wrote files of a single character (a <newline>), while writes from *ex* mode of an empty edit buffer wrote empty files.
3. Put and read commands into an empty edit buffer left an empty line at the top of the edit buffer.

For consistency, POSIX.1-2008 does not permit any of these behaviors.

Historically, *vi* did not always return the terminal to its original modes; for example, ICRNL was modified if it was not originally set. POSIX.1-2008 does not permit this behavior.

### Command Descriptions in *vi*

Motion commands are among the most complicated aspects of *vi* to describe. With some exceptions, the text region and buffer type effect of a motion command on a *vi* command are described on a case-by-case basis. The descriptions of text regions in POSIX.1-2008 are not intended to imply direction; that is, an inclusive region from line *n* to line *n*+5 is identical to a region from line *n*+5 to line *n*. This is of more than academic interest—movements to marks can be in either direction, and, if the **wraps** option is set, so can movements to search points. Historically, lines are always stored into buffers in text order; that is, from the start of the edit buffer to the end. POSIX.1-2008 requires conformance to historical practice.

Historically, command counts were applied to any associated motion, and were multiplicative to any supplied motion count. For example, **2cw** is the same as **c2w**, and **2c3w** is the same as **c6w**. POSIX.1-2008 requires this behavior. Historically, *vi* commands that used bigwords, words, paragraphs, and sentences as objects treated groups of empty lines, or lines that contained only

<blank> characters, inconsistently. Some commands treated them as a single entity, while others treated each line separately. For example, the **w**, **W**, and **B** commands treated groups of empty lines as individual words; that is, the command would move the cursor to each new empty line. The **e** and **E** commands treated groups of empty lines as a single word; that is, the first use would move past the group of lines. The **b** command would just beep at the user, or if done from the start of the line as a motion command, fail in unexpected ways. If the lines contained only (or ended with) <blank> characters, the **w** and **W** commands would just beep at the user, the **E** and **e** commands would treat the group as a single word, and the **B** and **b** commands would treat the lines as individual words. For consistency and simplicity of specification, POSIX.1-2008 requires that all *vi* commands treat groups of empty or blank lines as a single entity, and that movement through lines ending with <blank> characters be consistent with other movements.

Historically, *vi* documentation indicated that any number of double-quotes were skipped after punctuation marks at sentence boundaries; however, implementations only skipped single-quotes. POSIX.1-2008 requires both to be skipped.

Historically, the first and last characters in the edit buffer were word boundaries. This historical practice is required by POSIX.1-2008.

Historically, *vi* attempted to update the minimum number of columns on the screen possible, which could lead to misleading information being displayed. POSIX.1-2008 makes no requirements other than that the current character being entered is displayed correctly, leaving all other decisions in this area up to the implementation.

Historically, lines were arbitrarily folded between columns of any characters that required multiple column positions on the screen, with the exception of tabs, which terminated at the right-hand margin. POSIX.1-2008 permits the former and requires the latter. Implementations that do not arbitrarily break lines between columns of characters that occupy multiple column positions should not permit the cursor to rest on a column that does not contain any part of a character.

The historical *vi* had a problem in that all movements were by buffer lines, not by display or screen lines. This is often the right thing to do; for example, single line movements, such as **j** or **k**, should work on buffer lines. Commands like **dj**, or **j.**, where **.** is a change command, only make sense for buffer lines. It is not, however, the right thing to do for screen motion or scrolling commands like <control>-D, <control>-F, and **H**. If the window is fairly small, using buffer lines in these cases can result in completely random motion; for example, **1<control>-D** can result in a completely changed screen, without any overlap. This is clearly not what the user wanted. The problem is even worse in the case of the **H**, **L**, and **M** commands—as they position the cursor at the first non-<blank> of the line, they may all refer to the same location in large lines, and will result in no movement at all.

In addition, if the line is larger than the screen, using buffer lines can make it impossible to display parts of the line—there are not any commands that do not display the beginning of the line in historical *vi*, and if both the beginning and end of the line cannot be on the screen at the same time, the user suffers. Finally, the page and half-page scrolling commands historically moved to the first non-<blank> in the new line. If the line is approximately the same size as the screen, this is inadequate because the cursor before and after a <control>-D command will refer to the same location on the screen.

Implementations of *ex* and *vi* exist that do not have these problems because the relevant commands (<control>-B, <control>-D, <control>-F, <control>-U, <control>-Y, <control>-E, **H**, **L**, and **M**) operate on display (screen) lines, not (edit) buffer lines.

POSIX.1-2008 does not permit this behavior by default because the standard developers believed that users would find it too confusing. However, historical practice has been relaxed. For

example, *ex* and *vi* historically attempted, albeit sometimes unsuccessfully, to never put part of a line on the last lines of a screen; for example, if a line would not fit in its entirety, no part of the line was displayed, and the screen lines corresponding to the line contained single '@' characters. This behavior is permitted, but not required by POSIX.1-2008, so that it is possible for implementations to support long lines in small screens more reasonably without changing the commands to be oriented to the display (instead of oriented to the buffer). POSIX.1-2008 also permits implementations to refuse to edit any edit buffer containing a line that will not fit on the screen in its entirety.

The display area (for example, the value of the **window** edit option) has historically been “grown”, or expanded, to display new text when local movements are done in displays where the number of lines displayed is less than the maximum possible. Expansion has historically been the first choice, when the target line is less than the maximum possible expansion value away. Scrolling has historically been the next choice, done when the target line is less than half a display away, and otherwise, the screen was redrawn. There were exceptions, however, in that *ex* commands generally always caused the screen to be redrawn. POSIX.1-2008 does not specify a standard behavior because there may be external issues, such as connection speed, the number of characters necessary to redraw as opposed to scroll, or terminal capabilities that implementations will have to accommodate.

The current line in POSIX.1-2008 maps one-to-one to a buffer line in the file. The current column does not. There are two different column values that are described by POSIX.1-2008. The first is the current column value as set by many of the *vi* commands. This value is remembered for the lifetime of the editor. The second column value is the actual position on the screen where the cursor rests. The two are not always the same. For example, when the cursor is backed by a multi-column character, the actual cursor position on the screen has historically been the last column of the character in command mode, and the first column of the character in input mode.

Commands that set the current line, but that do not set the current cursor value (for example, **j** and **k**) attempt to get as close as possible to the remembered column position, so that the cursor tends to restrict itself to a vertical column as the user moves around in the edit buffer. POSIX.1-2008 requires conformance to historical practice, requiring that the display location of the cursor on the display line be adjusted from the current column value as necessary to support this historical behavior.

Historically, only a single line (and for some terminals, a single line minus 1 column) of characters could be entered by the user for the line-oriented commands; that is, **:**, **!**, **/**, or **?**. POSIX.1-2008 permits, but does not require, this limitation.

Historically, “soft” errors in *vi* caused the terminal to be alerted, but no error message was displayed. As a general rule, no error message was displayed for errors in command execution in *vi*, when the error resulted from the user attempting an invalid or impossible action, or when a searched-for object was not found. Examples of soft errors included **h** at the left margin, **<control>-B** or **[** at the beginning of the file, **2G** at the end of the file, and so on. In addition, errors such as **%**, **]**, **)**, **N**, **n**, **f**, **F**, **t**, and **T** failing to find the searched-for object were soft as well. Less consistently, **/** and **?** displayed an error message if the pattern was not found, **/**, **?**, **N**, and **n** displayed an error message if no previous regular expression had been specified, and **;** did not display an error message if no previous **f**, **F**, **t**, or **T** command had occurred. Also, behavior in this area might reasonably be based on a runtime evaluation of the speed of a network connection. Finally, some implementations have provided error messages for soft errors in order to assist naive users, based on the value of a verbose edit option. POSIX.1-2008 does not list specific errors for which an error message shall be displayed. Implementations should conform to historical practice in the absence of any strong reason to diverge.

## Page Backwards

The <control>-B and <control>-F commands historically considered it an error to attempt to page past the beginning or end of the file, whereas the <control>-D and <control>-U commands simply moved to the beginning or end of the file. For consistency, POSIX.1-2008 requires the latter behavior for all four commands. All four commands still consider it an error if the current line is at the beginning (<control>-B, <control>-U) or end (<control>-F, <control>-D) of the file. Historically, the <control>-B and <control>-F commands skip two lines in order to include overlapping lines when a single command is entered. This makes less sense in the presence of a *count*, as there will be, by definition, no overlapping lines. The actual calculation used by historical implementations of the *vi* editor for <control>-B was:

$$((\text{current first line}) - \text{count} \times (\text{window edit option})) + 2$$

and for <control>-F was:

$$((\text{current first line}) + \text{count} \times (\text{window edit option})) - 2$$

This calculation does not work well when intermixing commands with and without counts; for example, **3<control>-F is not equivalent to entering the <control>-F command three times, and is not reversible by entering the <control>-B command three times. For consistency with other *vi* commands that take counts, POSIX.1-2008 requires a different calculation.**

## Scroll Forward

The 4BSD and System V implementations of *vi* differed on the initial value used by the **scroll** command. 4BSD used:

$$((\text{window edit option}) + 1) / 2$$

while System V used the value of the **scroll** edit option. The System V version is specified by POSIX.1-2008 because the standard developers believed that it was more intuitive and permitted the user a method of setting the scroll value initially without also setting the number of lines that are displayed.

## Scroll Forward by Line

Historically, the <control>-E and <control>-Y commands considered it an error if the last and first lines, respectively, were already on the screen. POSIX.1-2008 requires conformance to historical practice. Historically, the <control>-E and <control>-Y commands had no effect in open mode. For simplicity and consistency of specification, POSIX.1-2008 requires that they behave as usual, albeit with a single line screen.

## Clear and Redisplay

The historical <control>-L command refreshed the screen exactly as it was supposed to be currently displayed, replacing any '@' characters for lines that had been deleted but not updated on the screen with refreshed '@' characters. The intent of the <control>-L command is to refresh when the screen has been accidentally overwritten; for example, by a **write** command from another user, or modem noise.



## Redraw Screen

The historical <control>-R command redisplayed only when necessary to update lines that had been deleted but not updated on the screen and that were flagged with '@' characters. There is no requirement that the screen be in any way refreshed if no lines of this form are currently displayed. POSIX.1-2008 permits implementations to extend this command to refresh lines on the screen flagged with '@' characters because they are too long to be displayed in the current framework; however, the current line and column need not be modified.

## Search for tagstring

Historically, the first non-<blank> at or after the cursor was the first character, and all subsequent characters that were word characters, up to the end of the line, were included. For example, with the cursor on the leading <space> or on the '#' character in the text "#bar@", the tag was "#bar". On the character 'b' it was "bar", and on the 'a' it was "ar". POSIX.1-2008 requires this behavior.

## Replace Text with Results from Shell Command

Historically, the <, >, and ! commands considered most cursor motions other than line-oriented motions an error; for example, the command >/foo<CR> succeeded, while the command >I failed, even though the text region described by the two commands might be identical. For consistency, all three commands only consider entire lines and not partial lines, and the region is defined as any line that contains a character that was specified by the motion.

## Move to Matching Character

Other matching characters have been left implementation-defined in order to allow extensions such as matching '<' and '>' for searching HTML, or #ifdef, #else, and #endif for searching C source.

## Repeat Substitution

POSIX.1-2008 requires that any c and g flags specified to the previous substitute command be ignored; however, the r flag may still apply, if supported by the implementation.

## Return to Previous (Context or Section)

The [I, ]I, (, ), {, and } commands are all affected by "section boundaries", but in some historical implementations not all of the commands recognize the same section boundaries. This is a bug, not a feature, and a unique section-boundary algorithm was not described for each command. One special case that is preserved is that the sentence command moves to the end of the last line of the edit buffer while the other commands go to the beginning, in order to preserve the traditional character cut semantics of the sentence command. Historically, vi section boundaries at the beginning and end of the edit buffer were the first non-<blank> on the first and last lines of the edit buffer if one exists; otherwise, the last character of the first and last lines of the edit buffer if one exists. To increase consistency with other section locations, this has been simplified by POSIX.1-2008 to the first character of the first and last lines of the edit buffer, or the first and the last lines of the edit buffer if they are empty.

Sentence boundaries were problematic in the historical vi. They were not only the boundaries as defined for the section and paragraph commands, but they were the first non-<blank> that occurred after those boundaries, as well. Historically, the vi section commands were documented as taking an optional window size as a count preceding the command. This was not implemented in historical versions, so POSIX.1-2008 requires that the count repeat the command,

115020 for consistency with other *vi* commands.

### 115021 Repeat

115022 Historically, mapped commands other than text input commands could not be repeated using  
115023 the **period** command. POSIX.1-2008 requires conformance to historical practice.

115024 The restrictions on the interpretation of special characters (for example, <control>-H) in the  
115025 repetition of text input mode commands is intended to match historical practice. For example,  
115026 given the input sequence:

```
115027 iab<control>-H<control>-H<control>-Hdef<escape>
```

115028 the user should be informed of an error when the sequence is first entered, but not during a  
115029 command repetition. The character <control>-T is specifically exempted from this restriction.  
115030 Historical implementations of *vi* ignored <control>-T characters that were input in the original  
115031 command during command repetition. POSIX.1-2008 prohibits this behavior.

### 115032 Find Regular Expression

115033 Historically, commands did not affect the line searched to or from if the motion command was a  
115034 search (*/*, *?*, *N*, *n*) and the final position was the start/end of the line. There were some special  
115035 cases and *vi* was not consistent. POSIX.1-2008 does not permit this behavior, for consistency.  
115036 Historical implementations permitted but were unable to handle searches as motion commands  
115037 that wrapped (that is, due to the edit option **wrapscreen**) to the original location. POSIX.1-2008  
115038 requires that this behavior be treated as an error.

115039 Historically, the syntax */RE/0* was used to force the command to cut text in line mode.  
115040 POSIX.1-2008 requires conformance to historical practice.

115041 Historically, in open mode, a **z** specified to a search command redisplayed the current line  
115042 instead of displaying the current screen with the current line highlighted. For consistency and  
115043 simplicity of specification, POSIX.1-2008 does not permit this behavior.

115044 Historically, trailing **z** commands were permitted and ignored if entered as part of a search used  
115045 as a motion command. For consistency and simplicity of specification, POSIX.1-2008 does not  
115046 permit this behavior.

### 115047 Execute an *ex* Command

115048 Historically, *vi* implementations restricted the commands that could be entered on the colon  
115049 command line (for example, **append** and **change**), and some other commands were known to  
115050 cause them to fail catastrophically. For consistency, POSIX.1-2008 does not permit these  
115051 restrictions. When executing an *ex* command by entering *:*, it is not possible to enter a <newline>  
115052 as part of the command because it is considered the end of the command. A different approach  
115053 is to enter *ex* command mode by using the *vi* **Q** command (and later resuming visual mode with  
115054 the *ex* **vi** command). In *ex* command mode, the single-line limitation does not exist. So, for  
115055 example, the following is valid:

```
115056 Q
115057 s/break here/break\
115058 here/
115059 vi
```

115060 POSIX.1-2008 requires that, if the *ex* command overwrites any part of the screen that would be  
115061 erased by a refresh, *vi* pauses for a character from the user. Historically, this character could be  
115062 any character; for example, a character input by the user before the message appeared, or even a

mapped character. This is probably a bug, but implementations that have tried to be more rigorous by requiring that the user enter a specific character, or that the user enter a character after the message was displayed, have been forced by user indignation back into historical behavior. POSIX.1-2008 requires conformance to historical practice.

### Shift Left (Right)

Refer to the Rationale for the **!** and **/** commands. Historically, the **<** and **>** commands sometimes moved the cursor to the first non-**<blank>** (for example if the command was repeated or with **\_** as the motion command), and sometimes left it unchanged. POSIX.1-2008 does not permit this inconsistency, requiring instead that the cursor always move to the first non-**<blank>**. Historically, the **<** and **>** commands did not support buffer arguments, although some implementations allow the specification of an optional buffer. This behavior is neither required nor disallowed by POSIX.1-2008.

### Execute

Historically, buffers could execute other buffers, and loops, infinite and otherwise, were possible. POSIX.1-2008 requires conformance to historical practice. The *\*buffer* syntax of *ex* is not required in *vi*, because it is not historical practice and has been used in some *vi* implementations to support additional scripting languages.

### Reverse Case

Historically, the **~** command ignored any associated *count*, and acted only on the characters in the current line. For consistency with other *vi* commands, POSIX.1-2008 requires that an associated *count* act on the next *count* characters, and that the command move to subsequent lines if warranted by *count*, to make it possible to modify large pieces of text in a reasonably efficient manner. There exist *vi* implementations that optionally require an associated motion command for the **~** command. Implementations supporting this functionality are encouraged to base it on the **tildedop** edit option and handle the text regions and cursor positioning identically to the **yank** command.

### Append

Historically, *counts* specified to the **A**, **a**, **I**, and **i** commands repeated the input of the first line *count* times, and did not repeat the subsequent lines of the input text. POSIX.1-2008 requires that the entire text input be repeated *count* times.

### Move Backward to Preceding Word

Historically, *vi* became confused if word commands were used as motion commands in empty files. POSIX.1-2008 requires that this be an error. Historical implementations of *vi* had a large number of bugs in the word movement commands, and they varied greatly in behavior in the presence of empty lines, “words” made up of a single character, and lines containing only **<blank>** characters. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.



### Change to End-of-Line

Some historical implementations of the **C** command did not behave as described by POSIX.1-2008 when the **\$** key was remapped because they were implemented by pushing the **\$** key onto the input queue and reprocessing it. POSIX.1-2008 does not permit this behavior. Historically, the **C**, **S**, and **s** commands did not copy replaced text into the numeric buffers. For consistency and simplicity of specification, POSIX.1-2008 requires that they behave like their respective **c** commands in all respects.

### Delete

Historically, lines in open mode that were deleted were scrolled up, and an **@** glyph written over the beginning of the line. In the case of terminals that are incapable of the necessary cursor motions, the editor erased the deleted line from the screen. POSIX.1-2008 requires conformance to historical practice; that is, if the terminal cannot display the **'@'** character, the line cannot remain on the screen.

### Delete to End-of-Line

Some historical implementations of the **D** command did not behave as described by POSIX.1-2008 when the **\$** key was remapped because they were implemented by pushing the **\$** key onto the input queue and reprocessing it. POSIX.1-2008 does not permit this behavior.

### Join

An historical oddity of *vi* is that the commands **J**, **1J**, and **2J** are all equivalent. POSIX.1-2008 requires conformance to historical practice. The *vi* **J** command is specified in terms of the *ex* **join** command with an *ex* command *count* value. The address correction for a *count* that is past the end of the edit buffer is necessary for historical compatibility for both *ex* and *vi*.

### Mark Position

Historical practice is that only lowercase letters, plus backquote and single-quote, could be used to mark a cursor position. POSIX.1-2008 requires conformance to historical practice, but encourages implementations to support other characters as marks as well.

### Repeat Regular Expression Find (Forward and Reverse)

Historically, the **N** and **n** commands could not be used as motion components for the **c** command. With the exception of the **cN** command, which worked if the search crossed a line boundary, the text region would be discarded, and the user would not be in text input mode. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

### Insert Empty Line (Below and Above)

Historically, counts to the **O** and **o** commands were used as the number of physical lines to open, if the terminal was dumb and the **slowopen** option was not set. This was intended to minimize traffic over slow connections and repainting for dumb terminals. POSIX.1-2008 does not permit this behavior, requiring that a *count* to the open command behave as for other text input commands. This change to historical practice was made for consistency, and because a superset of the functionality is provided by the **slowopen** edit option.

### Put from Buffer (Following and Before)

Historically, *counts* to the **p** and **P** commands were ignored if the buffer was a line mode buffer, but were (mostly) implemented as described in POSIX.1-2008 if the buffer was a character mode buffer. Because implementations exist that do not have this limitation, and because pasting lines multiple times is generally useful, POSIX.1-2008 requires that *count* be supported for all **p** and **P** commands.

Historical implementations of *vi* were widely known to have major problems in the **p** and **P** commands, particularly when unusual regions of text were copied into the edit buffer. The standard developers viewed these as bugs, and they are not permitted for consistency and simplicity of specification.

Historically, a **P** or **p** command (or an *ex* **put** command executed from open or visual mode) executed in an empty file, left an empty line as the first line of the file. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

### Replace Character

Historically, the **r** command did not correctly handle the *erase* and *word erase* characters as arguments, nor did it handle an associated *count* greater than 1 with a <carriage-return> argument, for which it replaced *count* characters with a single <newline>. POSIX.1-2008 does not permit these inconsistencies.

Historically, the **r** command permitted the <control>-V escaping of entered characters, such as <ESC> and the <carriage-return>; however, it required two leading <control>-V characters instead of one. POSIX.1-2008 requires that this be changed for consistency with the other text input commands of *vi*.

Historically, it is an error to enter the **r** command if there are less than *count* characters at or after the cursor in the line. While a reasonable and unambiguous extension would be to permit the **r** command on empty lines, it would require that too large a *count* be adjusted to match the number of characters at or after the cursor for consistency, which is sufficiently different from historical practice to be avoided. POSIX.1-2008 requires conformance to historical practice.

### Replace Characters

Historically, if there were **autoindent** characters in the line on which the **R** command was run, and **autoindent** was set, the first <newline> would be properly indented and no characters would be replaced by the <newline>. Each additional <newline> would replace *n* characters, where *n* was the number of characters that were needed to indent the rest of the line to the proper indentation level. This behavior is a bug and is not permitted by POSIX.1-2008.

### Undo

Historical practice for cursor positioning after undoing commands was mixed. In most cases, when undoing commands that affected a single line, the cursor was moved to the start of added or changed text, or immediately after deleted text. However, if the user had moved from the line being changed, the column was either set to the first non-<blank>, returned to the origin of the command, or remained unchanged. When undoing commands that affected multiple lines or entire lines, the cursor was moved to the first character in the first line restored. As an example of how inconsistent this was, a search, followed by an **o** text input command, followed by an **undo** would return the cursor to the location where the **o** command was entered, but a **cw** command followed by an **o** command followed by an **undo** would return the cursor to the first non-<blank> of the line. POSIX.1-2008 requires the most useful of these behaviors, and discards the least useful, in the interest of consistency and simplicity of specification.

## Yank

Historically, the **yank** command did not move to the end of the motion if the motion was in the forward direction. It moved to the end of the motion if the motion was in the backward direction, except for the **\_** command, or for the **G** and **'** commands when the end of the motion was on the current line. This was further complicated by the fact that for a number of motion commands, the **yank** command moved the cursor but did not update the screen; for example, a subsequent command would move the cursor from the end of the motion, even though the cursor on the screen had not reflected the cursor movement for the **yank** command. POSIX.1-2008 requires that all **yank** commands associated with backward motions move the cursor to the end of the motion for consistency, and specifically, to make **'** commands as motions consistent with search patterns as motions.

## Yank Current Line

Some historical implementations of the **Y** command did not behave as described by POSIX.1-2008 when the **'\_'** key was remapped because they were implemented by pushing the **'\_'** key onto the input queue and reprocessing it. POSIX.1-2008 does not permit this behavior.

## Redraw Window

Historically, the **z** command always redrew the screen. This is permitted but not required by POSIX.1-2008, because of the frequent use of the **z** command in macros such as **map n nz.** for screen positioning, instead of its use to change the screen size. The standard developers believed that expanding or scrolling the screen offered a better interface for users. The ability to redraw the screen is preserved if the optional new window size is specified, and in the **<control>-L** and **<control>-R** commands.

The semantics of **z^** are confusing at best. Historical practice is that the screen before the screen that ended with the specified line is displayed. POSIX.1-2008 requires conformance to historical practice.

Historically, the **z** command would not display a partial line at the top or bottom of the screen. If the partial line would normally have been displayed at the bottom of the screen, the command worked, but the partial line was replaced with **'@'** characters. If the partial line would normally have been displayed at the top of the screen, the command would fail. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

Historically, the **z** command with a line specification of 1 ignored the command. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

Historically, the **z** command did not set the cursor column to the first non-**<blank>** for the character if the first screen was to be displayed, and was already displayed. For consistency and simplicity of specification, POSIX.1-2008 does not permit this behavior.

## Input Mode Commands in vi

Historical implementations of **vi** did not permit the user to erase more than a single line of input, or to use normal erase characters such as *line erase*, *worderase*, and *erase* to erase **autoindent** characters. As there exist implementations of **vi** that do not have these limitations, both behaviors are permitted, but only historical practice is required. In the case of these extensions, **vi** is required to pause at the **autoindent** and previous line boundaries.

Historical implementations of **vi** updated only the portion of the screen where the current cursor character was displayed. For example, consider the **vi** input keystrokes:

```
iabcd<escape>0C<tab>
```

Historically, the <tab> would overwrite the characters "abcd" when it was displayed. Other implementations replace only the 'a' character with the <tab>, and then push the rest of the characters ahead of the cursor. Both implementations have problems. The historical implementation is probably visually nicer for the above example; however, for the keystrokes:

```
iabcd<ESC>OR<tab><ESC>
```

the historical implementation results in the string "bcd" disappearing and then magically reappearing when the <ESC> character is entered. POSIX.1-2008 requires the former behavior when overwriting erase-columns—that is, overwriting characters that are no longer logically part of the edit buffer—and the latter behavior otherwise.

Historical implementations of *vi* discarded the <control>-D and <control>-T characters when they were entered at places where their command functionality was not appropriate. POSIX.1-2008 requires that the <control>-T functionality always be available, and that <control>-D be treated as any other key when not operating on **autoindent** characters.

## NUL

Some historical implementations of *vi* limited the number of characters entered using the NUL input character to 256 bytes. POSIX.1-2008 permits this limitation; however, implementations are encouraged to remove this limit.

## <control>-D

See also Rationale for the input mode command <newline>. The hidden assumptions in the <control>-D command (and in the *vi* **autoindent** specification in general) is that <space> characters take up a single column on the screen and that <tab> characters are comprised of an integral number of <space> characters.

## <newline>

Implementations are permitted to rewrite **autoindent** characters in the line when <newline>, <carriage-return>, <control>-D, and <control>-T are entered, or when the **shift** commands are used, because historical implementations have both done so and found it necessary to do so. For example, a <control>-D when the cursor is preceded by a single <tab>, with **tabstop** set to 8, and **shiftwidth** set to 3, will result in the <tab> being replaced by several <space> characters.

## <control>-T

See also the Rationale for the input mode command <newline>. Historically, <control>-T only worked if no non-<blank> characters had yet been input in the current input line. In addition, the characters inserted by <control>-T were treated as **autoindent** characters, and could not be erased using normal user erase characters. Because implementations exist that do not have these limitations, and as moving to a column boundary is generally useful, POSIX.1-2008 requires that both limitations be removed.

**<control>-V**

Historically, *vi* used **^V**, regardless of the value of the literal-next character of the terminal. POSIX.1-2008 requires conformance to historical practice.

The uses described for <control>-V can also be accomplished with <control>-Q, which is useful on terminals that use <control>-V for the down-arrow function. However, most historical implementations use <control>-Q for the *termios* START character, so the editor will generally not receive the <control>-Q unless **stty ixon** mode is set to off. (In addition, some historical implementations of *vi* explicitly set **ixon** mode to on, so it was difficult for the user to set it to off.) Any of the command characters described in POSIX.1-2008 can be made ineffective by their selection as *termios* control characters, using the *stty* utility or other methods described in the System Interfaces volume of POSIX.1-2008.

**<ESC>**

Historically, SIGINT alerted the terminal when used to end input mode. This behavior is permitted, but not required, by POSIX.1-2008.

**FUTURE DIRECTIONS**

None.

**SEE ALSO**

*ed*, *ex*, *stty*

XBD [Section 12.2](#) (on page 216)

**CHANGE HISTORY**

First released in Issue 2.

**Issue 5**

The FUTURE DIRECTIONS section is added.

**Issue 6**

This utility is marked as part of the User Portability Utilities option.

The APPLICATION USAGE section is added.

The obsolescent SYNOPSIS is removed.

The following new requirements on POSIX implementations derive from alignment with the Single UNIX Specification:

- The **reindent** command description is added.

The *vi* utility has been extensively rewritten for alignment with the IEEE P1003.2b draft standard.

IEEE PASC Interpretations 1003.2 #57, #62, #63, #64, #78, and #188 are applied.

IEEE PASC Interpretation 1003.2 #207 is applied, clarifying the description of the **R** command in a manner similar to the descriptions of other text input mode commands such as **i**, **o**, and **O**.

The **-l** option is removed.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/41 is applied, adding *[count]* to the Synopsis for **[[**.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/42 is applied, adding *[count]* to the Synopsis for **]]**.

115302 **Issue 7**

115303 Austin Group Interpretation 1003.1-2001 #027 is applied, clarifying that '+' may be recognized  
115304 as an option delimiter in the OPTIONS section.

115305 Austin Group Interpretation 1003.1-2001 #087 is applied, updating the Put from Buffer Before (P)  
115306 command description to address multi-line requirements.

115307 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

115308 **NAME**

115309       wait — await process completion

115310 **SYNOPSIS**115311       wait [*pid*...]115312 **DESCRIPTION**

115313       When an asynchronous list (see [Section 2.9.3.1](#), on page 2360) is started by the shell, the process  
 115314       ID of the last command in each element of the asynchronous list shall become known in the  
 115315       current shell execution environment; see [Section 2.12](#) (on page 2371).

115316       If the *wait* utility is invoked with no operands, it shall wait until all process IDs known to the  
 115317       invoking shell have terminated and exit with a zero exit status.

115318       If one or more *pid* operands are specified that represent known process IDs, the *wait* utility shall  
 115319       wait until all of them have terminated. If one or more *pid* operands are specified that represent  
 115320       unknown process IDs, *wait* shall treat them as if they were known process IDs that exited with  
 115321       exit status 127. The exit status returned by the *wait* utility shall be the exit status of the process  
 115322       requested by the last *pid* operand.

115323       The known process IDs are applicable only for invocations of *wait* in the current shell execution  
 115324       environment.

115325 **OPTIONS**

115326       None.

115327 **OPERANDS**

115328       The following operand shall be supported:

115329       *pid*           One of the following:

- 115330           1. The unsigned decimal integer process ID of a command, for which the  
 115331           utility is to wait for the termination.
- 115332           2. A job control job ID (see XBD [Section 3.204](#), on page 66) that identifies a  
 115333           background process group to be waited for. The job control job ID notation  
 115334           is applicable only for invocations of *wait* in the current shell execution  
 115335           environment; see [Section 2.12](#) (on page 2371). The exit status of *wait* shall be  
 115336           determined by the last command in the pipeline.

115337       **Note:**       The job control job ID type of *pid* is only available on systems supporting  
 115338       the User Portability Utilities option.

115339 **STDIN**

115340       Not used.

115341 **INPUT FILES**

115342       None.

115343 **ENVIRONMENT VARIABLES**115344       The following environment variables shall affect the execution of *wait*:

115345       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 115346       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 115347       variables used to determine the values of locale categories.)

115348       *LC\_ALL*      If set to a non-empty string value, override the values of all the other  
 115349       internationalization variables.



115350	<i>LC_CTYPE</i>	Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments).
115351		
115352		
115353	<i>LC_MESSAGES</i>	
115354		Determine the locale that should be used to affect the format and contents of diagnostic messages written to standard error.
115355		
115356	XSI <i>NLSPATH</i>	Determine the location of message catalogs for the processing of <i>LC_MESSAGES</i> .
115357	<b>ASYNCHRONOUS EVENTS</b>	
115358		Default.
115359	<b>STDOUT</b>	
115360		Not used.
115361	<b>STDERR</b>	
115362		The standard error shall be used only for diagnostic messages.
115363	<b>OUTPUT FILES</b>	
115364		None.
115365	<b>EXTENDED DESCRIPTION</b>	
115366		None.
115367	<b>EXIT STATUS</b>	
115368		If one or more operands were specified, all of them have terminated or were not known by the invoking shell, and the status of the last operand specified is known, then the exit status of <i>wait</i> shall be the exit status information of the command indicated by the last operand specified. If the process terminated abnormally due to the receipt of a signal, the exit status shall be greater than 128 and shall be distinct from the exit status generated by other signals, but the exact value is unspecified. (See the <i>kill</i> <i>-l</i> option.) Otherwise, the <i>wait</i> utility shall exit with one of the following values:
115369		
115370		
115371		
115372		
115373		
115374		
115375	0	The <i>wait</i> utility was invoked with no operands and all process IDs known by the invoking shell have terminated.
115376		
115377	1-126	The <i>wait</i> utility detected an error.
115378	127	The command identified by the last <i>pid</i> operand specified is unknown.
115379	<b>CONSEQUENCES OF ERRORS</b>	
115380		Default.
115381	<b>APPLICATION USAGE</b>	
115382		On most implementations, <i>wait</i> is a shell built-in. If it is called in a subshell or separate utility execution environment, such as one of the following:
115383		
115384		
115385		( <i>wait</i> )
115386		<i>nohup wait</i> ...
115387		<i>find . -exec wait</i> ... \;
115388		it returns immediately because there are no known process IDs to wait for in those environments.
115389		Historical implementations of interactive shells have discarded the exit status of terminated background processes before each shell prompt. Therefore, the status of background processes was usually lost unless it terminated while <i>wait</i> was waiting for it. This could be a serious problem when a job that was expected to run for a long time actually terminated quickly with a syntax or initialization error because the exit status returned was usually zero if the requested
115390		
115391		
115392		
115393		



process ID was not found. This volume of POSIX.1-2008 requires the implementation to keep the status of terminated jobs available until the status is requested, so that scripts like:

```
j1&
p1=$!
j2&
wait $p1
echo Job 1 exited with status $?
wait $!
echo Job 2 exited with status $?
```

work without losing status on any of the jobs. The shell is allowed to discard the status of any process if it determines that the application cannot get the process ID for that process from the shell. It is also required to remember only {CHILD\_MAX} number of processes in this way. Since the only way to get the process ID from the shell is by using the '!' shell parameter, the shell is allowed to discard the status of an asynchronous list if "\$!" was not referenced before another asynchronous list was started. (This means that the shell only has to keep the status of the last asynchronous list started if the application did not reference "\$!". If the implementation of the shell is smart enough to determine that a reference to "\$!" was not saved anywhere that the application can retrieve it later, it can use this information to trim the list of saved information. Note also that a successful call to *wait* with no operands discards the exit status of all asynchronous lists.)

If the exit status of *wait* is greater than 128, there is no way for the application to know if the waited-for process exited with that value or was killed by a signal. Since most utilities exit with small values, there is seldom any ambiguity. Even in the ambiguous cases, most applications just need to know that the asynchronous job failed; it does not matter whether it detected an error and failed or was killed and did not complete its job normally.

#### EXAMPLES

Although the exact value used when a process is terminated by a signal is unspecified, if it is known that a signal terminated a process, a script can still reliably determine which signal by using *kill* as shown by the following script:

```
sleep 1000&
pid=$!
kill -kill $pid
wait $pid
echo $pid was terminated by a SIG$(kill -l $?) signal.
```

If the following sequence of commands is run in less than 31 seconds:

```
sleep 257 | sleep 31 &
jobs -l %%
```

either of the following commands returns the exit status of the second *sleep* in the pipeline:

```
wait <pid of sleep 31>
wait %%
```

#### RATIONALE

The description of *wait* does not refer to the *waitpid()* function from the System Interfaces volume of POSIX.1-2008 because that would needlessly overspecify this interface. However, the wording means that *wait* is required to wait for an explicit process when it is given an argument so that the status information of other processes is not consumed. Historical implementations use the *wait()* function defined in the System Interfaces volume of POSIX.1-2008 until *wait()* returns the requested process ID or finds that the requested process does not exist. Because this

115441 means that a shell script could not reliably get the status of all background children if a second  
115442 background job was ever started before the first job finished, it is recommended that the *wait*  
115443 utility use a method such as the functionality provided by the *waitpid()* function.

115444 The ability to wait for multiple *pid* operands was adopted from the KornShell.

115445 This new functionality was added because it is needed to determine the exit status of any  
115446 asynchronous list accurately. The only compatibility problem that this change creates is for a  
115447 script like

```
115448 while sleep 60 do  
115449     job& echo Job started $(date) as $! done
```

115450 which causes the shell to monitor all of the jobs started until the script terminates or runs out of  
115451 memory. This would not be a problem if the loop did not reference "\$!" or if the script would  
115452 occasionally *wait* for jobs it started.

#### 115453 FUTURE DIRECTIONS

115454 None.

#### 115455 SEE ALSO

115456 [Chapter 2](#) (on page 2335), *kill*, *sh*

115457 XBD [Section 3.204](#) (on page 66), [Chapter 8](#) (on page 173)

115458 XSH *wait()*

#### 115459 CHANGE HISTORY

115460 First released in Issue 2.

115461 **NAME**115462 `wc` — word, line, and byte or character count115463 **SYNOPSIS**115464 `wc [-c|-m] [-lw] [file...]`115465 **DESCRIPTION**115466 The *wc* utility shall read one or more input files and, by default, write the number of <newline>  
115467 characters, words, and bytes contained in each input file to the standard output.115468 The utility also shall write a total count for all named files, if more than one input file is  
115469 specified.115470 The *wc* utility shall consider a *word* to be a non-zero-length string of characters delimited by  
115471 white space.115472 **OPTIONS**115473 The *wc* utility shall conform to XBD [Section 12.2](#) (on page 216).

115474 The following options shall be supported:

115475 **-c** Write to the standard output the number of bytes in each input file.115476 **-l** Write to the standard output the number of <newline> characters in each input  
115477 file.115478 **-m** Write to the standard output the number of characters in each input file.115479 **-w** Write to the standard output the number of words in each input file.115480 When any option is specified, *wc* shall report only the information requested by the specified  
115481 options.115482 **OPERANDS**

115483 The following operand shall be supported:

115484 *file* A pathname of an input file. If no *file* operands are specified, the standard input  
115485 shall be used.115486 **STDIN**115487 The standard input shall be used if no *file* operands are specified, and shall be used if a *file*  
115488 operand is '-' and the implementation treats the '-' as meaning standard input. Otherwise,  
115489 the standard input shall not be used. See the INPUT FILES section.115490 **INPUT FILES**

115491 The input files may be of any type.

115492 **ENVIRONMENT VARIABLES**115493 The following environment variables shall affect the execution of *wc*:115494 **LANG** Provide a default value for the internationalization variables that are unset or null.  
115495 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
115496 variables used to determine the values of locale categories.)115497 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
115498 internationalization variables.115499 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
115500 characters (for example, single-byte as opposed to multi-byte characters in  
115501 arguments and input files) and which characters are defined as white-space  
115502 characters.

- 115503 *LC\_MESSAGES*
- 115504 Determine the locale that should be used to affect the format and contents of
- 115505 diagnostic messages written to standard error and informative messages written to
- 115506 standard output.
- 115507 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 115508 **ASYNCHRONOUS EVENTS**
- 115509 Default.
- 115510 **STDOUT**
- 115511 By default, the standard output shall contain an entry for each input file of the form:
- 115512 "%d %d %d %s\n", <newlines>, <words>, <bytes>, <file>
- 115513 If the **-m** option is specified, the number of characters shall replace the <bytes> field in this
- 115514 format.
- 115515 If any options are specified and the **-l** option is not specified, the number of <newline>
- 115516 characters shall not be written.
- 115517 If any options are specified and the **-w** option is not specified, the number of words shall not be
- 115518 written.
- 115519 If any options are specified and neither **-c** nor **-m** is specified, the number of bytes or characters
- 115520 shall not be written.
- 115521 If no input *file* operands are specified, no name shall be written and no <blank> characters
- 115522 preceding the pathname shall be written.
- 115523 If more than one input *file* operand is specified, an additional line shall be written, of the same
- 115524 format as the other lines, except that the word **total** (in the POSIX locale) shall be written instead
- 115525 of a pathname and the total of each column shall be written as appropriate. Such an additional
- 115526 line, if any, is written at the end of the output.
- 115527 **STDERR**
- 115528 The standard error shall be used only for diagnostic messages.
- 115529 **OUTPUT FILES**
- 115530 None.
- 115531 **EXTENDED DESCRIPTION**
- 115532 None.
- 115533 **EXIT STATUS**
- 115534 The following exit values shall be returned:
- 115535 0 Successful completion.
- 115536 >0 An error occurred.
- 115537 **CONSEQUENCES OF ERRORS**
- 115538 Default.

**115539 APPLICATION USAGE**

115540 The **-m** option is not a switch, but an option at the same level as **-c**. Thus, to produce the full  
115541 default output with character counts instead of bytes, the command required is:

115542 `wc -mlw`

**115543 EXAMPLES**

115544 None.

**115545 RATIONALE**

115546 The output file format *pseudo-printf()* string differs from the System V version of *wc*:

115547 `"%7d%7d%7d %s\n"`

115548 which produces possibly ambiguous and unparseable results for very large files, as it assumes no  
115549 number shall exceed six digits.

115550 Some historical implementations use only <space>, <tab>, and <newline> as word separators.  
115551 The equivalent of the ISO C standard *isspace()* function is more appropriate.

115552 The **-c** option stands for “character” count, even though it counts bytes. This stems from the  
115553 sometimes erroneous historical view that bytes and characters are the same size. Due to  
115554 international requirements, the **-m** option (reminiscent of “multi-byte”) was added to obtain  
115555 actual character counts.

115556 Early proposals only specified the results when input files were text files. The current  
115557 specification more closely matches historical practice. (Bytes, words, and <newline> characters  
115558 are counted separately and the results are written when an end-of-file is detected.)

115559 Historical implementations of the *wc* utility only accepted one argument to specify the options  
115560 **-c**, **-l**, and **-w**. Some of them also had multiple occurrences of an option cause the  
115561 corresponding count to be written multiple times and had the order of specification of the  
115562 options affect the order of the fields on output, but did not document either of these. Because  
115563 common usage either specifies no options or only one option, and because none of this was  
115564 documented, the changes required by this volume of POSIX.1-2008 should not break many  
115565 historical applications (and do not break any historical conforming applications).

**115566 FUTURE DIRECTIONS**

115567 None.

**115568 SEE ALSO**

115569 *cksum*

115570 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

**115571 CHANGE HISTORY**

115572 First released in Issue 2.

**115573 Issue 7**

115574 Austin Group Interpretation 1003.1-2001 #092 is applied.

115575 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

115576 **NAME**115577        what — identify SCCS files (**DEVELOPMENT**)115578 **SYNOPSIS**115579 XSI        `what [-s] file...`115580 **DESCRIPTION**

115581        The *what* utility shall search the given files for all occurrences of the pattern that *get* (see *get*)  
 115582        substitutes for the %Z% keyword ("@( # )") and shall write to standard output what follows  
 115583        until the first occurrence of one of the following:

115584        "    &gt;    newline    \    NUL

115585 **OPTIONS**115586        The *what* utility shall conform to XBD [Section 12.2](#) (on page 216).

115587        The following option shall be supported:

115588        -s            Quit after finding the first occurrence of the pattern in each file.

115589 **OPERANDS**

115590        The following operands shall be supported:

115591        *file*            A pathname of a file to search.115592 **STDIN**

115593        Not used.

115594 **INPUT FILES**

115595        The input files shall be of any file type.

115596 **ENVIRONMENT VARIABLES**115597        The following environment variables shall affect the execution of *what*:

115598        *LANG*            Provide a default value for the internationalization variables that are unset or null.  
 115599                          (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 115600                          variables used to determine the values of locale categories.)

115601        *LC\_ALL*          If set to a non-empty string value, override the values of all the other  
 115602                          internationalization variables.

115603        *LC\_CTYPE*        Determine the locale for the interpretation of sequences of bytes of text data as  
 115604                          characters (for example, single-byte as opposed to multi-byte characters in  
 115605                          arguments and input files).

115606        *LC\_MESSAGES*

115607                          Determine the locale that should be used to affect the format and contents of  
 115608                          diagnostic messages written to standard error.

115609        *NLSPATH*        Determine the location of message catalogs for the processing of *LC\_MESSAGES*.115610 **ASYNCHRONOUS EVENTS**

115611        Default.

115612 **STDOUT**115613        The standard output shall consist of the following for each *file* operand:

115614        "%s:\n\t%s\n", &lt;pathname&gt;, &lt;identification string&gt;

**115615 STDERR**

115616 The standard error shall be used only for diagnostic messages.

**115617 OUTPUT FILES**

115618 None.

**115619 EXTENDED DESCRIPTION**

115620 None.

**115621 EXIT STATUS**

115622 The following exit values shall be returned:

115623 0 Any matches were found.

115624 1 Otherwise.

**115625 CONSEQUENCES OF ERRORS**

115626 Default.

**115627 APPLICATION USAGE**

115628 The *what* utility is intended to be used in conjunction with the SCCS command *get*, which  
115629 automatically inserts identifying information, but it can also be used where the information is  
115630 inserted by any other means.

115631 When the string "@(#)" is included in a library routine in a shared library, it might not be found  
115632 in an **a.out** file using that library routine.

**115633 EXAMPLES**

115634 If the C-language program in file **f.c** contains:

115635 `char ident[] = "@(#)identification information";`

115636 and **f.c** is compiled to yield **f.o** and **a.out**, then the command:

115637 `what f.c f.o a.out`

115638 writes:

115639 **f.c:**  
115640     identification information  
115641     ...  
115642 **f.o:**  
115643     identification information  
115644     ...  
115645 **a.out:**  
115646     identification information  
115647     ...

**115648 RATIONALE**

115649 None.

**115650 FUTURE DIRECTIONS**

115651 None.

**115652 SEE ALSO**

115653 *get*

115654 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

115655 **CHANGE HISTORY**  
115656 First released in Issue 2.



115657 **NAME**

115658       who — display who is on the system

115659 **SYNOPSIS**

115660 XSI       who [-mTu] [-abdHlprt] [file]

115661 XSI       who [-mu] -s [-bHlprt] [file]

115662       who -q [file]

115663       who am i

115664       who am I

115665 **DESCRIPTION**115666       The *who* utility shall list various pieces of information about accessible users. The domain of  
115667       accessibility is implementation-defined.115668 XSI       Based on the options given, *who* can also list the user's name, terminal line, login time, elapsed  
115669       time since activity occurred on the line, and the process ID of the command interpreter for each  
115670       current system user.115671 **OPTIONS**115672       The *who* utility shall conform to XBD [Section 12.2](#) (on page 216).115673       The following options shall be supported. The metavariables, such as *<line>*, refer to fields  
115674       described in the STDOUT section.115675 XSI       **-a**       Process the implementation-defined database or named file with the **-b**, **-d**, **-l**, **-p**,  
115676       **-r**, **-t**, **-T** and **-u** options turned on.115677 XSI       **-b**       Write the time and date of the last system reboot. The system reboot time is the  
115678       time at which the implementation is able to commence running processes.115679 XSI       **-d**       Write a list of all processes that have expired and not been respawned by the *init*  
115680       system process. The *<exit>* field shall appear for dead processes and contain the  
115681       termination and exit values of the dead process. This can be useful in determining  
115682       why a process terminated.115683 XSI       **-H**       Write column headings above the regular output.115684 XSI       **-l**       (The letter ell.) List only those lines on which the system is waiting for someone to  
115685       login. The *<name>* field shall be **LOGIN** in such cases. Other fields shall be the  
115686       same as for user entries except that the *<state>* field does not exist.115687       **-m**       Output only information about the current terminal.115688 XSI       **-p**       List any other process that is currently active and has been previously spawned by  
115689       *init*.115690 XSI       **-q**       (Quick.) List only the names and the number of users currently logged on. When  
115691       this option is used, all other options shall be ignored.115692 XSI       **-r**       Write the current *run-level* of the *init* process.115693 XSI       **-s**       List only the *<name>*, *<line>*, and *<time>* fields. This is the default case.115694 XSI       **-t**       Indicate the last change to the system clock.115695       **-T**       Show the state of each terminal, as described in the STDOUT section.

115696        **-u**        Write “idle time” for each displayed user in addition to any other information. The  
 115697 idle time is the time since any activity occurred on the user’s terminal. The method  
 115698 XSI of determining this is unspecified. This option shall list only those users who are  
 115699 currently logged in. The *<name>* is the user’s login name. The *<line>* is the name  
 115700 of the line as found in the directory */dev*. The *<time>* is the time that the user  
 115701 logged in. The *<activity>* is the number of hours and minutes since activity last  
 115702 occurred on that particular line. A dot indicates that the terminal has seen activity  
 115703 in the last minute and is therefore “current”. If more than twenty-four hours have  
 115704 elapsed or the line has not been used since boot time, the entry shall be marked  
 115705 *<old>*. This field is useful when trying to determine whether a person is working at  
 115706 the terminal or not. The *<pid>* is the process ID of the user’s login process.

#### 115707 OPERANDS

115708 XSI        The following operands shall be supported:

115709        **am i, am I**        In the POSIX locale, limit the output to describing the invoking user, equivalent to  
 115710 the **-m** option. The **am** and **i** or **I** must be separate arguments.

115711        *file*                Specify a pathname of a file to substitute for the implementation-defined database  
 115712 of logged-on users that *who* uses by default.

#### 115713 STDIN

115714        Not used.

#### 115715 INPUT FILES

115716        None.

#### 115717 ENVIRONMENT VARIABLES

115718        The following environment variables shall affect the execution of *who*:

115719        *LANG*                Provide a default value for the internationalization variables that are unset or null.  
 115720 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 115721 variables used to determine the values of locale categories.)

115722        *LC\_ALL*                If set to a non-empty string value, override the values of all the other  
 115723 internationalization variables.

115724        *LC\_CTYPE*             Determine the locale for the interpretation of sequences of bytes of text data as  
 115725 characters (for example, single-byte as opposed to multi-byte characters in  
 115726 arguments).

115727        *LC\_MESSAGES*           Determine the locale that should be used to affect the format and contents of  
 115728 diagnostic messages written to standard error.

115729        *LC\_TIME*                Determine the locale used for the format and contents of the date and time strings.

115730        *NLSPATH*             Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

115731 XSI        *TZ*                        Determine the timezone used when writing date and time information. If *TZ* is  
 115732 unset or null, an unspecified default timezone shall be used.

#### 115734 ASYNCHRONOUS EVENTS

115735        Default.

#### 115736 STDOUT

115737        The *who* utility shall write its default format to the standard output in an implementation-  
 115738 defined format, subject only to the requirement of containing the information described above.

115739 XSI OF XSI-conformant systems shall write the default information to the standard output in the  
 115740 following general format:

115741 `<name>[<state>]<line><time>[<activity>][<pid>][<comment>][<exit>]`

115742 For the `-b` option, `<line>` shall be "system boot". The `<name>` is unspecified.

115743 The following format shall be used for the `-T` option:

115744 `"%s %c %s %s\n" <name>, <terminal state>, <terminal name>,  
 115745 <time of login>`

115746 where `<terminal state>` is one of the following characters:

115747 + The terminal allows write access to other users.

115748 – The terminal denies write access to other users.

115749 ? The terminal write-access state cannot be determined.

115750 `<space>` This entry is not associated with a terminal.

115751 In the POSIX locale, the `<time of login>` shall be equivalent in format to the output of:

115752 `date +"%b %e %H:%M"`

115753 If the `-u` option is used with `-T`, the idle time shall be added to the end of the previous format in  
 115754 an unspecified format.

#### 115755 **STDERR**

115756 The standard error shall be used only for diagnostic messages.

#### 115757 **OUTPUT FILES**

115758 None.

#### 115759 **EXTENDED DESCRIPTION**

115760 None.

#### 115761 **EXIT STATUS**

115762 The following exit values shall be returned:

115763 0 Successful completion.

115764 >0 An error occurred.

#### 115765 **CONSEQUENCES OF ERRORS**

115766 Default.

#### 115767 **APPLICATION USAGE**

115768 The name *init* used for the system process is the most commonly used on historical systems, but  
 115769 it may vary.

115770 The “domain of accessibility” referred to is a broad concept that permits interpretation either on  
 115771 a very secure basis or even to allow a network-wide implementation like the historical *rwwho*.

#### 115772 **EXAMPLES**

115773 None.

#### 115774 **RATIONALE**

115775 Due to differences between historical implementations, the base options provided were a  
 115776 compromise to allow users to work with those functions. The standard developers also  
 115777 considered removing all the options, but felt that these options offered users valuable  
 115778 functionality. Additional options to match historical systems are available on XSI-conformant

- 115779 systems.
- 115780 It is recognized that the *who* command may be of limited usefulness, especially in a multi-level  
115781 secure environment. The standard developers considered, however, that having some standard  
115782 method of determining the “accessibility” of other users would aid user portability.
- 115783 No format was specified for the default *who* output for systems not supporting the XSI option. In  
115784 such a user-oriented command, designed only for human use, this was not considered to be a  
115785 deficiency.
- 115786 The format of the terminal name is unspecified, but the descriptions of *ps*, *talk*, and *write* require  
115787 that they use the same format.
- 115788 It is acceptable for an implementation to produce no output for an invocation of *who* **mil**.
- 115789 **FUTURE DIRECTIONS**
- 115790 None.
- 115791 **SEE ALSO**
- 115792 *mesg*
- 115793 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)
- 115794 **CHANGE HISTORY**
- 115795 First released in Issue 2.
- 115796 **Issue 6**
- 115797 This utility is marked as part of the User Portability Utilities option.
- 115798 The *TZ* entry is added to the ENVIRONMENT VARIABLES section.
- 115799 **Issue 7**
- 115800 SD5-XCU-ERN-58 is applied, clarifying the **-b** option.
- 115801 The *who* utility is moved from the User Portability Utilities option to the Base. User Portability  
115802 Utilities is now an option for interactive utilities.
- 115803 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

115804 **NAME**

115805 write — write to another user

115806 **SYNOPSIS**115807 write *user\_name* [*terminal*]115808 **DESCRIPTION**

115809 The *write* utility shall read lines from the standard input and write them to the terminal of the  
 115810 specified user. When first invoked, it shall write the message:

115811 **Message from sender-login-id** (*sending-terminal*) [*date*]...

115812 to *user\_name*. When it has successfully completed the connection, the sender's terminal shall be  
 115813 alerted twice to indicate that what the sender is typing is being written to the recipient's  
 115814 terminal.

115815 If the recipient wants to reply, this can be accomplished by typing:

115816 write *sender-login-id* [*sending-terminal*]

115817 upon receipt of the initial message. Whenever a line of input as delimited by an NL, EOF, or  
 115818 EOL special character (see XBD [Chapter 11](#), on page 199) is accumulated while in canonical  
 115819 input mode, the accumulated data shall be written on the other user's terminal. Characters shall  
 115820 be processed as follows:

- 115821 • Typing <alert> shall write the <alert> character to the recipient's terminal.
- 115822 • Typing the erase and kill characters shall affect the sender's terminal in the manner  
 115823 described by the **termios** interface in XBD [Chapter 11](#) (on page 199).
- 115824 • Typing the interrupt or end-of-file characters shall cause *write* to write an appropriate  
 115825 message ("EOT\n" in the POSIX locale) to the recipient's terminal and exit.
- 115826 • Typing characters from *LC\_CTYPE* classifications **print** or **space** shall cause those  
 115827 characters to be sent to the recipient's terminal.
- 115828 • When and only when the *stty* **ixten** local mode is enabled, the existence and processing of  
 115829 additional special control characters and multi-byte or single-byte functions is  
 115830 implementation-defined.
- 115831 • Typing other non-printable characters shall cause implementation-defined sequences of  
 115832 printable characters to be written to the recipient's terminal.

115833 To write to a user who is logged in more than once, the *terminal* argument can be used to  
 115834 indicate which terminal to write to; otherwise, the recipient's terminal is selected in an  
 115835 implementation-defined manner and an informational message is written to the sender's  
 115836 standard output, indicating which terminal was chosen.

115837 Permission to be a recipient of a *write* message can be denied or granted by use of the *mesg*  
 115838 utility. However, a user's privilege may further constrain the domain of accessibility of other  
 115839 users' terminals. The *write* utility shall fail when the user lacks appropriate privileges to perform  
 115840 the requested action.

115841 **OPTIONS**

115842 None.

115843 **OPERANDS**

- 115844 The following operands shall be supported:
- 115845 *user\_name* Login name of the person to whom the message shall be written. The application  
115846 shall ensure that this operand is of the form returned by the *who* utility.
- 115847 *terminal* Terminal identification in the same format provided by the *who* utility.
- 115848 **STDIN**
- 115849 Lines to be copied to the recipient's terminal are read from standard input.
- 115850 **INPUT FILES**
- 115851 None.
- 115852 **ENVIRONMENT VARIABLES**
- 115853 The following environment variables shall affect the execution of *write*:
- 115854 *LANG* Provide a default value for the internationalization variables that are unset or null.  
115855 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
115856 variables used to determine the values of locale categories.)
- 115857 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
115858 internationalization variables.
- 115859 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
115860 characters (for example, single-byte as opposed to multi-byte characters in  
115861 arguments and input files). If the recipient's locale does not use an *LC\_CTYPE*  
115862 equivalent to the sender's, the results are undefined.
- 115863 *LC\_MESSAGES*
- 115864 Determine the locale that should be used to affect the format and contents of  
115865 diagnostic messages written to standard error and informative messages written to  
115866 standard output.
- 115867 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.
- 115868 **ASYNCHRONOUS EVENTS**
- 115869 If an interrupt signal is received, *write* shall write an appropriate message on the recipient's  
115870 terminal and exit with a status of zero. It shall take the standard action for all other signals.
- 115871 **STDOUT**
- 115872 An informational message shall be written to standard output if a recipient is logged in more  
115873 than once.
- 115874 **STDERR**
- 115875 The standard error shall be used only for diagnostic messages.
- 115876 **OUTPUT FILES**
- 115877 The recipient's terminal is used for output.
- 115878 **EXTENDED DESCRIPTION**
- 115879 None.
- 115880 **EXIT STATUS**
- 115881 The following exit values shall be returned:
- 115882 0 Successful completion.
- 115883 >0 The addressed user is not logged on or the addressed user denies permission.

**115884 CONSEQUENCES OF ERRORS**

115885 Default.

**115886 APPLICATION USAGE**

115887 The *talk* utility is considered by some users to be a more usable utility on full-screen terminals.

**115888 EXAMPLES**

115889 None.

**115890 RATIONALE**

115891 The *write* utility was included in this volume of POSIX.1-2008 since it can be implemented on all  
115892 terminal types. The standard developers considered the *talk* utility, which cannot be  
115893 implemented on certain terminals, to be a “better” communications interface. Both of these  
115894 programs are in widespread use on historical implementations. Therefore, the standard  
115895 developers decided that both utilities should be specified.

115896 The format of the terminal name is unspecified, but the descriptions of *ps*, *talk*, *who*, and *write*  
115897 require that they all use or accept the same format.

**115898 FUTURE DIRECTIONS**

115899 None.

**115900 SEE ALSO**

115901 *mesg*, *talk*, *who*

115902 XBD [Chapter 8](#) (on page 173), [Chapter 11](#) (on page 199)

**115903 CHANGE HISTORY**

115904 First released in Issue 2.

**115905 Issue 5**

115906 The FUTURE DIRECTIONS section is added.

**115907 Issue 6**

115908 This utility is marked as part of the User Portability Utilities option.

115909 The normative text is reworded to avoid use of the term “must” for application requirements.

**115910 Issue 7**

115911 The *write* utility is moved from the User Portability Utilities option to the Base. User Portability  
115912 Utilities is now an option for interactive utilities.



115913 **NAME**115914 **xargs** — construct argument lists and invoke utility115915 **SYNOPSIS**

```
115916 XSI      xargs [-ptx] [-E eofstr] [-I replstr|-L number] [-n number]
115917          [-s size] [utility [argument...]]
```

115918 **DESCRIPTION**

115919 The *xargs* utility shall construct a command line consisting of the *utility* and *argument* operands  
 115920 specified followed by as many arguments read in sequence from standard input as fit in length  
 115921 and number constraints specified by the options. The *xargs* utility shall then invoke the  
 115922 constructed command line and wait for its completion. This sequence shall be repeated until one  
 115923 of the following occurs:

- 115924 • An end-of-file condition is detected on standard input.
- 115925 • An argument consisting of just the logical end-of-file string (see the `-E eofstr` option) is  
 115926 found on standard input after double-quote processing, <apostrophe> processing, and  
 115927 <backslash>-escape processing (see next paragraph). All arguments up to but not  
 115928 including the argument consisting of just the logical end-of-file string shall be used as  
 115929 arguments in constructed command lines.
- 115930 • An invocation of a constructed command line returns an exit status of 255.

115931 The application shall ensure that arguments in the standard input are separated by unquoted  
 115932 <blank> characters, unescaped <blank> characters, or <newline> characters. A string of zero or  
 115933 more non-double-quote ( ' " ' ) characters and non-<newline> characters can be quoted by  
 115934 enclosing them in double-quotes. A string of zero or more non-<apostrophe> ( ' \ ' ' ) characters  
 115935 and non-<newline> characters can be quoted by enclosing them in <apostrophe> characters.  
 115936 Any unquoted character can be escaped by preceding it with a <backslash>. The utility named  
 115937 by *utility* shall be executed one or more times until the end-of-file is reached or the logical end-of  
 115938 file string is found. The results are unspecified if the utility named by *utility* attempts to read  
 115939 from its standard input.

115940 The generated command line length shall be the sum of the size in bytes of the utility name and  
 115941 each argument treated as strings, including a null byte terminator for each of these strings. The  
 115942 *xargs* utility shall limit the command line length such that when the command line is invoked,  
 115943 the combined argument and environment lists (see the *exec* family of functions in the System  
 115944 Interfaces volume of POSIX.1-2008) shall not exceed {ARG\_MAX}-2048 bytes. Within this  
 115945 constraint, if neither the `-n` nor the `-s` option is specified, the default command line length shall  
 115946 be at least {LINE\_MAX}.

115947 **OPTIONS**115948 The *xargs* utility shall conform to XBD [Section 12.2](#) (on page 216).

115949 The following options shall be supported:

115950 `-E eofstr` Use *eofstr* as the logical end-of-file string. If `-E` is not specified, it is unspecified  
 115951 whether the logical end-of-file string is the <underscore> character ( ' \_ ' ) or the  
 115952 end-of-file string capability is disabled. When *eofstr* is the null string, the logical  
 115953 end-of-file string capability shall be disabled and <underscore> characters shall be  
 115954 taken literally.

115955 XSI `-I replstr` Insert mode: *utility* is executed for each logical line from standard input.  
 115956 Arguments in the standard input shall be separated only by unescaped <newline>  
 115957 characters, not by <blank> characters. Any unquoted unescaped <blank>  
 115958 characters at the beginning of each line shall be ignored. The resulting argument  
 115959 shall be inserted in *arguments* in place of each occurrence of *replstr*. At least five



115960		arguments in <i>arguments</i> can each contain one or more instances of <i>replstr</i> . Each of
115961		these constructed arguments cannot grow larger than an implementation-defined
115962		limit greater than or equal to 255 bytes. Option <b>-x</b> shall be forced on.
115963	XSI	<b>-L number</b> The <i>utility</i> shall be executed for each non-empty <i>number</i> lines of arguments from
115964		standard input. The last invocation of <i>utility</i> shall be with fewer lines of arguments
115965		if fewer than <i>number</i> remain. A line is considered to end with the first <newline>
115966		unless the last character of the line is an unescaped <blank>; a trailing unescaped
115967		<blank> signals continuation to the next non-empty line, inclusive.
115968		<b>-n number</b> Invoke <i>utility</i> using as many standard input arguments as possible, up to <i>number</i> (a
115969		positive decimal integer) arguments maximum. Fewer arguments shall be used if:
115970		• The command line length accumulated exceeds the size specified by the <b>-s</b>
115971		option (or {LINE_MAX} if there is no <b>-s</b> option).
115972		• The last iteration has fewer than <i>number</i> , but not zero, operands remaining.
115973		<b>-p</b> Prompt mode: the user is asked whether to execute <i>utility</i> at each invocation. Trace
115974		mode ( <b>-t</b> ) is turned on to write the command instance to be executed, followed by
115975		a prompt to standard error. An affirmative response read from <b>/dev/tty</b> shall
115976		execute the command; otherwise, that particular invocation of <i>utility</i> shall be
115977		skipped.
115978		<b>-s size</b> Invoke <i>utility</i> using as many standard input arguments as possible yielding a
115979		command line length less than <i>size</i> (a positive decimal integer) bytes. Fewer
115980		arguments shall be used if:
115981		• The total number of arguments exceeds that specified by the <b>-n</b> option.
115982	XSI	• The total number of lines exceeds that specified by the <b>-L</b> option.
115983		• End-of-file is encountered on standard input before <i>size</i> bytes are
115984		accumulated.
115985		Values of <i>size</i> up to at least {LINE_MAX} bytes shall be supported, provided that
115986		the constraints specified in the DESCRIPTION are met. It shall not be considered
115987		an error if a value larger than that supported by the implementation or exceeding
115988		the constraints specified in the DESCRIPTION is given; <i>xargs</i> shall use the largest
115989		value it supports within the constraints.
115990		<b>-t</b> Enable trace mode. Each generated command line shall be written to standard
115991		error just prior to invocation.
115992		<b>-x</b> Terminate if a constructed command line will not fit in the implied or specified size
115993		(see the <b>-s</b> option above).

## 115994 OPERANDS

115995 The following operands shall be supported:

115996	<i>utility</i>	The name of the utility to be invoked, found by search path using the <i>PATH</i>
115997		environment variable, described in XBD Chapter 8 (on page 173). If <i>utility</i> is
115998		omitted, the default shall be the <i>echo</i> utility. If the <i>utility</i> operand names any of the
115999		special built-in utilities in Section 2.14 (on page 2374), the results are undefined.
116000	<i>argument</i>	An initial option or operand for the invocation of <i>utility</i> .

116001 **STDIN**

116002 The standard input shall be a text file. The results are unspecified if an end-of-file condition is  
 116003 detected immediately following an escaped <newline>.

116004 **INPUT FILES**

116005 The file `/dev/tty` shall be used to read responses required by the `-p` option.

116006 **ENVIRONMENT VARIABLES**

116007 The following environment variables shall affect the execution of *xargs*:

116008 **LANG** Provide a default value for the internationalization variables that are unset or null.  
 116009 (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 116010 variables used to determine the values of locale categories.)

116011 **LC\_ALL** If set to a non-empty string value, override the values of all the other  
 116012 internationalization variables.

116013 **LC\_COLLATE**

116014 Determine the locale for the behavior of ranges, equivalence classes, and multi-  
 116015 character collating elements used in the extended regular expression defined for  
 116016 the **yesexpr** locale keyword in the *LC\_MESSAGES* category.

116017 **LC\_CTYPE** Determine the locale for the interpretation of sequences of bytes of text data as  
 116018 characters (for example, single-byte as opposed to multi-byte characters in  
 116019 arguments and input files) and the behavior of character classes used in the  
 116020 extended regular expression defined for the **yesexpr** locale keyword in the  
 116021 *LC\_MESSAGES* category.

116022 **LC\_MESSAGES**

116023 Determine the locale used to process affirmative responses, and the locale used to  
 116024 affect the format and contents of diagnostic messages and prompts written to  
 116025 standard error.

116026 XSI **NLSPATH** Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

116027 **PATH** Determine the location of *utility*, as described in XBD [Chapter 8](#) (on page 173).

116028 **ASYNCHRONOUS EVENTS**

116029 Default.

116030 **STDOUT**

116031 Not used.

116032 **STDERR**

116033 The standard error shall be used for diagnostic messages and the `-t` and `-p` options. If the `-t`  
 116034 option is specified, the *utility* and its constructed argument list shall be written to standard error,  
 116035 as it will be invoked, prior to invocation. If `-p` is specified, a prompt of the following format  
 116036 shall be written (in the POSIX locale):

116037 " ? . . . "

116038 at the end of the line of the output from `-t`.

116039 **OUTPUT FILES**

116040 None.

116041 **EXTENDED DESCRIPTION**

116042 None.

116043 **EXIT STATUS**

116044 The following exit values shall be returned:

116045 0 All invocations of *utility* returned exit status zero.

116046 1-125 A command line meeting the specified requirements could not be assembled, one or  
116047 more of the invocations of *utility* returned a non-zero exit status, or some other error  
116048 occurred.

116049 126 The utility specified by *utility* was found but could not be invoked.116050 127 The utility specified by *utility* could not be found.116051 **CONSEQUENCES OF ERRORS**

116052 If a command line meeting the specified requirements cannot be assembled, the utility cannot be  
116053 invoked, an invocation of the utility is terminated by a signal, or an invocation of the utility exits  
116054 with exit status 255, the *xargs* utility shall write a diagnostic message and exit without  
116055 processing any remaining input.

116056 **APPLICATION USAGE**

116057 The 255 exit status allows a utility being used by *xargs* to tell *xargs* to terminate if it knows no  
116058 further invocations using the current data stream will succeed. Thus, *utility* should explicitly *exit*  
116059 with an appropriate value to avoid accidentally returning with 255.

116060 Note that since input is parsed as lines, <blank> characters separate arguments, and  
116061 <backslash>, <apostrophe>, and double-quote characters are used for quoting, if *xargs* is used to  
116062 bundle the output of commands like *find dir -print* or *ls* into commands to be executed,  
116063 unexpected results are likely if any filenames contain <blank>, <newline>, or quoting characters.  
116064 This can be solved by using *find* to call a script that converts each file found into a quoted string  
116065 that is then piped to *xargs*, but in most cases it is preferable just to have *find* do the argument  
116066 aggregation itself by using *-exec* with a '+' terminator instead of ';' . Note that the quoting  
116067 rules used by *xargs* are not the same as in the shell. They were not made consistent here because  
116068 existing applications depend on the current rules. An easy (but inefficient) method that can be  
116069 used to transform input consisting of one argument per line into a quoted form that *xargs*  
116070 interprets correctly is to precede each non-<newline> character with a <backslash>. More  
116071 efficient alternatives are shown in Example 2 and Example 5 below.

116072 On implementations with a large value for {ARG\_MAX}, *xargs* may produce command lines  
116073 longer than {LINE\_MAX}. For invocation of utilities, this is not a problem. If *xargs* is being used  
116074 to create a text file, users should explicitly set the maximum command line length with the *-s*  
116075 option.

116076 The *command*, *env*, *nice*, *nohup*, *time*, and *xargs* utilities have been specified to use exit code 127 if  
116077 an error occurs so that applications can distinguish "failure to find a utility" from "invoked  
116078 utility exited with an error indication". The value 127 was chosen because it is not commonly  
116079 used for other meanings; most utilities use small values for "normal error conditions" and the  
116080 values above 128 can be confused with termination due to receipt of a signal. The value 126 was  
116081 chosen in a similar manner to indicate that the utility could be found, but not invoked. Some  
116082 scripts produce meaningful error messages differentiating the 126 and 127 cases. The distinction  
116083 between exit codes 126 and 127 is based on KornShell practice that uses 127 when all attempts to  
116084 *exec* the utility fail with [ENOENT], and uses 126 when any attempt to *exec* the utility fails for  
116085 any other reason.

## 116086 EXAMPLES

- 116087 1. The following command combines the output of the parenthesized commands (minus the  
 116088 <apostrophe> characters) onto one line, which is then appended to the file log. It assumes  
 116089 that the expansion of "\$0 \$" does not include any <apostrophe> or <newline>  
 116090 characters.

116091 (logname; date; printf "'%s'\n" "\$0 \$") | xargs -E "" >>log

- 116092 2. The following command invokes *diff* with successive pairs of arguments originally typed  
 116093 as command line arguments. It assumes there are no embedded <newline> characters in  
 116094 the elements of the original argument list.

116095 printf "%s\n" "\$@" | sed 's/^[^:alnum:]]/\&/g' |  
 116096 xargs -E "" -n 2 -x diff

- 116097 3. In the following commands, the user is asked which files in the current directory  
 116098 (excluding dotfiles) are to be archived. The files are archived into **arch**; *a*, one at a time or  
 116099 *b*, many at a time. The commands assume that no filenames contain <blank>, <newline>,  
 116100 <backslash>, <apostrophe>, or double-quote characters.

116101 a. ls | xargs -E "" -p -L 1 ar -r arch

116102 b. ls | xargs -E "" -p -L 1 | xargs -E "" ar -r arch

- 116103 4. The following command invokes *command1* one or more times with multiple arguments,  
 116104 stopping if an invocation of *command1* has a non-zero exit status.

116105 xargs -E "" sh -c 'command1 "\$@" || exit 255' sh < xargs\_input

- 116106 5. On XSI-conformant systems, the following command moves all files from directory **\$1** to  
 116107 directory **\$2**, and echoes each move command just before doing it. It assumes no  
 116108 filenames contain <newline> characters and that neither **\$1** nor **\$2** contains the sequence  
 116109 "{ }".

116110 ls -A "\$1" | sed -e 's/"/"\&"/g' -e 's/.\*/"&"/' |  
 116111 xargs -E "" -I {} -t mv "\$1"/{} "\$2"/{ }

## 116112 RATIONALE

116113 The *xargs* utility was usually found only in System V-based systems; BSD systems included an  
 116114 *apply* utility that provided functionality similar to *xargs -n number*. The SVID lists *xargs* as a  
 116115 software development extension. This volume of POSIX.1-2008 does not share the view that it is  
 116116 used only for development, and therefore it is not optional.

116117 The classic application of the *xargs* utility is in conjunction with the *find* utility to reduce the  
 116118 number of processes launched by a simplistic use of the *find -exec* combination. The *xargs* utility  
 116119 is also used to enforce an upper limit on memory required to launch a process. With this basis in  
 116120 mind, this volume of POSIX.1-2008 selected only the minimal features required.

116121 Although the 255 exit status is mostly an accident of historical implementations, it allows a  
 116122 utility being used by *xargs* to tell *xargs* to terminate if it knows no further invocations using the  
 116123 current data stream shall succeed. Any non-zero exit status from a utility falls into the 1-125  
 116124 range when *xargs* exits. There is no statement of how the various non-zero utility exit status  
 116125 codes are accumulated by *xargs*. The value could be the addition of all codes, their highest  
 116126 value, the last one received, or a single value such as 1. Since no algorithm is arguably better  
 116127 than the others, and since many of the standard utilities say little more (portably) than  
 116128 "pass/fail", no new algorithm was invented.

116129 Several other *xargs* options were removed because simple alternatives already exist within this

volume of POSIX.1-2008. For example, the `-i replstr` option can be just as efficiently performed using a shell **for** loop. Since *xargs* calls an *exec* function with each input line, the `-i` option does not usually exploit the grouping capabilities of *xargs*.

The requirement that *xargs* never produces command lines such that invocation of *utility* is within 2048 bytes of hitting the POSIX *exec* {ARG\_MAX} limitations is intended to guarantee that the invoked utility has room to modify its environment variables and command line arguments and still be able to invoke another utility. Note that the minimum {ARG\_MAX} allowed by the System Interfaces volume of POSIX.1-2008 is 4096 bytes and the minimum value allowed by this volume of POSIX.1-2008 is 2048 bytes; therefore, the 2048 bytes difference seems reasonable. Note, however, that *xargs* may never be able to invoke a utility if the environment passed in to *xargs* comes close to using {ARG\_MAX} bytes.

The version of *xargs* required by this volume of POSIX.1-2008 is required to wait for the completion of the invoked command before invoking another command. This was done because historical scripts using *xargs* assumed sequential execution. Implementations wanting to provide parallel operation of the invoked utilities are encouraged to add an option enabling parallel invocation, but should still wait for termination of all of the children before *xargs* terminates normally.

The `-e` option was omitted from the ISO POSIX-2:1993 standard in the belief that the *eofstr* option-argument was recognized only when it was on a line by itself and before quote and escape processing were performed, and that the logical end-of-file processing was only enabled if a `-e` option was specified. In that case, a simple *sed* script could be used to duplicate the `-e` functionality. Further investigation revealed that:

- The logical end-of-file string was checked for after quote and escape processing, making a *sed* script that provided equivalent functionality much more difficult to write.
- The default was to perform logical end-of-file processing with an `<underscore>` as the logical end-of-file string.

To correct this misunderstanding, the `-E eofstr` option was adopted from the X/Open Portability Guide. Users should note that the description of the `-E` option matches historical documentation of the `-e` option (which was not adopted because it did not support the Utility Syntax Guidelines), by saying that if *eofstr* is the null string, logical end-of-file processing is disabled. Historical implementations of *xargs* actually did not disable logical end-of-file processing; they treated a null argument found in the input as a logical end-of-file string. (A null *string* argument could be generated using single or double-quotes (' ' or " "). Since this behavior was not documented historically, it is considered to be a bug.

The `-I`, `-L`, and `-n` options are mutually-exclusive. Some implementations use the last one specified if more than one is given on a command line; other implementations treat combinations of the options in different ways.

## FUTURE DIRECTIONS

None.

## SEE ALSO

[Chapter 2](#) (on page 2335), [diff](#), [echo](#), [find](#)

[XBD Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

[XSH exec](#)

**CHANGE HISTORY**

116173 First released in Issue 2.

**Issue 5**

116176 A second FUTURE DIRECTION is added.

**Issue 6**

116178 The obsolescent `-e`, `-i`, and `-l` options are removed.

116179 The following new requirements on POSIX implementations derive from alignment with the  
116180 Single UNIX Specification:

- 116181 • The `-p` option is added.
- 116182 • In the INPUT FILES section, the file `/dev/tty` is used to read responses required by the `-p`  
116183 option.
- 116184 • The STDERR section is updated to describe the `-p` option.

116185 The description of the `-E` option is aligned with the ISO POSIX-2: 1993 standard.

116186 The normative text is reworded to avoid use of the term “must” for application requirements.

**Issue 7**

116188 Austin Group Interpretation 1003.1-2001 #123 is applied, changing the description of the `xargs -I`  
116189 option.

116190 Austin Group Interpretation 1003.1-2001 #126 is applied, changing the description of the  
116191 `LC_MESSAGES` environment variable.

116192 SD5-XCU-ERN-68 is applied.

116193 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

116194 SD5-XCU-ERN-128 is applied, clarifying the DESCRIPTION of the logical end-of-file string.

116195 SD5-XCU-ERN-132 is applied, updating the EXAMPLES section.

116196 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0149 [342] is applied.



116197 **NAME**116198 yacc — yet another compiler compiler (**DEVELOPMENT**)116199 **SYNOPSIS**116200 CD `yacc [-dltv] [-b file_prefix] [-p sym_prefix] grammar`116201 **DESCRIPTION**

116202 The *yacc* utility shall read a description of a context-free grammar in *grammar* and write C source  
 116203 code, conforming to the ISO C standard, to a code file, and optionally header information into a  
 116204 header file, in the current directory. The generated source code shall not depend on any  
 116205 undefined, unspecified, or implementation-defined behavior, except in cases where it is copied  
 116206 directly from the supplied grammar, or in cases that are documented by the implementation.  
 116207 The C code shall define a function and related routines and macros for an automaton that  
 116208 executes a parsing algorithm meeting the requirements in [Algorithms](#) (on page 3454).

116209 The form and meaning of the grammar are described in the EXTENDED DESCRIPTION section.

116210 The C source code and header file shall be produced in a form suitable as input for the C  
 116211 compiler (see [c99](#)).

116212 **OPTIONS**

116213 The *yacc* utility shall conform to XBD [Section 12.2](#) (on page 216), except for Guideline 9.

116214 The following options shall be supported:

116215 **-b *file\_prefix*** Use *file\_prefix* instead of **y** as the prefix for all output filenames. The code file  
 116216 **y.tab.c**, the header file **y.tab.h** (created when **-d** is specified), and the description  
 116217 file **y.output** (created when **-v** is specified), shall be changed to *file\_prefix.tab.c*,  
 116218 *file\_prefix.tab.h*, and *file\_prefix.output*, respectively.

116219 **-d** Write the header file; by default only the code file is written. See the OUTPUT  
 116220 FILES section.

116221 **-l** Produce a code file that does not contain any **#line** constructs. If this option is not  
 116222 present, it is unspecified whether the code file or header file contains **#line**  
 116223 directives. This should only be used after the grammar and the associated actions  
 116224 are fully debugged.

116225 **-p *sym\_prefix***  
 116226 Use *sym\_prefix* instead of **yy** as the prefix for all external names produced by *yacc*.  
 116227 The names affected shall include the functions *yyparse()*, *yylex()*, and *yyerror()*,  
 116228 and the variables *yyval*, *yychar*, and *yydebug*. (In the remainder of this section, the  
 116229 six symbols cited are referenced using their default names only as a notational  
 116230 convenience.) Local names may also be affected by the **-p** option; however, the **-p**  
 116231 option shall not affect **#define** symbols generated by *yacc*.

116232 **-t** Modify conditional compilation directives to permit compilation of debugging  
 116233 code in the code file. Runtime debugging statements shall always be contained in  
 116234 the code file, but by default conditional compilation directives prevent their  
 116235 compilation.

116236 **-v** Write a file containing a description of the parser and a report of conflicts  
 116237 generated by ambiguities in the grammar.

**116238 OPERANDS**

116239 The following operand is required:

116240 *grammar* A pathname of a file containing instructions, hereafter called *grammar*, for which a  
 116241 parser is to be created. The format for the grammar is described in the EXTENDED  
 116242 DESCRIPTION section.

**116243 STDIN**

116244 Not used.

**116245 INPUT FILES**

116246 The file *grammar* shall be a text file formatted as specified in the EXTENDED DESCRIPTION  
 116247 section.

**116248 ENVIRONMENT VARIABLES**

116249 The following environment variables shall affect the execution of *yacc*:

116250 *LANG* Provide a default value for the internationalization variables that are unset or null.  
 116251 (See XBD Section 8.2 (on page 174) for the precedence of internationalization  
 116252 variables used to determine the values of locale categories.)

116253 *LC\_ALL* If set to a non-empty string value, override the values of all the other  
 116254 internationalization variables.

116255 *LC\_CTYPE* Determine the locale for the interpretation of sequences of bytes of text data as  
 116256 characters (for example, single-byte as opposed to multi-byte characters in  
 116257 arguments and input files).

116258 *LC\_MESSAGES*

116259 Determine the locale that should be used to affect the format and contents of  
 116260 diagnostic messages written to standard error.

116261 XSI *NLSPATH* Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

116262 The *LANG* and *LC\_\** variables affect the execution of the *yacc* utility as stated. The *main()*  
 116263 function defined in Yacc Library (on page 3454) shall call:

116264 `setlocale(LC_ALL, " ")`

116265 and thus the program generated by *yacc* shall also be affected by the contents of these variables  
 116266 at runtime.

**116267 ASYNCHRONOUS EVENTS**

116268 Default.

**116269 STDOUT**

116270 Not used.

**116271 STDERR**

116272 If shift/reduce or reduce/reduce conflicts are detected in *grammar*, *yacc* shall write a report of  
 116273 those conflicts to the standard error in an unspecified format.

116274 Standard error shall also be used for diagnostic messages.

**116275 OUTPUT FILES**

116276 The code file, the header file, and the description file shall be text files. All are described in the  
 116277 following sections.



## Code File

This file shall contain the C source code for the *yyparse()* function. It shall contain code for the various semantic actions with macro substitution performed on them as described in the EXTENDED DESCRIPTION section. It also shall contain a copy of the **#define** statements in the header file. If a **%union** declaration is used, the declaration for YYSTYPE shall also be included in this file.

## Header File

The header file shall contain **#define** statements that associate the token numbers with the token names. This allows source files other than the code file to access the token codes. If a **%union** declaration is used, the declaration for YYSTYPE and an *extern YYSTYPE yylval* declaration shall also be included in this file.

## Description File

The description file shall be a text file containing a description of the state machine corresponding to the parser, using an unspecified format. Limits for internal tables (see [Limits](#), on page 3455) shall also be reported, in an implementation-defined manner. (Some implementations may use dynamic allocation techniques and have no specific limit values to report.)

## EXTENDED DESCRIPTION

The *yacc* command accepts a language that is used to define a grammar for a target language to be parsed by the tables and code generated by *yacc*. The language accepted by *yacc* as a grammar for the target language is described below using the *yacc* input language itself.

The input *grammar* includes rules describing the input structure of the target language and code to be invoked when these rules are recognized to provide the associated semantic action. The code to be executed shall appear as bodies of text that are intended to be C-language code. These bodies of text shall not contain C-language trigraphs. The C-language inclusions are presumed to form a correct function when processed by *yacc* into its output files. The code included in this way shall be executed during the recognition of the target language.

Given a grammar, the *yacc* utility generates the files described in the OUTPUT FILES section. The code file can be compiled and linked using *c99*. If the declaration and programs sections of the grammar file did not include definitions of *main()*, *yylex()*, and *yyerror()*, the compiled output requires linking with externally supplied versions of those functions. Default versions of *main()* and *yyerror()* are supplied in the *yacc* library and can be linked in by using the *-ly* operand to *c99*. The *yacc* library interfaces need not support interfaces with other than the default **yy** symbol prefix. The application provides the lexical analyzer function, *yylex()*; the *lex* utility is specifically designed to generate such a routine.

## Input Language

The application shall ensure that every specification file consists of three sections in order: *declarations*, *grammar rules*, and *programs*, separated by double <percent-sign> characters ("%"). The declarations and programs sections can be empty. If the latter is empty, the preceding "%" mark separating it from the rules section can be omitted.

The input is free form text following the structure of the grammar defined below.

## Lexical Structure of the Grammar

The <blank>, <newline>, and <form-feed> character shall be ignored, except that the application shall ensure that they do not appear in names or multi-character reserved symbols. Comments shall be enclosed in `" / * . . . * / "`, and can appear wherever a name is valid.

Names are of arbitrary length, made up of letters, periods ( `' . '` ), underscores ( `' _ '` ), and non-initial digits. Uppercase and lowercase letters are distinct. Conforming applications shall not use names beginning in `yy` or `YY` since the *yacc* parser uses such names. Many of the names appear in the final output of *yacc*, and thus they should be chosen to conform with any additional rules created by the C compiler to be used. In particular they appear in `#define` statements.

A literal shall consist of a single character enclosed in single-quote characters. All of the escape sequences supported for character constants by the ISO C standard shall be supported by *yacc*.

The relationship with the lexical analyzer is discussed in detail below.

The application shall ensure that the NUL character is not used in grammar rules or literals.

## Declarations Section

The declarations section is used to define the symbols used to define the target language and their relationship with each other. In particular, much of the additional information required to resolve ambiguities in the context-free grammar for the target language is provided here.

Usually *yacc* assigns the relationship between the symbolic names it generates and their underlying numeric value. The declarations section makes it possible to control the assignment of these values.

It is also possible to keep semantic information associated with the tokens currently on the parse stack in a user-defined C-language **union**, if the members of the union are associated with the various names in the grammar. The declarations section provides for this as well.

The first group of declarators below all take a list of names as arguments. That list can optionally be preceded by the name of a C union member (called a *tag* below) appearing within `' < '` and `' > '`. (As an exception to the typographical conventions of the rest of this volume of POSIX.1-2008, in this case `<tag>` does not represent a metavariable, but the literal angle bracket characters surrounding a symbol.) The use of *tag* specifies that the tokens named on this line shall be of the same C type as the union member referenced by *tag*. This is discussed in more detail below.

For lists used to define tokens, the first appearance of a given token can be followed by a positive integer (as a string of decimal digits). If this is done, the underlying value assigned to it for lexical purposes shall be taken to be that number.

The following declares *name* to be a token:

```
%token [<tag>] name [number] [name [number]]...
```

If *tag* is present, the C type for all tokens on this line shall be declared to be the type referenced by *tag*. If a positive integer, *number*, follows a *name*, that value shall be assigned to the token.

The following declares *name* to be a token, and assigns precedence to it:

```
%left [<tag>] name [number] [name [number]]...
```

```
%right [<tag>] name [number] [name [number]]...
```

One or more lines, each beginning with one of these symbols, can appear in this section. All tokens on the same line have the same precedence level and associativity; the lines are in order

116362 of increasing precedence or binding strength. **%left** denotes that the operators on that line are  
 116363 left associative, and **%right** similarly denotes right associative operators. If *tag* is present, it shall  
 116364 declare a C type for *names* as described for **%token**.

116365 The following declares *name* to be a token, and indicates that this cannot be used associatively:

```
116366 %nonassoc [<tag>] name [number] [name [number]]...
```

116367 If the parser encounters associative use of this token it reports an error. If *tag* is present, it shall  
 116368 declare a C type for *names* as described for **%token**.

116369 The following declares that union member *names* are non-terminals, and thus it is required to  
 116370 have a *tag* field at its beginning:

```
116371 %type <tag> name...
```

116372 Because it deals with non-terminals only, assigning a token number or using a literal is also  
 116373 prohibited. If this construct is present, *yacc* shall perform type checking; if this construct is not  
 116374 present, the parse stack shall hold only the **int** type.

116375 Every name used in *grammar* not defined by a **%token**, **%left**, **%right**, or **%nonassoc** declaration  
 116376 is assumed to represent a non-terminal symbol. The *yacc* utility shall report an error for any non-  
 116377 terminal symbol that does not appear on the left side of at least one grammar rule.

116378 Once the type, precedence, or token number of a name is specified, it shall not be changed. If the  
 116379 first declaration of a token does not assign a token number, *yacc* shall assign a token number.  
 116380 Once this assignment is made, the token number shall not be changed by explicit assignment.

116381 The following declarators do not follow the previous pattern.

116382 The following declares the non-terminal *name* to be the *start symbol*, which represents the largest,  
 116383 most general structure described by the grammar rules:

```
116384 %start name
```

116385 By default, it is the left-hand side of the first grammar rule; this default can be overridden with  
 116386 this declaration.

116387 The following declares the *yacc* value stack to be a union of the various types of values desired.

```
116388 %union { body of union (in C) }
```

116389 The body of the union shall not contain unbalanced curly brace preprocessing tokens.

116390 By default, the values returned by actions (see below) and the lexical analyzer shall be of type  
 116391 **int**. The *yacc* utility keeps track of types, and it shall insert corresponding union member names  
 116392 in order to perform strict type checking of the resulting parser.

116393 Alternatively, given that at least one *<tag>* construct is used, the union can be declared in a  
 116394 header file (which shall be included in the declarations section by using a **#include** construct  
 116395 within **%{** and **%}**), and a **typedef** used to define the symbol **YYSTYPE** to represent this union.  
 116396 The effect of **%union** is to provide the declaration of **YYSTYPE** directly from the *yacc* input.

116397 C-language declarations and definitions can appear in the declarations section, enclosed by the  
 116398 following marks:

```
116399 %{ ... %}
```

116400 These statements shall be copied into the code file, and have global scope within it so that they  
 116401 can be used in the rules and program sections. The statements shall not contain **"%}"** outside a  
 116402 comment, string literal, or multi-character constant.

116403 The application shall ensure that the declarations section is terminated by the token `%%`.

#### 116404 Grammar Rules in yacc

116405 The rules section defines the context-free grammar to be accepted by the function *yacc* generates,  
116406 and associates with those rules C-language actions and additional precedence information. The  
116407 grammar is described below, and a formal definition follows.

116408 The rules section is comprised of one or more grammar rules. A grammar rule has the form:

116409 `A : BODY ;`

116410 The symbol **A** represents a non-terminal name, and **BODY** represents a sequence of zero or  
116411 more *names*, *literals*, and *semantic actions* that can then be followed by optional *precedence rules*.  
116412 Only the names and literals participate in the formation of the grammar; the semantic actions  
116413 and precedence rules are used in other ways. The `<colon>` and the `<semicolon>` are *yacc*  
116414 punctuation. If there are several successive grammar rules with the same left-hand side, the  
116415 `<vertical-line>` (`'|'`) can be used to avoid rewriting the left-hand side; in this case the  
116416 `<semicolon>` appears only after the last rule. The BODY part can be empty (or empty of names  
116417 and literals) to indicate that the non-terminal symbol matches the empty string.

116418 The *yacc* utility assigns a unique number to each rule. Rules using the vertical bar notation are  
116419 distinct rules. The number assigned to the rule appears in the description file.

116420 The elements comprising a BODY are:

116421 *name, literal* These form the rules of the grammar: *name* is either a *token* or a *non-terminal*; *literal*  
116422 stands for itself (less the lexically required quotation marks).

116423 *semantic action*

116424 With each grammar rule, the user can associate actions to be performed each time  
116425 the rule is recognized in the input process. (Note that the word “action” can also  
116426 refer to the actions of the parser—shift, reduce, and so on.)

116427 These actions can return values and can obtain the values returned by previous  
116428 actions. These values are kept in objects of type YYSTYPE (see `%union`). The  
116429 result value of the action shall be kept on the parse stack with the left-hand side of  
116430 the rule, to be accessed by other reductions as part of their right-hand side. By  
116431 using the `<tag>` information provided in the declarations section, the code  
116432 generated by *yacc* can be strictly type checked and contain arbitrary information. In  
116433 addition, the lexical analyzer can provide the same kinds of values for tokens, if  
116434 desired.

116435 An action is an arbitrary C statement and as such can do input or output, call  
116436 subprograms, and alter external variables. An action is one or more C statements  
116437 enclosed in curly braces `'{'` and `'}'`. The statements shall not contain  
116438 unbalanced curly brace preprocessing tokens.

116439 Certain pseudo-variables can be used in the action. These are macros for access to  
116440 data structures known internally to *yacc*.

116441 `$$` The value of the action can be set by assigning it to `$$`. If type  
116442 checking is enabled and the type of the value to be assigned cannot  
116443 be determined, a diagnostic message may be generated.

116444 `$number` This refers to the value returned by the component specified by the  
116445 token *number* in the right side of a rule, reading from left to right;  
116446 *number* can be zero or negative. If *number* is zero or negative, it refers

116447 to the data associated with the name on the parser's stack preceding  
 116448 the leftmost symbol of the current rule. (That is, "\$0" refers to the  
 116449 name immediately preceding the leftmost name in the current rule to  
 116450 be found on the parser's stack and "\$-1" refers to the symbol to *its*  
 116451 left.) If *number* refers to an element past the current point in the rule,  
 116452 or beyond the bottom of the stack, the result is undefined. If type  
 116453 checking is enabled and the type of the value to be assigned cannot  
 116454 be determined, a diagnostic message may be generated.

116455 \$<tag>number

116456 These correspond exactly to the corresponding symbols without the  
 116457 tag inclusion, but allow for strict type checking (and preclude  
 116458 unwanted type conversions). The effect is that the macro is expanded  
 116459 to use *tag* to select an element from the YYSTYPE union (using  
 116460 *dataname.tag*). This is particularly useful if *number* is not positive.

116461 \$<tag>\$

116462 This imposes on the reference the type of the union member  
 116463 referenced by *tag*. This construction is applicable when a reference to  
 116464 a left context value occurs in the grammar, and provides *yacc* with a  
 means for selecting a type.

116465 Actions can occur anywhere in a rule (not just at the end); an action can access  
 116466 values returned by actions to its left, and in turn the value it returns can be  
 116467 accessed by actions to its right. An action appearing in the middle of a rule shall be  
 116468 equivalent to replacing the action with a new non-terminal symbol and adding an  
 116469 empty rule with that non-terminal symbol on the left-hand side. The semantic  
 116470 action associated with the new rule shall be equivalent to the original action. The  
 116471 use of actions within rules might introduce conflicts that would not otherwise  
 116472 exist.

116473 By default, the value of a rule shall be the value of the first element in it. If the first  
 116474 element does not have a type (particularly in the case of a literal) and type  
 116475 checking is turned on by **%type**, an error message shall result.

116476 *precedence* The keyword **%prec** can be used to change the precedence level associated with a  
 116477 particular grammar rule. Examples of this are in cases where a unary and binary  
 116478 operator have the same symbolic representation, but need to be given different  
 116479 precedences, or where the handling of an ambiguous if-else construction is  
 116480 necessary. The reserved symbol **%prec** can appear immediately after the body of  
 116481 the grammar rule and can be followed by a token name or a literal. It shall cause  
 116482 the precedence of the grammar rule to become that of the following token name or  
 116483 literal. The action for the rule as a whole can follow **%prec**.

116484 If a program section follows, the application shall ensure that the grammar rules are terminated  
 116485 by **%%**.

## 116486 Programs Section

116487 The *programs* section can include the definition of the lexical analyzer *yylex()*, and any other  
 116488 functions; for example, those used in the actions specified in the grammar rules. It is unspecified  
 116489 whether the programs section precedes or follows the semantic actions in the output file;  
 116490 therefore, if the application contains any macro definitions and declarations intended to apply to  
 116491 the code in the semantic actions, it shall place them within "%{ . . . %}" in the declarations  
 116492 section.

116493 **Input Grammar**

116494 The following input to *yacc* yields a parser for the input to *yacc*. This formal syntax takes  
 116495 precedence over the preceding text syntax description.

116496 The lexical structure is defined less precisely; [Lexical Structure of the Grammar](#) (on page 3446)  
 116497 defines most terms. The correspondence between the previous terms and the tokens below is as  
 116498 follows.

116499 **IDENTIFIER** This corresponds to the concept of *name*, given previously. It also includes  
 116500 literals as defined previously.

116501 **C\_IDENTIFIER** This is a name, and additionally it is known to be followed by a <colon>. A  
 116502 literal cannot yield this token.

116503 **NUMBER** A string of digits (a non-negative decimal integer).

116504 **TYPE, LEFT, MARK, LCURL, RCURL**

116505 These correspond directly to **%type**, **%left**, **%%**, **%{**, and **%}**.

116506 **{...}** This indicates C-language source code, with the possible inclusion of '\$'  
 116507 macros as discussed previously.

```

116508 /* Grammar for the input to yacc. */
116509 /* Basic entries. */
116510 /* The following are recognized by the lexical analyzer. */

116511 %token    IDENTIFIER    /* Includes identifiers and literals */
116512 %token    C_IDENTIFIER  /* identifier (but not literal)
116513                        followed by a :. */
116514 %token    NUMBER        /* [0-9][0-9]* */

116515 /* Reserved words : %type=>TYPE %left=>LEFT, and so on */
116516 %token    LEFT RIGHT NONASSOC TOKEN PREC TYPE START UNION

116517 %token    MARK           /* The %% mark. */
116518 %token    LCURL          /* The %{ mark. */
116519 %token    RCURL          /* The %} mark. */

116520 /* 8-bit character literals stand for themselves; */
116521 /* tokens have to be defined for multi-byte characters. */

116522 %start    spec

116523 %%

116524 spec      : defs MARK rules tail
116525           ;
116526 tail      : MARK
116527           {
116528             /* In this action, set up the rest of the file. */
116529           }
116530           | /* Empty; the second MARK is optional. */
116531           ;
116532 defs      : /* Empty. */
116533           | defs def
116534           ;
116535 def       : START IDENTIFIER
116536           | UNION

```

```

116537         {
116538             /* Copy union definition to output. */
116539         }
116540         |
116541         {
116542             /* Copy C code to output file. */
116543         }
116544         |
116545         |
116546         ;
116547     rword : TOKEN
116548         |
116549         |
116550         |
116551         |
116552         ;
116553     tag : /* Empty: union tag ID optional. */
116554         | '<' IDENTIFIER '>'
116555         ;
116556     nlist : nmno
116557         |
116558         |
116559         ;
116560     nmno : IDENTIFIER /* Note: literal invalid with % type. */
116561         |
116562         |
116563         ;
116564     /* Rule section */
116565     rules : C_IDENTIFIER rbody prec
116566         |
116567         |
116568         ;
116569     rule : C_IDENTIFIER rbody prec
116570         |
116571         |
116572         ;
116573     rbody : /* empty */
116574         |
116575         |
116576         |
116577         ;
116578     act : '{'
116579         |
116580         |
116581         |
116582         |
116583         ;
116584     prec : /* Empty */
116585         |
116586         |
116587         |
116588         ;

```



## Conflicts

The parser produced for an input grammar may contain states in which conflicts occur. The conflicts occur because the grammar is not LALR(1). An ambiguous grammar always contains at least one LALR(1) conflict. The *yacc* utility shall resolve all conflicts, using either default rules or user-specified precedence rules.

Conflicts are either shift/reduce conflicts or reduce/reduce conflicts. A shift/reduce conflict is where, for a given state and lookahead symbol, both a shift action and a reduce action are possible. A reduce/reduce conflict is where, for a given state and lookahead symbol, reductions by two different rules are possible.

The rules below describe how to specify what actions to take when a conflict occurs. Not all shift/reduce conflicts can be successfully resolved this way because the conflict may be due to something other than ambiguity, so incautious use of these facilities can cause the language accepted by the parser to be much different from that which was intended. The description file shall contain sufficient information to understand the cause of the conflict. Where ambiguity is the reason either the default or explicit rules should be adequate to produce a working parser.

The declared precedences and associativities (see [Declarations Section](#), on page 3446) are used to resolve parsing conflicts as follows:

1. A precedence and associativity is associated with each grammar rule; it is the precedence and associativity of the last token or literal in the body of the rule. If the **%prec** keyword is used, it overrides this default. Some grammar rules might not have both precedence and associativity.
2. If there is a shift/reduce conflict, and both the grammar rule and the input symbol have precedence and associativity associated with them, then the conflict is resolved in favor of the action (shift or reduce) associated with the higher precedence. If the precedences are the same, then the associativity is used; left associative implies reduce, right associative implies shift, and non-associative implies an error in the string being parsed.
3. When there is a shift/reduce conflict that cannot be resolved by rule 2, the shift is done. Conflicts resolved this way are counted in the diagnostic output described in [Error Handling](#).
4. When there is a reduce/reduce conflict, a reduction is done by the grammar rule that occurs earlier in the input sequence. Conflicts resolved this way are counted in the diagnostic output described in [Error Handling](#).

Conflicts resolved by precedence or associativity shall not be counted in the shift/reduce and reduce/reduce conflicts reported by *yacc* on either standard error or in the description file.

## Error Handling

The token **error** shall be reserved for error handling. The name **error** can be used in grammar rules. It indicates places where the parser can recover from a syntax error. The default value of **error** shall be 256. Its value can be changed using a **%token** declaration. The lexical analyzer should not return the value of **error**.

The parser shall detect a syntax error when it is in a state where the action associated with the lookahead symbol is **error**. A semantic action can cause the parser to initiate error handling by executing the macro YYERROR. When YYERROR is executed, the semantic action passes control back to the parser. YYERROR cannot be used outside of semantic actions.

When the parser detects a syntax error, it normally calls *yyerror()* with the character string "syntax error" as its argument. The call shall not be made if the parser is still recovering



from a previous error when the error is detected. The parser is considered to be recovering from a previous error until the parser has shifted over at least three normal input symbols since the last error was detected or a semantic action has executed the macro *yterror*. The parser shall not call *yterror*() when YYERROR is executed.

The macro function YYRECOVERING shall return 1 if a syntax error has been detected and the parser has not yet fully recovered from it. Otherwise, zero shall be returned.

When a syntax error is detected by the parser, the parser shall check if a previous syntax error has been detected. If a previous error was detected, and if no normal input symbols have been shifted since the preceding error was detected, the parser checks if the lookahead symbol is an endmarker (see [Interface to the Lexical Analyzer](#)). If it is, the parser shall return with a non-zero value. Otherwise, the lookahead symbol shall be discarded and normal parsing shall resume.

When YYERROR is executed or when the parser detects a syntax error and no previous error has been detected, or at least one normal input symbol has been shifted since the previous error was detected, the parser shall pop back one state at a time until the parse stack is empty or the current state allows a shift over **error**. If the parser empties the parse stack, it shall return with a non-zero value. Otherwise, it shall shift over **error** and then resume normal parsing. If the parser reads a lookahead symbol before the error was detected, that symbol shall still be the lookahead symbol when parsing is resumed.

The macro *yterror* in a semantic action shall cause the parser to act as if it has fully recovered from any previous errors. The macro *yyclearin* shall cause the parser to discard the current lookahead token. If the current lookahead token has not yet been read, *yyclearin* shall have no effect.

The macro YYACCEPT shall cause the parser to return with the value zero. The macro YYABORT shall cause the parser to return with a non-zero value.

### Interface to the Lexical Analyzer

The *yylex*() function is an integer-valued function that returns a *token number* representing the kind of token read. If there is a value associated with the token returned by *yylex*() (see the discussion of *tag* above), it shall be assigned to the external variable *yylval*.

If the parser and *yylex*() do not agree on these token numbers, reliable communication between them cannot occur. For (single-byte character) literals, the token is simply the numeric value of the character in the current character set. The numbers for other tokens can either be chosen by *yacc*, or chosen by the user. In either case, the **#define** construct of C is used to allow *yylex*() to return these numbers symbolically. The **#define** statements are put into the code file, and the header file if that file is requested. The set of characters permitted by *yacc* in an identifier is larger than that permitted by C. Token names found to contain such characters shall not be included in the **#define** declarations.

If the token numbers are chosen by *yacc*, the tokens other than literals shall be assigned numbers greater than 256, although no order is implied. A token can be explicitly assigned a number by following its first appearance in the declarations section with a number. Names and literals not defined this way retain their default definition. All token numbers assigned by *yacc* shall be unique and distinct from the token numbers used for literals and user-assigned tokens. If duplicate token numbers cause conflicts in parser generation, *yacc* shall report an error; otherwise, it is unspecified whether the token assignment is accepted or an error is reported.

The end of the input is marked by a special token called the *endmarker*, which has a token number that is zero or negative. (These values are invalid for any other token.) All lexical analyzers shall return zero or negative as a token number upon reaching the end of their input.

116675 If the tokens up to, but excluding, the endmarker form a structure that matches the start symbol,  
116676 the parser shall accept the input. If the endmarker is seen in any other context, it shall be  
116677 considered an error.

## 116678 **Completing the Program**

116679 In addition to *yyparse()* and *yylex()*, the functions *yyerror()* and *main()* are required to make a  
116680 complete program. The application can supply *main()* and *yyerror()*, or those routines can be  
116681 obtained from the *yacc* library.

## 116682 **Yacc Library**

116683 The following functions shall appear only in the *yacc* library accessible through the **-l y** operand  
116684 to *c99*; they can therefore be redefined by a conforming application:

### 116685 **int main(void)**

116686 This function shall call *yyparse()* and exit with an unspecified value. Other actions within  
116687 this function are unspecified.

### 116688 **int yyerror(const char \*s)**

116689 This function shall write the NUL-terminated argument to standard error, followed by a  
116690 <newline>.

116691 The order of the **-l y** and **-l l** operands given to *c99* is significant; the application shall either  
116692 provide its own *main()* function or ensure that **-l y** precedes **-l l**.

## 116693 **Debugging the Parser**

116694 The parser generated by *yacc* shall have diagnostic facilities in it that can be optionally enabled  
116695 at either compile time or at runtime (if enabled at compile time). The compilation of the runtime  
116696 debugging code is under the control of *YYDEBUG*, a preprocessor symbol. If *YYDEBUG* has a  
116697 non-zero value, the debugging code shall be included. If its value is zero, the code shall not be  
116698 included.

116699 In parsers where the debugging code has been included, the external **int yydebug** can be used to  
116700 turn debugging on (with a non-zero value) and off (zero value) at runtime. The initial value of  
116701 *yydebug* shall be zero.

116702 When **-t** is specified, the code file shall be built such that, if *YYDEBUG* is not already defined at  
116703 compilation time (using the *c99* **-D YYDEBUG** option, for example), *YYDEBUG* shall be set  
116704 explicitly to 1. When **-t** is not specified, the code file shall be built such that, if *YYDEBUG* is not  
116705 already defined, it shall be set explicitly to zero.

116706 The format of the debugging output is unspecified but includes at least enough information to  
116707 determine the shift and reduce actions, and the input symbols. It also provides information  
116708 about error recovery.

## 116709 **Algorithms**

116710 The parser constructed by *yacc* implements an LALR(1) parsing algorithm as documented in the  
116711 literature. It is unspecified whether the parser is table-driven or direct-coded.

116712 A parser generated by *yacc* shall never request an input symbol from *yylex()* while in a state  
116713 where the only actions other than the error action are reductions by a single rule.

116714 The literature of parsing theory defines these concepts.

## Limits

The *yacc* utility may have several internal tables. The minimum maximums for these tables are shown in the following table. The exact meaning of these values is implementation-defined. The implementation shall define the relationship between these values and between them and any error messages that the implementation may generate should it run out of space for any internal structure. An implementation may combine groups of these resources into a single pool as long as the total available to the user does not fall below the sum of the sizes specified by this section.

**Table 4-23** Internal Limits in *yacc*

Limit	Minimum Maximum	Description
{NTERMS}	126	Number of tokens.
{NNONTERM}	200	Number of non-terminals.
{NPROD}	300	Number of rules.
{NSTATES}	600	Number of states.
{MEMSIZE}	5 200	Length of rules. The total length, in names (tokens and non-terminals), of all the rules of the grammar. The left-hand side is counted for each rule, even if it is not explicitly repeated, as specified in <a href="#">Grammar Rules in yacc</a> (on page 3448).
{ACTSIZE}	4 000	Number of actions. “Actions” here (and in the description file) refer to parser actions (shift, reduce, and so on) not to semantic actions defined in <a href="#">Grammar Rules in yacc</a> (on page 3448).

## EXIT STATUS

The following exit values shall be returned:

0 Successful completion.

>0 An error occurred.

## CONSEQUENCES OF ERRORS

If any errors are encountered, the run is aborted and *yacc* exits with a non-zero status. Partial code files and header files may be produced. The summary information in the description file shall always be produced if the *-v* flag is present.

## APPLICATION USAGE

Historical implementations experience name conflicts on the names **yacc.tmp**, **yacc.acts**, **yacc.debug**, **y.tab.c**, **y.tab.h**, and **y.output** if more than one copy of *yacc* is running in a single directory at one time. The *-b* option was added to overcome this problem. The related problem of allowing multiple *yacc* parsers to be placed in the same file was addressed by adding a *-p* option to override the previously hard-coded **yy** variable prefix.

The description of the *-p* option specifies the minimal set of function and variable names that cause conflict when multiple parsers are linked together. **YYSTYPE** does not need to be changed. Instead, the programmer can use *-b* to give the header files for different parsers different names, and then the file with the *yylex()* for a given parser can include the header for that parser. Names such as *yyclearerr* do not need to be changed because they are used only in the actions; they do not have linkage. It is possible that an implementation has other names, either internal ones for implementing things such as *yyclearerr*, or providing non-standard features that it wants

to change with `-p`.

Unary operators that are the same token as a binary operator in general need their precedence adjusted. This is handled by the `%prec` advisory symbol associated with the particular grammar rule defining that unary operator. (See [Grammar Rules in yacc](#) (on page 3448).) Applications are not required to use this operator for unary operators, but the grammars that do not require it are rare.

## EXAMPLES

Access to the *yacc* library is obtained with library search operands to *c99*. To use the *yacc* library *main()*:

```
c99 y.tab.c -l y
```

Both the *lex* library and the *yacc* library contain *main()*. To access the *yacc main()*:

```
c99 y.tab.c lex.yy.c -l y -l l
```

This ensures that the *yacc* library is searched first, so that its *main()* is used.

The historical *yacc* libraries have contained two simple functions that are normally coded by the application programmer. These functions are similar to the following code:

```
#include <locale.h>
int main(void)
{
    extern int yyparse();

    setlocale(LC_ALL, "");

    /* If the following parser is one created by lex, the
       application must be careful to ensure that LC_CTYPE
       and LC_COLLATE are set to the POSIX locale. */
    (void) yyparse();
    return (0);
}

#include <stdio.h>

int yyerror(const char *msg)
{
    (void) fprintf(stderr, "%s\n", msg);
    return (0);
}
```

## RATIONALE

The references in **Referenced Documents** may be helpful in constructing the parser generator. The referenced DeRemer and Pennello article (along with the works it references) describes a technique to generate parsers that conform to this volume of POSIX.1-2008. Work in this area continues to be done, so implementors should consult current literature before doing any new implementations. The original Knuth article is the theoretical basis for this kind of parser, but the tables it generates are impractically large for reasonable grammars and should not be used. The “equivalent to” wording is intentional to assure that the best tables that are LALR(1) can be generated.

There has been confusion between the class of grammars, the algorithms needed to generate parsers, and the algorithms needed to parse the languages. They are all reasonably orthogonal. In particular, a parser generator that accepts the full range of LR(1) grammars need not generate a table any more complex than one that accepts SLR(1) (a relatively weak class of LR grammars)

for a grammar that happens to be SLR(1). Such an implementation need not recognize the case, either; table compression can yield the SLR(1) table (or one even smaller than that) without recognizing that the grammar is SLR(1). The speed of an LR(1) parser for any class is dependent more upon the table representation and compression (or the code generation if a direct parser is generated) than upon the class of grammar that the table generator handles.

The speed of the parser generator is somewhat dependent upon the class of grammar it handles. However, the original Knuth article algorithms for constructing LR parsers were judged by its author to be impractically slow at that time. Although full LR is more complex than LALR(1), as computer speeds and algorithms improve, the difference (in terms of acceptable wall-clock execution time) is becoming less significant.

Potential authors are cautioned that the referenced DeRemer and Pennello article previously cited identifies a bug (an over-simplification of the computation of LALR(1) lookahead sets) in some of the LALR(1) algorithm statements that preceded it to publication. They should take the time to seek out that paper, as well as current relevant work, particularly Aho's.

The **-b** option was added to provide a portable method for permitting *yacc* to work on multiple separate parsers in the same directory. If a directory contains more than one *yacc* grammar, and both grammars are constructed at the same time (by, for example, a parallel *make* program), conflict results. While the solution is not historical practice, it corrects a known deficiency in historical implementations. Corresponding changes were made to all sections that referenced the filenames **y.tab.c** (now "the code file"), **y.tab.h** (now "the header file"), and **y.output** (now "the description file").

The grammar for *yacc* input is based on System V documentation. The textual description shows there that the **' ; '** is required at the end of the rule. The grammar and the implementation do not require this. (The use of **C\_IDENTIFIER** causes a reduce to occur in the right place.)

Also, in that implementation, the constructs such as **%token** can be terminated by a **<semicolon>**, but this is not permitted by the grammar. The keywords such as **%token** can also appear in uppercase, which is again not discussed. In most places where **' % '** is used, **<backslash>** can be substituted, and there are alternate spellings for some of the symbols (for example, **%LEFT** can be **"%<"** or even **"\<"**).

Historically, **<tag>** can contain any characters except **' > '**, including white space, in the implementation. However, since the *tag* must reference an ISO C standard union member, in practice conforming implementations need to support only the set of characters for ISO C standard identifiers in this context.

Some historical implementations are known to accept actions that are terminated by a period. Historical implementations often allow **' \$ '** in names. A conforming implementation does not need to support either of these behaviors.

Deciding when to use **%prec** illustrates the difficulty in specifying the behavior of *yacc*. There may be situations in which the *grammar* is not, strictly speaking, in error, and yet *yacc* cannot interpret it unambiguously. The resolution of ambiguities in the grammar can in many instances be resolved by providing additional information, such as using **%type** or **%union** declarations. It is often easier and it usually yields a smaller parser to take this alternative when it is appropriate.

The size and execution time of a program produced without the runtime debugging code is usually smaller and slightly faster in historical implementations.

Statistics messages from several historical implementations include the following types of information:

116852 *n*/512 terminals, *n*/300 non-terminals  
116853 *n*/600 grammar rules, *n*/1500 states  
116854 *n* shift/reduce, *n* reduce/reduce conflicts reported  
116855 *n*/350 working sets used  
116856 Memory: states, etc. *n*/15 000, parser *n*/15 000  
116857 *n*/600 distinct lookahead sets  
116858 *n* extra closures  
116859 *n* shift entries, *n* exceptions  
116860 *n* goto entries  
116861 *n* entries saved by goto default  
116862 Optimizer space used: input *n*/15 000, output *n*/15 000  
116863 *n* table entries, *n* zero  
116864 Maximum spread: *n*, Maximum offset: *n*

116865 The report of internal tables in the description file is left implementation-defined because all  
116866 aspects of these limits are also implementation-defined. Some implementations may use  
116867 dynamic allocation techniques and have no specific limit values to report.

116868 The format of the **y.output** file is not given because specification of the format was not seen to  
116869 enhance applications portability. The listing is primarily intended to help human users  
116870 understand and debug the parser; use of **y.output** by a conforming application script would be  
116871 unusual. Furthermore, implementations have not produced consistent output and no popular  
116872 format was apparent. The format selected by the implementation should be human-readable, in  
116873 addition to the requirement that it be a text file.

116874 Standard error reports are not specifically described because they are seldom of use to  
116875 conforming applications and there was no reason to restrict implementations.

116876 Some implementations recognize "`={"`" as equivalent to '`{`' because it appears in historical  
116877 documentation. This construction was recognized and documented as obsolete as long ago as  
116878 1978, in the referenced *Yacc: Yet Another Compiler-Compiler*. This volume of POSIX.1-2008 chose to  
116879 leave it as obsolete and omit it.

116880 Multi-byte characters should be recognized by the lexical analyzer and returned as tokens. They  
116881 should not be returned as multi-byte character literals. The token **error** that is used for error  
116882 recovery is normally assigned the value 256 in the historical implementation. Thus, the token  
116883 value 256, which is used in many multi-byte character sets, is not available for use as the value  
116884 of a user-defined token.

#### 116885 FUTURE DIRECTIONS

116886 None.

#### 116887 SEE ALSO

116888 *c99*, *lex*

116889 XBD [Chapter 8](#) (on page 173), [Section 12.2](#) (on page 216)

#### 116890 CHANGE HISTORY

116891 First released in Issue 2.

#### 116892 Issue 5

116893 The FUTURE DIRECTIONS section is added.

#### 116894 Issue 6

116895 This utility is marked as part of the C-Language Development Utilities option.

116896 Minor changes have been added to align with the IEEE P1003.2b draft standard.



- 116897 The normative text is reworded to avoid use of the term “must” for application requirements.
- 116898 IEEE PASC Interpretation 1003.2 #177 is applied, changing the comment on **RCURL** from the }%  
116899 token to the %}.
- 116900 **Issue 7**
- 116901 Austin Group Interpretation 1003.1-2001 #190 is applied, clarifying the requirements for  
116902 generated code to conform to the ISO C standard.
- 116903 Austin Group Interpretation 1003.1-2001 #191 is applied, clarifying the handling of C-language  
116904 trigraphs and curly brace preprocessing tokens.
- 116905 SD5-XCU-ERN-6 is applied, clarifying that Guideline 9 of the Utility Syntax Guidelines does not  
116906 apply.
- 116907 SD5-XCU-ERN-97 is applied, updating the SYNOPSIS.

116908 **NAME**

116909           zcat — expand and concatenate data

116910 **SYNOPSIS**116911 XSI        zcat [*file...*]116912 **DESCRIPTION**

116913       The *zcat* utility shall write to standard output the uncompressed form of files that have been  
 116914       compressed using the *compress* utility. It is the equivalent of *uncompress -c*. Input files are not  
 116915       affected.

116916 **OPTIONS**

116917       None.

116918 **OPERANDS**

116919       The following operand shall be supported:

116920       *file*       The pathname of a file previously processed by the *compress* utility. If *file* already  
 116921       has the *.Z* suffix specified, it is used as submitted. Otherwise, the *.Z* suffix is  
 116922       appended to the filename prior to processing.

116923 **STDIN**116924       The standard input shall be used only if no *file* operands are specified, or if a *file* operand is *'-'*.116925 **INPUT FILES**116926       Input files shall be compressed files that are in the format produced by the *compress* utility.116927 **ENVIRONMENT VARIABLES**116928       The following environment variables shall affect the execution of *zcat*:

116929       *LANG*       Provide a default value for the internationalization variables that are unset or null.  
 116930       (See XBD [Section 8.2](#) (on page 174) for the precedence of internationalization  
 116931       variables used to determine the values of locale categories.)

116932       *LC\_ALL*     If set to a non-empty string value, override the values of all the other  
 116933       internationalization variables.

116934       *LC\_CTYPE*   Determine the locale for the interpretation of sequences of bytes of text data as  
 116935       characters (for example, single-byte as opposed to multi-byte characters in  
 116936       arguments).

116937       *LC\_MESSAGES*

116938       Determine the locale that should be used to affect the format and contents of  
 116939       diagnostic messages written to standard error.

116940       *NLSPATH*   Determine the location of message catalogs for the processing of *LC\_MESSAGES*.

116941 **ASYNCHRONOUS EVENTS**

116942       Default.

116943 **STDOUT**

116944       The compressed files given as input shall be written on standard output in their uncompressed  
 116945       form.

116946 **STDERR**

116947       The standard error shall be used only for diagnostic messages.



116948 **OUTPUT FILES**

116949       None.

116950 **EXTENDED DESCRIPTION**

116951       None.

116952 **EXIT STATUS**

116953       The following exit values shall be returned:

116954       0   Successful completion.

116955       &gt;0   An error occurred.

116956 **CONSEQUENCES OF ERRORS**

116957       Default.

116958 **APPLICATION USAGE**

116959       None.

116960 **EXAMPLES**

116961       None.

116962 **RATIONALE**

116963       None.

116964 **FUTURE DIRECTIONS**

116965       None.

116966 **SEE ALSO**116967       *compress, uncompress*116968       XBD [Chapter 8](#) (on page 173)116969 **CHANGE HISTORY**

116970       First released in Issue 4.



116971



116972

**Vol. 4:**

116973

**Rationale (Informative), Issue 7**

116974

*The Open Group*

116975

*The Institute of Electrical and Electronics Engineers, Inc.*



116976

# *Rationale (Informative)*

116977

## **Part A:**

116978

## **Base Definitions**

116979

*The Open Group*

116980

*The Institute of Electrical and Electronics Engineers, Inc.*



116981

Appendix A

116982

# Rationale for Base Definitions

## A.1 Introduction

### A.1.1 Scope

POSIX.1-2008 is one of a family of standards known as POSIX. The family of standards extends to many topics; POSIX.1 consists of both operating system interfaces and shell and utilities. POSIX.1-2008 is technically identical to The Open Group Base Specifications, Issue 7.

#### Scope of POSIX.1-2008

The (paraphrased) goals of this development were to revise the single document that is ISO/IEC 9945:2003 Parts 1 through 4, IEEE Std 1003.1, 2004 Edition, and the appropriate parts of The Open Group Single UNIX Specification, Version 3. This work has been undertaken by the Austin Group, a joint working group of IEEE, The Open Group, and ISO/IEC JTC 1/SC 22.

The following are the base documents in this version:

- IEEE Std 1003.1, 2004 Edition
- ISO/IEC 9899:1999, Programming Languages — C, including ISO/IEC 9899:1999/Cor.1:2001(E), ISO/IEC 9899:1999/Cor.2:2004(E), and ISO/IEC 9899:1999/Cor.3
- The Open Group Extended API Sets, Parts 1 through 4

This version has addressed the following areas:

- Issues raised by Austin Group defect reports, IEEE Interpretations against IEEE Std 1003.1, and ISO/IEC defect reports against ISO/IEC 9945

The repository of interpretations can be accessed at [www.opengroup.org/austin/interps](http://www.opengroup.org/austin/interps).

- Issues raised in corrigenda for The Open Group Technical Standards and working group resolutions from The Open Group
- Issues arising from ISO TR 24715:2006, Conflicts between POSIX and the LSB

This is a Type 3 informative technical report highlighting differences between the LSB 3.1 and the 2004 Edition of this standard.

- Changes to make the text self-consistent with the additional material merged

The new material merged has come from the The Open Group Extended API Sets, Parts 1 through 4. A list of the new interfaces is included in [Section B.1.1](#) (on page 3551).

- Features, marked legacy or obsolescent in the base documents, have been considered for removal in this version

See [Section B.1.1](#) (on page 3551) and [Section C.1.1](#) (on page 3695).

- 117014 • A review and reorganization of the options within the standard
- 117015 This has included marking the following options obsolescent:
- 117016 — Batch Environment Services and Utilities
- 117017 — Tracing
- 117018 — XSI STREAMS
- 117019 The UUCP Utilities option is a new option for this version.
- 117020 Functionality from the following former options is now mandatory in this version:
- 117021 AIO    \_POSIX\_ASYNCHRONOUS\_IO (Asynchronous Input and Output)
- 117022 BAR    \_POSIX\_BARRIERS (Barriers)
- 117023 CS     \_POSIX\_CLOCK\_SELECTION (Clock Selection)
- 117024 MF     \_POSIX\_MAPPED\_FILES (Memory Mapped Files)
- 117025 MPR    \_POSIX\_MEMORY\_PROTECTION (Memory Protection)
- 117026 RTS    \_POSIX\_REALTIME\_SIGNALS (Realtime Signals Extension)
- 117027 RWL    \_POSIX\_READER\_WRITER\_LOCKS (Read-Write Locks)
- 117028 SEM    \_POSIX\_SEMAPHORES (Semaphores)
- 117029 SPI    \_POSIX\_SPIN\_LOCKS (Spin Locks)
- 117030 THR    \_POSIX\_THREADS (Threads)
- 117031 TMO    \_POSIX\_TIMEOUTS (Timeouts)
- 117032 TMR    \_POSIX\_TIMERS (Timers)
- 117033 TSF    \_POSIX\_THREAD\_SAFE\_FUNCTIONS (Thread-Safe Functions)
- 117034 • Alignment with the ISO/IEC 9899:1999 standard, including ISO/IEC
- 117035 9899:1999/Cor.2:2004(E)
- 117036 • A review of the use of fixed path filenames within the standard
- 117037 For example, the *at*, *batch*, and *crontab* utilities previously had a requirement for use of the
- 117038 directory **/usr/lib/cron**.
- 117039 The following were requirements on POSIX.1-2008:
- 117040 • Backward-compatibility
- 117041 For interfaces carried forward, it was agreed that there should be no breakage of
- 117042 functionality in the existing base documents. All strictly conforming applications will be
- 117043 conforming but not necessarily strictly conforming to the revised standard. The goal is for
- 117044 system implementations to be able to support the existing and revised standards
- 117045 simultaneously.
- 117046 • Architecture and *n*-bit-neutral
- 117047 The common standard should not make any implicit assumptions about the system
- 117048 architecture or size of data types; for example, previously some 32-bit implicit assumptions
- 117049 had crept into the standards.
- 117050 • Extensibility
- 117051 It should be possible to extend the common standard without breaking backwards-
- 117052 compatibility; for example, the name space should be reserved and structured to avoid
- 117053 duplication of names between the standard and extensions to it.



**POSIX.1 and the ISO C Standard**

The standard developers believed it essential for a programmer to have a single complete reference place, but recognized that deference to the formal standard has to be addressed for the duplicate interface definitions between the ISO C standard and POSIX.1-2008.

Where an interface has a version in the ISO C standard, the DESCRIPTION section describes the relationship to the ISO C standard and markings are included as appropriate to show where the ISO C standard has been extended in the text.

A block of text is included at the start of each affected reference page stating whether the page is aligned with the ISO C standard or extended. Each page has been parsed for additions beyond the ISO C standard (that is, including both POSIX and UNIX extensions), and these extensions are marked as CX extensions (for C extensions).

**FIPS Requirements**

The Federal Information Processing Standards (FIPS) are a series of US government procurement standards managed and maintained on behalf of the US Department of Commerce by the National Institute of Standards and Technology (NIST).

The following restrictions were integrated into IEEE Std 1003.1-2001. They originally came from FIPS 151-2 which was withdrawn by NIST on February 25 2000.

- The implementation supports `_POSIX_CHOWN_RESTRICTED`.
- The limit `{NGROUPS_MAX}` is greater than or equal to 8.
- The implementation supports the setting of the group ID of a file (when it is created) to that of the parent directory.
- The implementation supports `_POSIX_SAVED_IDS`.
- The implementation supports `_POSIX_VDISABLE`.
- The implementation supports `_POSIX_JOB_CONTROL`.
- The implementation supports `_POSIX_NO_TRUNC`.
- The `read()` function returns the number of bytes read when interrupted by a signal and does not return `-1`.
- The `write()` function returns the number of bytes written when interrupted by a signal and does not return `-1`.
- In the environment for the login shell, the environment variables `LOGNAME` and `HOME` are defined and have the properties described in POSIX.1-2008.
- The value of `{CHILD_MAX}` is greater than or equal to 25.
- The value of `{OPEN_MAX}` is greater than or equal to 20.
- The implementation supports the functionality associated with the symbols `CS7`, `CS8`, `CSTOPB`, `PARODD`, and `PARENB` defined in **<termios.h>**.

**117089 A.1.2 Conformance**

117090 See [Section A.2](#) (on page 3473).

**117091 A.1.3 Normative References**

117092 There is no additional rationale provided for this section.

**117093 A.1.4 Change History**

117094 For Issue 7 onwards, in references to Technical Corrigenda, the original Austin Group defect  
117095 report numbers that gave rise to the change are included in square brackets after the change  
117096 number from the Technical Corrigendum. For more information on Austin Group defect reports  
117097 see [www.opengroup.org/austin/defectform.html](http://www.opengroup.org/austin/defectform.html).

**117098 A.1.5 Terminology**

117099 The meanings specified in POSIX.1-2008 for the words *shall*, *should*, and *may* are mandated by  
117100 ISO/IEC directives.

117101 In the Rationale (Informative) volume of POSIX.1-2008, the words *shall*, *should*, and *may* are  
117102 sometimes used to illustrate similar usages in POSIX.1-2008. However, the rationale itself does  
117103 not specify anything regarding implementations or applications.

**117104 conformance document**

117105 As a practical matter, the conformance document is effectively part of the system  
117106 documentation. Conformance documents are distinguished by POSIX.1-2008 so that they  
117107 can be referred to distinctly.

**117108 implementation-defined**

117109 This definition is analogous to that of the ISO C standard and, together with “undefined”  
117110 and “unspecified”, provides a range of specification of freedom allowed to the interface  
117111 implementor.

**117112 may**

117113 The use of *may* has been limited as much as possible, due both to confusion stemming from  
117114 its ordinary English meaning and to objections regarding the desirability of having as few  
117115 options as possible and those as clearly specified as possible.

117116 The usage of *can* and *may* were selected to contrast optional application behavior (can)  
117117 against optional implementation behavior (may).

**117118 shall**

117119 Declarative sentences are sometimes used in POSIX.1-2008 as if they included the word  
117120 *shall*, and facilities thus specified are no less required. For example, the two statements:

- 117121 1. The *foo()* function shall return zero.
- 117122 2. The *foo()* function returns zero.

117123 are meant to be exactly equivalent.

**117124 should**

117125 In POSIX.1-2008, the word *should* does not usually apply to the implementation, but rather  
117126 to the application. Thus, the important words regarding implementations are *shall*, which  
117127 indicates requirements, and *may*, which indicates options.

**obsolescent**

The term “obsolescent” means “do not use this feature in new applications”. A feature noted as obsolescent is supported by all implementations, but may be removed in a future version; new applications should not use these features. The obsolescence concept is not an ideal solution, but was used as a method of increasing consensus: many more objections would be heard from the user community if some of these historical features were suddenly removed without the grace period obsolescence implies. The phrase “may be removed in a future version” implies that the result of that consideration might in fact keep those features indefinitely if the predominance of applications do not migrate away from them quickly.

**legacy**

The term “legacy” was included in earlier versions of this standard but is no longer used in the current version.

**system documentation**

The system documentation should normally describe the whole of the implementation, including any extensions provided by the implementation. Such documents normally contain information at least as detailed as the specifications in POSIX.1-2008. Few requirements are made on the system documentation, but the term is needed to avoid a dangling pointer where the conformance document is permitted to point to the system documentation.

**undefined**

See *implementation-defined*.

**unspecified**

See *implementation-defined*.

The definitions for “unspecified” and “undefined” appear nearly identical at first examination, but are not. The term “unspecified” means that a conforming application may deal with the unspecified behavior, and it should not care what the outcome is. The term “undefined” says that a conforming application should not do it because no definition is provided for what it does (and implicitly it would care what the outcome was if it tried it). It is important to remember, however, that if the syntax permits the statement at all, it must have some outcome in a real implementation.

Thus, the terms “undefined” and “unspecified” apply to the way the application should think about the feature. In terms of the implementation, it is always “defined”—there is always some result, even if it is an error. The implementation is free to choose the behavior it prefers.

This also implies that an implementation, or another standard, could specify or define the result in a useful fashion. The terms apply to POSIX.1-2008 specifically.

The term “implementation-defined” implies requirements for documentation that are not required for “undefined” (or “unspecified”). Where there is no need for a conforming program to know the definition, the term “undefined” is used, even though “implementation-defined” could also have been used in this context. There could be a fourth term, specifying “this standard does not say what this does; it is acceptable to define it in an implementation, but it does not need to be documented”, and undefined would then be used very rarely for the few things for which any definition is not useful. In particular, implementation-defined is used where it is believed that certain classes of application will need to know such details to determine whether the application can be successfully ported to the implementation. Such applications are not always strictly portable, but nevertheless are common and useful; often the requirements met by the application cannot be met without dealing with the issues implied by “implementation-defined”. In some places the text refers to facilities supplied by the implementation that are

117177 outside the standard as implementation-supplied or implementation-provided. This is not  
 117178 intended to imply a requirement for documentation. If it were, the term “implementation-  
 117179 defined” would have been used.

117180 In many places POSIX.1-2008 is silent about the behavior of some possible construct. For  
 117181 example, a variable may be defined for a specified range of values and behaviors are  
 117182 described for those values; nothing is said about what happens if the variable has any other  
 117183 value. That kind of silence can imply an error in the standard, but it may also imply that the  
 117184 standard was intentionally silent and that any behavior is permitted. There is a natural  
 117185 tendency to infer that if the standard is silent, a behavior is prohibited. That is not the intent.  
 117186 Silence is intended to be equivalent to the term “unspecified”. -

117187 Three terms used within POSIX.1-2008 overlap in meaning: “macro”, “symbolic name”, and  
 117188 “symbolic constant”.

#### 117189 **macro**

117190 This usually describes a C preprocessor symbol, the result of the **#define** operator, with or  
 117191 without an argument. It may also be used to describe similar mechanisms in editors and  
 117192 text processors.

#### 117193 **symbolic name**

117194 In earlier versions of this standard this was also sometimes used to refer to a C preprocessor  
 117195 symbol (without arguments), but the intention is for all such uses to have been removed. It  
 117196 is now mainly used to refer to the names for characters in character sets, but is sometimes  
 117197 used to refer to host names and even filenames.

#### 117198 **symbolic constant**

117199 This also refers to a C preprocessor symbol, with specific associated requirements. See the  
 117200 definition in [Section 3.380](#) (on page 95).

### 117201 **A.1.6 Definitions and Concepts**

117202 There is no additional rationale provided for this section.

### 117203 **A.1.7 Portability**

117204 To aid the identification of options within POSIX.1-2008, a notation consisting of margin codes  
 117205 and shading is used. This is based on the notation used in earlier versions of The Open Group  
 117206 Base specifications.

117207 The benefit of this approach is a reduction in the number of *if* statements within the running  
 117208 text, that makes the text easier to read, and also an identification to the programmer that they  
 117209 need to ensure that their target platforms support the underlying options. For example, if  
 117210 functionality is marked with RPP in the margin, it will be available on all systems supporting  
 117211 the Robust Mutex Priority Protection option, but may not be available on some others.

#### 117212 *A.1.7.1 Codes*

117213 This section includes codes for options defined in XBD [Section 2.1.6](#) (on page 26), and the  
 117214 following additional codes for other purposes:

117215 CX This margin code is used to denote extensions beyond the ISO C standard. For  
 117216 interfaces that are duplicated between POSIX.1-2008 and the ISO C standard, a CX  
 117217 introduction block describes the nature of the duplication, with any extensions

- 117218 appropriately CX marked and shaded.
- 117219 Where an interface is added to an ISO C standard header, within the header the  
 117220 interface has an appropriate margin marker and shading (for example, CX, XSI, TSF,  
 117221 and so on) and the same marking appears on the reference page in the SYNOPSIS  
 117222 section. This enables a programmer to easily identify that the interface is extending an  
 117223 ISO C standard header.
- 117224 **MX and MXX**  
 117225 These two margin codes both relate to the IEC 60559 Floating-Point option. The MX  
 117226 code denotes functionality that is mandated by the ISO C standard for IEC 60559  
 117227 implementations; the MXX code denotes IEC 60559 functionality that is an extension to  
 117228 the ISO C standard.
- 117229 **OB** This margin code is used to denote obsolescent behavior and thus flag a possible future  
 117230 applications portability warning.
- 117231 **OH** The Single UNIX Specification has historically tried to reduce the number of headers an  
 117232 application has had to include when using a particular interface. Sometimes this was  
 117233 fewer than the base standard, and hence a notation is used to flag which headers are  
 117234 optional if you are using a system supporting the XSI option.

#### 117235 A.1.7.2 Margin Code Notation

- 117236 Since some features may depend on one or more options, or require more than one option, a  
 117237 notation is used. Where a feature requires support of a single option, a single margin code will  
 117238 occur in the margin. If it depends on two options and both are required, then the codes will  
 117239 appear with a <space> separator. If either of two options are required, then a logical OR is  
 117240 denoted using the ' | ' symbol. If more than two codes are used, a special notation is used.

## 117241 A.2 Conformance

- 117242 The terms “profile” and “profiling” are used throughout this section.
- 117243 A profile of a standard or standards is a codified set of option selections, such that by being  
 117244 conformant to a profile, particular classes of users are specifically supported.

### 117245 A.2.1 Implementation Conformance

- 117246 These definitions allow application developers to know what to depend on in an  
 117247 implementation.
- 117248 There is no definition of a “strictly conforming implementation”; that would be an  
 117249 implementation that provides *only* those facilities specified by POSIX.1 with no extensions  
 117250 whatsoever. This is because no actual operating system implementation can exist without  
 117251 system administration and initialization facilities that are beyond the scope of POSIX.1.

## 117252 A.2.1.1 Requirements

117253 The word “support” is used in certain instances, rather than “provide”, in order to allow an  
117254 implementation that has no resident software development facilities, but that supports the  
117255 execution of a *Strictly Conforming POSIX.1 Application*, to be a *conforming implementation*.

## 117256 A.2.1.2 Documentation

117257 The conformance documentation is required to use the same numbering scheme as POSIX.1 for  
117258 purposes of cross-referencing. All options that an implementation chooses are reflected in  
117259 **<limits.h>** and **<unistd.h>**.

117260 Note that the use of “may” in terms of where conformance documents record where  
117261 implementations may vary, implies that it is not required to describe those features identified as  
117262 undefined or unspecified.

117263 Other aspects of systems must be evaluated by purchasers for suitability. Many systems  
117264 incorporate buffering facilities, maintaining updated data in volatile storage and transferring  
117265 such updates to non-volatile storage asynchronously. Various exception conditions, such as a  
117266 power failure or a system crash, can cause this data to be lost. The data may be associated with a  
117267 file that is still open, with one that has been closed, with a directory, or with any other internal  
117268 system data structures associated with permanent storage. This data can be lost, in whole or  
117269 part, so that only careful inspection of file contents could determine that an update did not  
117270 occur.

117271 Also, interrelated file activities, where multiple files and/or directories are updated, or where  
117272 space is allocated or released in the file system structures, can leave inconsistencies in the  
117273 relationship between data in the various files and directories, or in the file system itself. Such  
117274 inconsistencies can break applications that expect updates to occur in a specific sequence, so that  
117275 updates in one place correspond with related updates in another place.

117276 For example, if a user creates a file, places information in the file, and then records this action in  
117277 another file, a system or power failure at this point followed by restart may result in a state in  
117278 which the record of the action is permanently recorded, but the file created (or some of its  
117279 information) has been lost. The consequences of this to the user may be undesirable. For a user  
117280 on such a system, the only safe action may be to require the system administrator to have a  
117281 policy that requires, after any system or power failure, that the entire file system must be  
117282 restored from the most recent backup copy (causing all intervening work to be lost).

117283 The characteristics of each implementation will vary in this respect and may or may not meet  
117284 the requirements of a given application or user. Enforcement of such requirements is beyond the  
117285 scope of POSIX.1. It is up to the purchaser to determine what facilities are provided in an  
117286 implementation that affect the exposure to possible data or sequence loss, and also what  
117287 underlying implementation techniques and/or facilities are provided that reduce or limit such  
117288 loss or its consequences.

## 117289 A.2.1.3 POSIX Conformance

117290 This really means conformance to the base standard; however, since this document includes the  
117291 core material of the Single UNIX Specification, the standard developers decided that it was  
117292 appropriate to segment the conformance requirements into two, the former for the base  
117293 standard, and the latter for the Single UNIX Specification (denoted XSI Conformance).

117294 Within POSIX.1 there are some symbolic constants that, if defined to a certain value or range of  
117295 values, indicate that a certain option is enabled. Other symbolic constants exist in POSIX.1 for  
117296 other reasons.



117297 In this version, some features that were previously optional have been made mandatory. For  
 117298 backwards compatibility, the symbolic constants associated with the option are still required  
 117299 now with fixed allowable ranges or values. The following options from the previous version of  
 117300 this standard are now mandatory:

117301        \_POSIX\_ASYNCHRONOUS\_IO  
 117302        \_POSIX\_BARRIERS  
 117303        \_POSIX\_CLOCK\_SELECTION  
 117304        \_POSIX\_MAPPED\_FILES  
 117305        \_POSIX\_MEMORY\_PROTECTION  
 117306        \_POSIX\_READER\_WRITER\_LOCKS  
 117307        \_POSIX\_REALTIME\_SIGNALS  
 117308        \_POSIX\_SEMAPHORES  
 117309        \_POSIX\_SPIN\_LOCKS  
 117310        \_POSIX\_THREAD\_SAFE\_FUNCTIONS  
 117311        \_POSIX\_THREADS  
 117312        \_POSIX\_TIMEOUTS  
 117313        \_POSIX\_TIMERS

117314 A POSIX-conformant system may support the XSI option required by the Single UNIX  
 117315 Specification. This was intentional since the standard developers intend them to be upwards-  
 117316 compatible, so that a system conforming to the Single UNIX Specification can also conform to  
 117317 the base standard at the same time.

#### 117318 A.2.1.4 XSI Conformance

117319 This section is included to describe the conformance requirements for the base volumes of the  
 117320 Single UNIX Specification.

117321 XSI conformance can be thought of as a profile, selecting certain options from POSIX.1-2008.

#### 117322 A.2.1.5 Option Groups

117323 The concept of “Option Groups” is included to allow collections of related functions or options  
 117324 to be grouped together. This has been used as follows: the “XSI Option Groups” have been  
 117325 created to allow super-options, collections of underlying options and related functions, to be  
 117326 collectively supported by XSI-conforming systems.

117327 The standard developers considered the matter of subprofiling and decided it was better to  
 117328 include an enabling mechanism rather than detailed normative requirements. A set of  
 117329 subprofiling options was developed and included later in this volume of POSIX.1-2008 as an  
 117330 informative illustration.

#### 117331 Subprofiling Considerations

117332 The goal of not simultaneously fixing maximums and minimums was to allow implementations  
 117333 of the base standard or standards to support multiple profiles without conflict.

The following summarizes the rules for the limit types:

Limit Type	Fixed Value	Minimum Acceptable Value	Maximum Acceptable Value
Standard Profile	Xs Xp == Xs (No change)	Ys Yp >= Ys (May increase the limit)	Zs Zp <= Zs (May decrease the limit)

The intent is that ranges specified by limits in profiles be entirely contained within the corresponding ranges of the base standard or standards being profiled, and that the unlimited end of a range in a base standard must remain unlimited in any profile of that standard.

Thus, the fixed `_POSIX_*` limits are constants and must not be changed by a profile. The variable counterparts (typically without the leading `_POSIX_`) can be changed but still remain semantically the same; that is, they still allow implementation values to vary as long as they meet the requirements for that value (be it a minimum or maximum).

Where a profile does not provide a feature upon which a limit is based, the limit is not relevant. Applications written to that profile should be written to operate independently of the value of the limit.

An example which has previously allowed implementations to support both the base standard and two other profiles in a compatible manner follows:

```
Base standard (POSIX.1-1996): _POSIX_CHILD_MAX 6
Base standard: CHILD_MAX    minimum maximum _POSIX_CHILD_MAX
FIPS profile/SUSv2  CHILD_MAX    25 (minimum maximum)
```

Another example:

```
Base standard (POSIX.1-1996): _POSIX_NGROUPS_MAX 0
Base standard: NGROUPS_MAX    minimum maximum _POSIX_NGROUP_MAX
FIPS profile/SUSv2  NGROUPS_MAX    8
```

A profile may lower a minimum maximum below the equivalent `_POSIX` value:

```
Base standard: _POSIX_foo_MAX    Z
Base standard: foo_MAX    _POSIX_foo_MAX
profile standard : foo_MAX    X (X can be less than, equal to,
                                or greater than _POSIX_foo_MAX)
```

In this case an implementation conforming to the profile may not conform to the base standard, but an implementation to the base standard will conform to the profile.

#### A.2.1.6 Options

The final subsections within *Implementation Conformance* list the core options within POSIX.1-2008. This includes both options for the System Interfaces volume of POSIX.1-2008 and the Shell and Utilities volume of POSIX.1-2008.



## 117370 A.2.2 Application Conformance

117371 These definitions guide users or adapters of applications in determining on which  
117372 implementations an application will run and how much adaptation would be required to make  
117373 it run on others. These definitions are modeled after related ones in the ISO C standard.

117374 POSIX.1 occasionally uses the expressions “portable application” or “conforming application”.  
117375 As they are used, these are synonyms for any of these terms. The differences between the classes  
117376 of application conformance relate to the requirements for other standards, the options supported  
117377 (such as the XSI option) or, in the case of the Conforming POSIX.1 Application Using Extensions,  
117378 to implementation extensions. When one of the less explicit expressions is used, it should be  
117379 apparent from the context of the discussion which of the more explicit names is appropriate

### 117380 A.2.2.1 Strictly Conforming POSIX Application

117381 This definition is analogous to that of an ISO C standard “conforming program”.

117382 The major difference between a Strictly Conforming POSIX Application and an ISO C standard  
117383 strictly conforming program is that the latter is not allowed to use features of POSIX that are not  
117384 in the ISO C standard.

### 117385 A.2.2.2 Conforming POSIX Application

117386 Examples of <National Bodies> include ANSI, BSI, and AFNOR.

### 117387 A.2.2.3 Conforming POSIX Application Using Extensions

117388 Due to possible requirements for configuration or implementation characteristics in excess of the  
117389 specifications in <limits.h> or related to the hardware (such as array size or file space), not  
117390 every Conforming POSIX Application Using Extensions will run on every conforming  
117391 implementation.

### 117392 A.2.2.4 Strictly Conforming XSI Application

117393 This is intended to be upwards-compatible with the definition of a Strictly Conforming POSIX  
117394 Application, with the addition of the facilities and functionality included in the XSI option.

### 117395 A.2.2.5 Conforming XSI Application Using Extensions

117396 Such applications may use extensions beyond the facilities defined by POSIX.1-2008 including  
117397 the XSI option, but need to document the additional requirements.

## 117398 A.2.3 Language-Dependent Services for the C Programming Language

117399 POSIX.1 is, for historical reasons, both a specification of an operating system interface, shell and  
117400 utilities, and a C binding for that specification. Efforts had been previously undertaken to  
117401 generate a language-independent specification; however, that had failed, and the fact that the  
117402 ISO C standard is the *de facto* primary language on POSIX and the UNIX system makes this a  
117403 necessary and workable situation.

#### 117404 A.2.4 Other Language-Related Specifications

117405 There is no additional rationale provided for this section.

### 117406 A.3 Definitions

117407 The definitions in this section are stated so that they can be used as exact substitutes for the  
117408 terms in text. They should not contain requirements or cross-references to sections within  
117409 POSIX.1-2008; that is accomplished by using an informative note. In addition, the term should  
117410 not be included in its own definition. Where requirements or descriptions need to be addressed  
117411 but cannot be included in the definitions, due to not meeting the above criteria, these occur in  
117412 the General Concepts chapter.

117413 In this version, the definitions have been reworked extensively to meet style requirements and to  
117414 include terms from the base documents (see the Scope).

117415 Many of these definitions are necessarily circular, and some of the terms (such as “process”) are  
117416 variants of basic computing science terms that are inherently hard to define. Where some  
117417 definitions are more conceptual and contain requirements, these appear in the General Concepts  
117418 chapter. Those listed in this section appear in an alphabetical glossary format of terms.

117419 Some definitions must allow extension to cover terms or facilities that are not explicitly  
117420 mentioned in POSIX.1-2008. For example, the definition of “Extended Security Controls”  
117421 permits implementations beyond those defined in POSIX.1-2008.

117422 Some terms in the following list of notes do not appear in POSIX.1-2008; these are marked  
117423 suffixed with an asterisk (\*). Many of them have been specifically excluded from POSIX.1-2008  
117424 because they concern system administration, implementation, or other issues that are not  
117425 specific to the programming interface. Those are marked with a reason, such as  
117426 “implementation-defined”.

#### 117427 Appropriate Privileges

117428 One of the fundamental security problems with many historical UNIX systems has been that the  
117429 privilege mechanism is monolithic—a user has either no privileges or *all* privileges. Thus, a  
117430 successful “trojan horse” attack on a privileged process defeats all security provisions.  
117431 Therefore, POSIX.1 allows more granular privilege mechanisms to be defined. For many  
117432 historical implementations of the UNIX system, the presence of the term “appropriate  
117433 privileges” in POSIX.1 may be understood as a synonym for “superuser” (UID 0). However,  
117434 other systems have emerged where this is not the case and each discrete controllable action has  
117435 *appropriate privileges* associated with it. Because this mechanism is implementation-defined, it  
117436 must be described in the conformance document. Although that description affects several parts  
117437 of POSIX.1 where the term “appropriate privilege” is used, because the term “implementation-  
117438 defined” only appears here, the description of the entire mechanism and its effects on these  
117439 other sections belongs in this equivalent section of the conformance document. This is especially  
117440 convenient for implementations with a single mechanism that applies in all areas, since it only  
117441 needs to be described once.

**Base Character\***

The term “Base Character” has been removed, as it was felt that the use of this term within POSIX.1-2008 was common usage English.

**Byte**

The restriction that a byte is now exactly eight bits was a conscious decision by the standard developers. It came about due to a combination of factors, primarily the use of the type `int8_t` within the networking functions and the alignment with the ISO/IEC 9899:1999 standard, where the `intN_t` types are now defined.

According to the ISO/IEC 9899:1999 standard:

- The `[u]intN_t` types must be two’s complement with no padding bits and no illegal values.
- All types (apart from bit fields, which are not relevant here) must occupy an integral number of bytes.
- If a type with width  $W$  occupies  $B$  bytes with  $C$  bits per byte ( $C$  is the value of `{CHAR_BIT}`), then it has  $P$  padding bits where  $P+W=B*C$ .
- Therefore, for `int8_t`  $P=0$ ,  $W=8$ . Since  $B \geq 1$ ,  $C \geq 8$ , the only solution is  $B=1$ ,  $C=8$ .

The standard developers also felt that this was not an undue restriction for the current state-of-the-art for this version of the standard, but recognize that if industry trends continue, a wider character type may be required in the future.

**Character**

The term “character” is used to mean a sequence of one or more bytes representing a single graphic symbol. The deviation in the exact text of the ISO C standard definition for “byte” meets the intent of the rationale of the ISO C standard also clears up the ambiguity raised by the term “basic execution character set”. The octet-minimum requirement is a reflection of the `{CHAR_BIT}` value.

**Child Process**

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/3 is applied, adding the `vfork()` function to those listed.

**Clock Tick**

The ISO C standard defines a similar interval for use by the `clock()` function. There is no requirement that these intervals be the same. In historical implementations these intervals are different.

**Command**

The terms “command” and “utility” are related but have distinct meanings. Command is defined as “a directive to a shell to perform a specific task”. The directive can be in the form of a single utility name (for example, `ls`), or the directive can take the form of a compound command (for example, `"ls | grep name | pr"`). A utility is a program that can be called by name from a shell. Issuing only the name of the utility to a shell is the equivalent of a one-word command. A utility may be invoked as a separate program that executes in a different process than the command language interpreter, or it may be implemented as a part of the command language interpreter. For example, the `echo` command (the directive to perform a specific task) may be implemented such that the `echo` utility (the logic that performs the task of echoing) is in a separate program; therefore, it is executed in a process that is different from the command

language interpreter. Conversely, the logic that performs the *echo* utility could be built into the command language interpreter; therefore, it could execute in the same process as the command language interpreter.

The terms “tool” and “application” can be thought of as being synonymous with “utility” from the perspective of the operating system kernel. Tools, applications, and utilities historically have run, typically, in processes above the kernel level. Tools and utilities historically have been a part of the operating system non-kernel code and have performed system-related functions, such as listing directory contents, checking file systems, repairing file systems, or extracting system status information. Applications have not generally been a part of the operating system, and they perform non-system-related functions, such as word processing, architectural design, mechanical design, workstation publishing, or financial analysis. Utilities have most frequently been provided by the operating system distributor, applications by third-party software distributors, or by the users themselves. Nevertheless, POSIX.1-2008 does not differentiate between tools, utilities, and applications when it comes to receiving services from the system, a shell, or the standard utilities. (For example, the *xargs* utility invokes another utility; it would be of fairly limited usefulness if the users could not run their own applications in place of the standard utilities.) Utilities are not applications in the sense that they are not themselves subject to the restrictions of POSIX.1-2008 or any other standard—there is no requirement for *grep*, *stty*, or any of the utilities defined here to be any of the classes of conforming applications.

### Column Positions

In most 1-byte character sets, such as ASCII, the concept of column positions is identical to character positions and to bytes. Therefore, it has been historically acceptable for some implementations to describe line folding or tab stops or table column alignment in terms of bytes or character positions. Other character sets pose complications, as they can have internal representations longer than one octet and they can have display characters that have different widths on the terminal screen or printer.

In POSIX.1-2008 the term “column positions” has been defined to mean character—not byte—positions in input files (such as “column position 7 of the FORTRAN input”). Output files describe the column position in terms of the display width of the narrowest printable character in the character set, adjusted to fit the characteristics of the output device. It is very possible that *n* column positions will not be able to hold *n* characters in some character sets, unless all of those characters are of the narrowest width. It is assumed that the implementation is aware of the width of the various characters, deriving this information from the value of *LC\_CTYPE*, and thus can determine how many column positions to allot for each character in those utilities where it is important.

The term “column position” was used instead of the more natural “column” because the latter is frequently used in the different contexts of columns of figures, columns of table values, and so on. Wherever confusion might result, these latter types of columns are referred to as “text columns”.

### Controlling Terminal

The question of which of possibly several special files referring to the terminal is meant is not addressed in POSIX.1. The pathname */dev/tty* is a synonym for the controlling terminal associated with a process.

117527	<b>Device Number*</b>
117528	The concept is handled in <i>stat()</i> as <i>ID of device</i> .
117529	<b>Direct I/O</b>
117530	Historically, direct I/O refers to the system bypassing intermediate buffering, but may be
117531	extended to cover implementation-defined optimizations.
117532	<b>Directory</b>
117533	The format of the directory file is implementation-defined and differs radically between
117534	System V and 4.3 BSD. However, routines (derived from 4.3 BSD) for accessing directories and
117535	certain constraints on the format of the information returned by those routines are described in
117536	the <b>&lt;dirent.h&gt;</b> header.
117537	<b>Directory Entry</b>
117538	Throughout POSIX.1-2008, the term “link” is used (about the <i>link()</i> function, for example) in
117539	describing the objects that point to files from directories.
117540	<b>Display</b>
117541	The Shell and Utilities volume of POSIX.1-2008 assigns precise requirements for the terms
117542	“display” and “write”. Some historical systems have chosen to implement certain utilities
117543	without using the traditional file descriptor model. For example, the <i>vi</i> editor might employ
117544	direct screen memory updates on a personal computer, rather than a <i>write()</i> system call. An
117545	instance of user prompting might appear in a dialog box, rather than with standard error. When
117546	the Shell and Utilities volume of POSIX.1-2008 uses the term “display”, the method of
117547	outputting to the terminal is unspecified; many historical implementations use <i>termcap</i> or
117548	<i>terminfo</i> , but this is not a requirement. The term “write” is used when the Shell and Utilities
117549	volume of POSIX.1-2008 mandates that a file descriptor be used and that the output can be
117550	redirected. However, it is assumed that when the writing is directly to the terminal (it has not
117551	been redirected elsewhere), there is no practical way for a user or test suite to determine whether
117552	a file descriptor is being used. Therefore, the use of a file descriptor is mandated only for the
117553	redirection case and the implementation is free to use any method when the output is not
117554	redirected. The verb <i>write</i> is used almost exclusively, with the very few exceptions of those
117555	utilities where output redirection need not be supported: <i>tabs</i> , <i>talk</i> , <i>tput</i> , and <i>vi</i> .
117556	<b>Dot</b>
117557	The symbolic name <i>dot</i> is carefully used in POSIX.1 to distinguish the working directory
117558	filename from a period or a decimal point.
117559	<b>Dot-Dot</b>
117560	Historical implementations permit the use of these filenames without their special meanings.
117561	Such use precludes any meaningful use of these filenames by a Conforming POSIX.1
117562	Application. Therefore, such use is considered an extension, the use of which makes an
117563	implementation non-conforming; see also <a href="#">Section A.4.12</a> (on page 3506).

**Epoch**

Historically, the origin of UNIX system time was referred to as “00:00:00 GMT, January 1, 1970”. Greenwich Mean Time is actually not a term acknowledged by the international standards community; therefore, this term, “Epoch”, is used to abbreviate the reference to the actual standard, Coordinated Universal Time.

**FIFO Special File**

See [Pipe](#) (on page 3490).

**File**

It is permissible for an implementation-defined file type to be non-readable or non-writable.

**File Classes**

These classes correspond to the historical sets of permission bits. The classes are general to allow implementations flexibility in expanding the access mechanism for more stringent security environments. Note that a process is in one and only one class, so there is no ambiguity.

**Filename**

Filenames are sequences of bytes, not sequences of characters. The only bytes that this standard says cannot appear in any filename are the slash byte and the null byte. This is a side-effect of the fact that no conforming implementations of the standard currently provide a way to pass information specifying the locale associated with strings passed between user-level applications and the kernel. This decision could be revisited if implementations develop a way to associate a locale with the strings passed between kernel space and user space.

Implementations may add other restrictions to the byte sequences allowed in filenames except that any filename consisting of no more than {NAME\_MAX} bytes from the set of characters in the portable filename character set must be allowed.

See [Pathname](#) (on page 3489).

**File System**

Historically, the meaning of this term has been overloaded with two meanings: that of the complete file hierarchy, and that of a mountable subset of that hierarchy; that is, a mounted file system. POSIX.1 uses the term “file system” in the second sense, except that it is limited to the scope of a process (and root directory of a process). This usage also clarifies the domain in which a file serial number is unique.

**Graphic Character**

This definition is made available for those definitions (in particular, *TZ*) which must exclude control characters.



117597 **Group Database**

117598 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/4 is applied, removing the words “of  
117599 implementation-defined format”. See [User Database](#) (on page 3499).

117600 **Group File\***

117601 Implementation-defined; see [User Database](#) (on page 3499).

117602 **Historical Implementations\***

117603 This refers to previously existing implementations of programming interfaces and operating  
117604 systems that are related to the interface specified by POSIX.1.

117605 **Hosted Implementation\***

117606 This refers to a POSIX.1 implementation that is accomplished through interfaces from the  
117607 POSIX.1 services to some alternate form of operating system kernel services. Note that the line  
117608 between a hosted implementation and a native implementation is blurred, since most  
117609 implementations will provide some services directly from the kernel and others through some  
117610 indirect path. (For example, *fopen()* might use *open()*; or *mkfifo()* might use *mknod()*.) There is  
117611 no necessary relationship between the type of implementation and its correctness, performance,  
117612 and/or reliability.

117613 **Implementation\***

117614 This term is generally used instead of its synonym, “system”, to emphasize the consequences of  
117615 decisions to be made by system implementors. Perhaps if no options or extensions to POSIX.1  
117616 were allowed, this usage would not have occurred.

117617 The term “specific implementation” is sometimes used as a synonym for “implementation”.  
117618 This should not be interpreted too narrowly; both terms can represent a relatively broad group  
117619 of systems. For example, a hardware vendor could market a very wide selection of systems that  
117620 all used the same instruction set, with some systems desktop models and others large multi-user  
117621 minicomputers. This wide range would probably share a common POSIX.1 operating system,  
117622 allowing an application compiled for one to be used on any of the others; this is a [*specific*]  
117623 *implementation*. However, such a wide range of machines probably has some differences  
117624 between the models. Some may have different clock rates, different file systems, different  
117625 resource limits, different network connections, and so on, depending on their sizes or intended  
117626 usages. Even on two identical machines, the system administrators may configure them  
117627 differently. Each of these different systems is known by the term “a specific instance of a specific  
117628 implementation”. This term is only used in the portions of POSIX.1 dealing with runtime  
117629 queries: *sysconf()* and *pathconf()*.

117630 **Incomplete Pathname\***

117631 Absolute pathname has been adequately defined.

## Job Control

In order to understand the job control facilities in POSIX.1 it is useful to understand how they are used by a job control-cognizant shell to create the user interface effect of job control.

While the job control facilities supplied by POSIX.1 can, in theory, support different types of interactive job control interfaces supplied by different types of shells, there was historically one particular interface that was most common when the standard was originally developed (provided by BSD C Shell).

This discussion describes that interface as a means of illustrating how the POSIX.1 job control facilities can be used.

Job control allows users to selectively stop (suspend) the execution of processes and continue (resume) their execution at a later point. The user typically employs this facility via the interactive interface jointly supplied by the terminal I/O driver and a command interpreter (shell).

The user can launch jobs (command pipelines) in either the foreground or background. When launched in the foreground, the shell waits for the job to complete before prompting for additional commands. When launched in the background, the shell does not wait, but immediately prompts for new commands.

If the user launches a job in the foreground and subsequently regrets this, the user can type the suspend character (typically set to <control>-Z), which causes the foreground job to stop and the shell to begin prompting for new commands. The stopped job can be continued by the user (via special shell commands) either as a foreground job or as a background job. Background jobs can also be moved into the foreground via shell commands.

If a background job attempts to access the login terminal (controlling terminal), it is stopped by the terminal driver and the shell is notified, which, in turn, notifies the user. (Terminal access includes *read()* and certain terminal control functions, and conditionally includes *write()*.) The user can continue the stopped job in the foreground, thus allowing the terminal access to succeed in an orderly fashion. After the terminal access succeeds, the user can optionally move the job into the background via the suspend character and shell commands.

### Implementing Job Control Shells

The interactive interface described previously can be accomplished using the POSIX.1 job control facilities in the following way.

The key feature necessary to provide job control is a way to group processes into jobs. This grouping is necessary in order to direct signals to a single job and also to identify which job is in the foreground. (There is at most one job that is in the foreground on any controlling terminal at a time.)

The concept of process groups is used to provide this grouping. The shell places each job in a separate process group via the *setpgid()* function. To do this, the *setpgid()* function is invoked by the shell for each process in the job. It is actually useful to invoke *setpgid()* twice for each process: once in the child process, after calling *fork()* to create the process, but before calling one of the *exec* family of functions to begin execution of the program, and once in the parent shell process, after calling *fork()* to create the child. The redundant invocation avoids a race condition by ensuring that the child process is placed into the new process group before either the parent or the child relies on this being the case. The process group ID for the job is selected by the shell to be equal to the process ID of one of the processes in the job. Some shells choose to make one process in the job be the parent of the other processes in the job (if any). Other shells (for example, the C Shell) choose to make themselves the parent of all processes in the pipeline (job). In order to support this latter case, the *setpgid()* function accepts a process group ID parameter



since the correct process group ID cannot be inherited from the shell. The shell itself is considered to be a job and is the sole process in its own process group.

The shell also controls which job is currently in the foreground. A foreground and background job differ in two ways: the shell waits for a foreground command to complete (or stop) before continuing to read new commands, and the terminal I/O driver inhibits terminal access by background jobs (causing the processes to stop). Thus, the shell must work cooperatively with the terminal I/O driver and have a common understanding of which job is currently in the foreground. It is the user who decides which command should be currently in the foreground, and the user informs the shell via shell commands. The shell, in turn, informs the terminal I/O driver via the *tcsetpgrp()* function. This indicates to the terminal I/O driver the process group ID of the foreground process group (job). When the current foreground job either stops or terminates, the shell places itself in the foreground via *tcsetpgrp()* before prompting for additional commands. Note that when a job is created the new process group begins as a background process group. It requires an explicit act of the shell via *tcsetpgrp()* to move a process group (job) into the foreground.

When a process in a job stops or terminates, its parent (for example, the shell) receives synchronous notification by calling the *waitpid()* function with the WUNTRACED flag set. Asynchronous notification is also provided when the parent establishes a signal handler for SIGCHLD and does not specify the SA\_NOCLDSTOP flag. Usually all processes in a job stop as a unit since the terminal I/O driver always sends job control stop signals to all processes in the process group.

To continue a stopped job, the shell sends the SIGCONT signal to the process group of the job. In addition, if the job is being continued in the foreground, the shell invokes *tcsetpgrp()* to place the job in the foreground before sending SIGCONT. Otherwise, the shell leaves itself in the foreground and reads additional commands.

There is additional flexibility in the POSIX.1 job control facilities that allows deviations from the typical interface. Clearing the TOSTOP terminal flag allows background jobs to perform *write()* functions without stopping. The same effect can be achieved on a per-process basis by having a process set the signal action for SIGTTOU to SIG\_IGN.

Note that the terms “job” and “process group” can be used interchangeably. A login session that is not using the job control facilities can be thought of as a large collection of processes that are all in the same job (process group). Such a login session may have a partial distinction between foreground and background processes; that is, the shell may choose to wait for some processes before continuing to read new commands and may not wait for other processes. However, the terminal I/O driver will consider all these processes to be in the foreground since they are all members of the same process group.

In addition to the basic job control operations already mentioned, a job control-cognizant shell needs to perform the following actions.

When a foreground (not background) job stops, the shell must sample and remember the current terminal settings so that it can restore them later when it continues the stopped job in the foreground (via the *tcgetattr()* and *tcsetattr()* functions).

Because a shell itself can be spawned from a shell, it must take special action to ensure that subshells interact well with their parent shells.

A subshell can be spawned to perform an interactive function (prompting the terminal for commands) or a non-interactive function (reading commands from a file). When operating non-interactively, the job control shell will refrain from performing the job control-specific actions described above. It will behave as a shell that does not support job control. For example, all jobs will be left in the same process group as the shell, which itself remains in the process group

established for it by its parent. This allows the shell and its children to be treated as a single job by a parent shell, and they can be affected as a unit by terminal keyboard signals.

An interactive subshell can be spawned from another job control-cognizant shell in either the foreground or background. (For example, from the C Shell, the user can execute the command, `csch &`.) Before the subshell activates job control by calling `setpgid()` to place itself in its own process group and `tcsetpgrp()` to place its new process group in the foreground, it needs to ensure that it has already been placed in the foreground by its parent. (Otherwise, there could be multiple job control shells that simultaneously attempt to control mediation of the terminal.) To determine this, the shell retrieves its own process group via `getpgrp()` and the process group of the current foreground job via `tcgetpgrp()`. If these are not equal, the shell sends SIGTTIN to its own process group, causing itself to stop. When continued later by its parent, the shell repeats the process group check. When the process groups finally match, the shell is in the foreground and it can proceed to take control. After this point, the shell ignores all the job control stop signals so that it does not inadvertently stop itself.

#### *Implementing Job Control Applications*

Most applications do not need to be aware of job control signals and operations; the intuitively correct behavior happens by default. However, sometimes an application can inadvertently interfere with normal job control processing, or an application may choose to overtly effect job control in cooperation with normal shell procedures.

An application can inadvertently subvert job control processing by “blindly” altering the handling of signals. A common application error is to learn how many signals the system supports and to ignore or catch them all. Such an application makes the assumption that it does not know what this signal is, but knows the right handling action for it. The system may initialize the handling of job control stop signals so that they are being ignored. This allows shells that do not support job control to inherit and propagate these settings and hence to be immune to stop signals. A job control shell will set the handling to the default action and propagate this, allowing processes to stop. In doing so, the job control shell is taking responsibility for restarting the stopped applications. If an application wishes to catch the stop signals itself, it should first determine their inherited handling states. If a stop signal is being ignored, the application should continue to ignore it. This is directly analogous to the recommended handling of SIGINT described in the referenced UNIX Programmer’s Manual.

If an application is reading the terminal and has disabled the interpretation of special characters (by clearing the ISIG flag), the terminal I/O driver will not send SIGTSTP when the suspend character is typed. Such an application can simulate the effect of the suspend character by recognizing it and sending SIGTSTP to its process group as the terminal driver would have done. Note that the signal is sent to the process group, not just to the application itself; this ensures that other processes in the job also stop. (Note also that other processes in the job could be children, siblings, or even ancestors.) Applications should not assume that the suspend character is <control>-Z (or any particular value); they should retrieve the current setting at startup.

#### *Implementing Job Control Systems*

The intent in adding 4.2 BSD-style job control functionality was to adopt the necessary 4.2 BSD programmatic interface with only minimal changes to resolve syntactic or semantic conflicts with System V or to close recognized security holes. The goal was to maximize the ease of providing both conforming implementations and Conforming POSIX.1 Applications.

It is only useful for a process to be affected by job control signals if it is the descendant of a job control shell. Otherwise, there will be nothing that continues the stopped process.

POSIX.1 does not specify how controlling terminal access is affected by a user logging out (that

117775 is, by a controlling process terminating). 4.2 BSD uses the *vhangup()* function to prevent any  
 117776 access to the controlling terminal through file descriptors opened prior to logout. System V does  
 117777 not prevent controlling terminal access through file descriptors opened prior to logout (except  
 117778 for the case of the special file, */dev/tty*). Some implementations choose to make processes  
 117779 immune from job control after logout (that is, such processes are always treated as if in the  
 117780 foreground); other implementations continue to enforce foreground/background checks after  
 117781 logout. Therefore, a Conforming POSIX.1 Application should not attempt to access the  
 117782 controlling terminal after logout since such access is unreliable. If an implementation chooses to  
 117783 deny access to a controlling terminal after its controlling process exits, POSIX.1 requires a certain  
 117784 type of behavior (see [Controlling Terminal](#), on page 3480).

#### 117785 **Kernel\***

117786 See [System Call\\*](#) (on page 3497).

#### 117787 **Library Routine\***

117788 See [System Call\\*](#) (on page 3497).

#### 117789 **Logical Device\***

117790 Implementation-defined.

#### 117791 **Map**

117792 The definition of map is included to clarify the usage of mapped pages in the description of the  
 117793 behavior of process memory locking.

#### 117794 **Memory-Resident**

117795 The term “memory-resident” is historically understood to mean that the so-called resident pages  
 117796 are actually present in the physical memory of the computer system and are immune from  
 117797 swapping, paging, copy-on-write faults, and so on. This is the actual intent of POSIX.1-2008 in  
 117798 the process memory locking section for implementations where this is logical. But for some  
 117799 implementations—primarily mainframes—actually locking pages into primary storage is not  
 117800 advantageous to other system objectives, such as maximizing throughput. For such  
 117801 implementations, memory locking is a “hint” to the implementation that the application wishes  
 117802 to avoid situations that would cause long latencies in accessing memory. Furthermore, there are  
 117803 other implementation-defined issues with minimizing memory access latencies that “memory  
 117804 residency” does not address—such as MMU reload faults. The definition attempts to  
 117805 accommodate various implementations while allowing conforming applications to specify to the  
 117806 implementation that they want or need the best memory access times that the implementation  
 117807 can provide.

#### 117808 **Memory Object\***

117809 The term “memory object” usually implies shared memory. If the object is the same as a  
 117810 filename in the file system name space of the implementation, it is expected that the data written  
 117811 into the memory object be preserved on disk. A memory object may also apply to a physical  
 117812 device on an implementation. In this case, writes to the memory object are sent to the controller  
 117813 for the device and reads result in control registers being returned.

**Mount Point\***

The directory on which a “mounted file system” is mounted. This term, like *mount()* and *umount()*, was not included because it was implementation-defined.

**Mounted File System\***

See [File System](#) (on page 3482).

**Name**

There are no explicit limits in POSIX.1-2008 on the sizes of names, words (see the definition of word in the Base Definitions volume of POSIX.1-2008), lines, or other objects. However, other implicit limits do apply: shell script lines produced by many of the standard utilities cannot exceed {LINE\_MAX} and the sum of exported variables comes under the {ARG\_MAX} limit. Historical shells dynamically allocate memory for names and words and parse incoming lines a character at a time. Lines cannot have an arbitrary {LINE\_MAX} limit because of historical practice, such as *makefiles*, where *make* removes the <newline> characters associated with the commands for a target and presents the shell with one very long line. The text on INPUT FILES in XCU [Section 1.4](#) (on page 2326) does allow a shell to run out of memory, but it cannot have arbitrary programming limits.

**Native Implementation\***

This refers to an implementation of POSIX.1 that interfaces directly to an operating system kernel; see also *hosted implementation*. A similar concept is a native UNIX system, which would be a kernel derived from one of the original UNIX system products.

**Nice Value**

This definition is not intended to suggest that all processes in a system have priorities that are comparable. Scheduling policy extensions, such as adding realtime priorities, make the notion of a single underlying priority for all scheduling policies problematic. Some implementations may implement the features related to *nice* to affect all processes on the system, others to affect just the general time-sharing activities implied by POSIX.1-2008, and others may have no effect at all. Because of the use of “implementation-defined” in *nice* and *renice*, a wide range of implementation strategies is possible.

**Open File Description**

An “open file description”, as it is currently named, describes how a file is being accessed. What is currently called a “file descriptor” is actually just an identifier or “handle”; it does not actually describe anything.

The following alternate names were discussed:

- For “open file description”:  
“open instance”, “file access description”, “open file information”, and “file access information”.
- For “file descriptor”:  
“file handle”, “file number” (cf., *fileno()*). Some historical implementations use the term “file table entry”.

**Orphaned Process Group**

Historical implementations have a concept of an orphaned process, which is a process whose parent process has exited. When job control is in use, it is necessary to prevent processes from being stopped in response to interactions with the terminal after they no longer are controlled by a job control-cognizant program. Because signals generated by the terminal are sent to a process group and not to individual processes, and because a signal may be provoked by a process that is not orphaned, but sent to another process that is orphaned, it is necessary to define an orphaned process group. The definition assumes that a process group will be manipulated as a group and that the job control-cognizant process controlling the group is outside of the group and is the parent of at least one process in the group (so that state changes may be reported via *waitpid()*). Therefore, a group is considered to be controlled as long as at least one process in the group has a parent that is outside of the process group, but within the session.

This definition of orphaned process groups ensures that a session leader's process group is always considered to be orphaned, and thus it is prevented from stopping in response to terminal signals.

**Page**

The term "page" is defined to support the description of the behavior of memory mapping for shared memory and memory mapped files, and the description of the behavior of process memory locking. It is not intended to imply that shared memory/file mapping and memory locking are applicable only to "paged" architectures. For the purposes of POSIX.1-2008, whatever the granularity on which an architecture supports mapping or locking, this is considered to be a "page". If an architecture cannot support the memory mapping or locking functions specified by POSIX.1-2008 on any granularity, then these options will not be implemented on the architecture.

**Pathname**

Pathnames historically allowed all bytes except for the <slash> and <NUL> characters. For compatibility with existing file systems, this usage is maintained throughout the standard by noting that a pathname need not be a valid character string in all locales. However, the properties of the portable filename character set are such that a pathname using only those characters and the <slash> is portable in all locales as a character string.

**Passwd File\***

Implementation-defined; see [User Database](#) (on page 3499).

**Parent Directory**

There may be more than one directory entry pointing to a given directory in some implementations. The wording here identifies that exactly one of those is the parent directory. In pathname resolution, dot-dot is identified as the way that the unique directory is identified. (That is, the parent directory is the one to which dot-dot points.) In the case of a remote file system, if the same file system is mounted several times, it would appear as if they were distinct file systems (with interesting synchronization properties).

**117892 Pipe**

117893 It proved convenient to define a pipe as a special case of a FIFO, even though historically the  
117894 latter was not introduced until System III and does not exist at all in 4.3 BSD.

**117895 Portable Filename Character Set**

117896 The encoding of this character set is not specified—specifically, ASCII is not required. But the  
117897 implementation must provide a unique character code for each of the printable graphics  
117898 specified by POSIX.1; see also [Section A.4.6](#) (on page 3502).

117899 Situations where characters beyond the portable filename character set (or historically ASCII or  
117900 the ISO/IEC 646:1991 standard) would be used (in a context where the portable filename  
117901 character set or the ISO/IEC 646:1991 standard is required by POSIX.1) are expected to be  
117902 common. Although such a situation renders the use technically non-compliant, mutual  
117903 agreement among the users of an extended character set will make such use portable between  
117904 those users. Such a mutual agreement could be formalized as an optional extension to POSIX.1.  
117905 (Making it required would eliminate too many possible systems, as even those systems using the  
117906 ISO/IEC 646:1991 standard as a base character set extend their character sets for Western  
117907 Europe and the rest of the world in different ways.)

117908 Nothing in POSIX.1 is intended to preclude the use of extended characters where interchange is  
117909 not required or where mutual agreement is obtained. It has been suggested that in several places  
117910 “should” be used instead of “shall”. Because (in the worst case) use of any character beyond the  
117911 portable filename character set would render the program or data not portable to all possible  
117912 systems, no extensions are permitted in this context.

**117913 Process Lifetime**

117914 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/5 is applied, adding *fork()*, *posix\_spawn()*,  
117915 *posix\_spawnp()*, and *vfork()* to the list of functions.

**117916 Process Termination**

117917 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/6 is applied, rewording the definition to  
117918 address the “passive exit” on termination of the last thread or the *\_Exit()* function.

**117919 Regular File**

117920 POSIX.1 does not intend to preclude the addition of structuring data (for example, record  
117921 lengths) in the file, as long as such data is not visible to an application that uses the features  
117922 described in POSIX.1.

**117923 Root Directory**

117924 This definition permits the operation of *chroot()*, even though that function is not in POSIX.1; see  
117925 also [Section A.4.5](#) (on page 3501).

**117926 Root File System\***

117927 Implementation-defined.



**Root of a File System\***

Implementation-defined; see [Mount Point\\*](#) (on page 3488).

**Signal**

The definition implies a double meaning for the term. Although a signal is an event, common usage implies that a signal is an identifier of the class of event.

**Superuser\***

This concept, with great historical significance to UNIX system users, has been replaced with the notion of appropriate privileges.

**Supplementary Group ID**

The POSIX.1-1990 standard is inconsistent in its treatment of supplementary groups. The definition of supplementary group ID explicitly permits the effective group ID to be included in the set, but wording in the description of the *setuid()* and *setgid()* functions states: “Any supplementary group IDs of the calling process remain unchanged by these function calls”. In the case of *setgid()* this contradicts that definition. In addition, some felt that the unspecified behavior in the definition of supplementary group IDs adds unnecessary portability problems. The standard developers considered several solutions to this problem:

1. Reword the description of *setgid()* to permit it to change the supplementary group IDs to reflect the new effective group ID. A problem with this is that it adds more “may”s to the wording and does not address the portability problems of this optional behavior.
2. Mandate the inclusion of the effective group ID in the supplementary set (giving {NGROUPS\_MAX} a minimum value of 1). This is the behavior of 4.4 BSD. In that system, the effective group ID is the first element of the array of supplementary group IDs (there is no separate copy stored, and changes to the effective group ID are made only in the supplementary group set). By convention, the initial value of the effective group ID is duplicated elsewhere in the array so that the initial value is not lost when executing a set-group-ID program.
3. Change the definition of supplementary group ID to exclude the effective group ID and specify that the effective group ID does not change the set of supplementary group IDs. This is the behavior of 4.2 BSD, 4.3 BSD, and System V Release 4.
4. Change the definition of supplementary group ID to exclude the effective group ID, and require that *getgroups()* return the union of the effective group ID and the supplementary group IDs.
5. Change the definition of {NGROUPS\_MAX} to be one more than the number of supplementary group IDs, so it continues to be the number of values returned by *getgroups()* and existing applications continue to work. This alternative is effectively the same as the second (and might actually have the same implementation).

The standard developers decided to permit either 2 or 3. The effective group ID is orthogonal to the set of supplementary group IDs, and it is implementation-defined whether *getgroups()* returns this. If the effective group ID is returned with the set of supplementary group IDs, then all changes to the effective group ID affect the supplementary group set returned by *getgroups()*. It is permissible to eliminate duplicates from the list returned by *getgroups()*. However, if a group ID is contained in the set of supplementary group IDs, setting the group ID to that value and then to a different value should not remove that value from the supplementary group IDs.

The definition of supplementary group IDs has been changed to not include the effective group

117972 ID. This simplifies permanent rationale and makes the relevant functions easier to understand.  
 117973 The *getgroups()* function has been modified so that it can, on an implementation-defined basis,  
 117974 return the effective group ID. By making this change, functions that modify the effective group  
 117975 ID do not need to discuss adding to the supplementary group list; the only view into the  
 117976 supplementary group list that the application developer has is through the *getgroups()* function.

## 117977 Symbolic Constant

117978 Earlier versions of this standard used a variety of terms other than “macro” for many of the  
 117979 constants defined in headers, and it was not clear in which of these cases they were required to  
 117980 be macros or not, or to be pre-processor constants (i.e., usable in *#if*) or not. In cases where the  
 117981 symbols had a reserved prefix or suffix, there was often inconsistency between whether the  
 117982 prefix/suffix was reserved only for macros or for any use, and whether the term “macro” or a  
 117983 different term was used in the descriptions of the symbols. There were also some unintentional  
 117984 differences from the ISO C standard.

117985 One of the most commonly used terms was “symbolic constant”. This has now been designated  
 117986 as the default term to be used wherever appropriate, and a formal definition of the term has  
 117987 been added giving the exact requirements for symbols that are described as symbolic constants.

117988 The standard developers have performed a major rationalization of the header descriptions of  
 117989 symbols with constant values according to the following policy:

- 117990 • Where symbols are from the ISO C standard, the wording from the ISO C standard (or  
 117991 equivalent, in cases where the exact wording is not appropriate) is used to describe them.
- 117992 • For all other constants, the first choice is to use “symbolic constant” when the  
 117993 requirements for the symbol are a reasonably close fit with those of the term.

117994 The description of the symbol can override individual requirements for symbolic  
 117995 constants; e.g., to specify a non-integer type, or to add a requirement that the symbol is  
 117996 usable in *#if* preprocessor directives.

- 117997 • When neither of the above apply, the exact requirements are stated in the description.  
 117998 (Note that macros are not required to be usable in *#if*, or even to expand to constant  
 117999 expressions, unless explicitly stated.)

- 118000 • In cases where there is a reserved prefix or suffix, if the symbol(s) with that prefix/suffix  
 118001 are from the ISO C standard and are required to be macros, or if the symbol is required to  
 118002 be usable in *#if*, then the prefix/suffix is reserved for use only as macros. If the symbol(s)  
 118003 are “symbolic constants” and not required to be usable in *#if*, the prefix/suffix is reserved  
 118004 for any use except in a few special cases.

118005 Where a constant is required to be a macro but is also allowed to be another type of constant  
 118006 such as an enumeration constant, on implementations which do define it as another type of  
 118007 constant the macro is typically defined as follows:

```
118008 #define macro_name macro_name
```

118009 This allows applications to use *#ifdef*, etc. to determine whether the macro is defined, but the  
 118010 macro is not usable in *#if* preprocessor directives because the preprocessor will treat the  
 118011 unexpanded word *macro\_name* as having the value zero.



## Symbolic Link

Earlier versions of this standard did not require symbolic links to have attributes such as ownership and a file serial number. This was because the 4.4 BSD implementation did not have them, and it was expected that other implementations may wish to do the same. However, experience with 4.4 BSD has shown that symbolic links implemented in this way cause problems for users and application developers, and later BSD systems have reverted to using *inodes* to implement symbolic links. Allowing *no-inode* symbolic links also caused problems in the standard. For example, leaving the *st\_ino* value for symbolic links unspecified meant that the common technique of comparing the *st\_dev* and *st\_ino* values for two pathnames to see if they refer to the same file could only be used with *stat()* in conforming applications and not with *lstat()*. The standard now requires symbolic links to have meaningful values for the same **struct stat** fields as regular files, except for the file mode bits in *st\_mode*. Historically, the file mode bits were unused (the contents of a symbolic link could always be read), but implementations differed as to whether the file mode bits (as returned in *st\_mode* or reported by *ls -l*) were set according to the *umask* or just to a fixed value such as 0777. Accordingly, the standard requires the file mode bits to be ignored by *readlink()* and when a symbolic link is followed during pathname resolution, but leaves the corresponding part of the value returned in *st\_mode* unspecified.

Historical implementations were followed when determining which interfaces should apply to symbolic links. Interfaces that historically followed symbolic links include *chmod()*, *stat()*, and *utime()*. Interfaces that historically did not follow symbolic links include *lstat()*, *rename()*, *remove()*, *rmdir()*, and *unlink()*. For *chown()* and *link()*, historical implementations differed. POSIX.1-2008 inherited the *lchown()* function from the Single UNIX Specification, Version 2, and therefore requires *chown()* to follow symbolic links. Earlier versions of this standard required *link()* to follow symbolic links, but with the addition of the *linkat()* function (which has a flag to indicate whether to follow symbolic links), both behaviors are now allowed for *link()*.

When the final component of a pathname is a symbolic link, the standard requires that a trailing *<slash>* causes the link to be followed. This is the behavior of historical implementations. For example, for */a/b* and */a/b/*, if */a/b* is a symbolic link to a directory, then */a/b* refers to the symbolic link, and */a/b/* refers to the directory to which the symbolic link points.

Because a symbolic link and its referenced object coexist in the file system name space, confusion can arise in distinguishing between the link itself and the referenced object. Historically, utilities and system calls have adopted their own link following conventions in a somewhat *ad hoc* fashion. Rules for a uniform approach are outlined here, although historical practice has been adhered to as much as was possible. To promote consistent system use, user-written utilities are encouraged to follow these same rules.

Symbolic links are handled either by operating on the link itself, or by operating on the object referenced by the link. In the latter case, an application or system call is said to “follow” the link. Symbolic links may reference other symbolic links, in which case links are dereferenced until an object that is not a symbolic link is found, a symbolic link that references a file that does not exist is found, or a loop is detected. (Current implementations do not detect loops, but have a limit on the number of symbolic links that they will dereference before declaring it an error.)

There are four domains for which default symbolic link policy is established in a system. In almost all cases, there are utility options that override this default behavior. The four domains are as follows:

1. Symbolic links specified to system calls that take pathname arguments
2. Symbolic links specified as command line pathname arguments to utilities that are not performing a traversal of a file hierarchy

3. Symbolic links referencing files not of type directory, specified to utilities that are performing a traversal of a file hierarchy

4. Symbolic links referencing files of type directory, specified to utilities that are performing a traversal of a file hierarchy

#### First Domain

The first domain is considered in earlier rationale.

#### Second Domain

The reason this category is restricted to utilities that are not traversing the file hierarchy is that some standard utilities take an option that specifies a hierarchical traversal, but by default operate on the arguments themselves. Generally, users specifying the option for a file hierarchy traversal wish to operate on a single, physical hierarchy, and therefore symbolic links, which may reference files outside of the hierarchy, are ignored. For example, *chown owner file* is a different operation from the same command with the **-R** option specified. In this example, the behavior of the command *chown owner file* is described here, while the behavior of the command *chown -R owner file* is described in the third and fourth domains.

The general rule is that the utilities in this category follow symbolic links named as arguments.

Exceptions in the second domain are:

- The *mv* and *rm* utilities do not follow symbolic links named as arguments, but respectively attempt to rename or delete them.
- The *ls* utility is also an exception to this rule. For compatibility with historical systems, when the **-R** option is not specified, the *ls* utility follows symbolic links named as arguments if the **-L** option is specified or if the **-F**, **-d**, or **-l** options are not specified. (If the **-L** option is specified, *ls* always follows symbolic links; it is the only utility where the **-L** option affects its behavior even though a tree walk is not being performed.)

All other standard utilities, when not traversing a file hierarchy, always follow symbolic links named as arguments.

Historical practice is that the **-h** option is specified if standard utilities are to act upon symbolic links instead of upon their targets. Examples of commands that have historically had a **-h** option for this purpose are the *chgrp*, *chown*, *file*, and *test* utilities.

#### Third Domain

The third domain is symbolic links, referencing files not of type directory, specified to utilities that are performing a traversal of a file hierarchy. (This includes symbolic links specified as command line pathname arguments or encountered during the traversal.)

The intention of the Shell and Utilities volume of POSIX.1-2008 is that the operation that the utility is performing is applied to the symbolic link itself, if that operation is applicable to symbolic links. If the operation is not applicable to symbolic links, the symbolic link should be ignored. Specifically, by default, no change should be made to the file referenced by the symbolic link.

#### Fourth Domain

The fourth domain is symbolic links referencing files of type directory, specified to utilities that are performing a traversal of a file hierarchy. (This includes symbolic links specified as command line pathname arguments or encountered during the traversal.)

Most standard utilities do not, by default, indirect into the file hierarchy referenced by the symbolic link. (The Shell and Utilities volume of POSIX.1-2008 uses the informal term “physical

walk” to describe this case. The case where the utility does indirect through the symbolic link is termed a “logical walk”.)

There are three reasons for the default to be a physical walk:

1. With very few exceptions, a physical walk has been the historical default on UNIX systems supporting symbolic links. Because some utilities (that is, *rm*) must default to a physical walk, regardless, changing historical practice in this regard would be confusing to users and needlessly incompatible.
2. For systems where symbolic links have the historical file attributes (that is, *owner*, *group*, *mode*), defaulting to a logical traversal would require the addition of a new option to the commands to modify the attributes of the link itself. This is painful and more complex than the alternatives.
3. There is a security issue with defaulting to a logical walk. Historically, the command *chown -R user file* has been safe for the superuser because *setuid* and *setgid* bits were lost when the ownership of the file was changed. If the walk were logical, changing ownership would no longer be safe because a user might have inserted a symbolic link pointing to any file in the tree. Again, this would necessitate the addition of an option to the commands doing hierarchy traversal to not indirect through the symbolic links, and historical scripts doing recursive walks would instantly become security problems. While this is mostly an issue for system administrators, it is preferable to not have different defaults for different classes of users.

However, the standard developers agreed to leave it unspecified to achieve consensus.

As consistently as possible, users may cause standard utilities performing a file hierarchy traversal to follow any symbolic links named on the command line, regardless of the type of file they reference, by specifying the **-H** (for half logical) option. This option is intended to make the command line name space look like the logical name space.

As consistently as possible, users may cause standard utilities performing a file hierarchy traversal to follow any symbolic links named on the command line as well as any symbolic links encountered during the traversal, regardless of the type of file they reference, by specifying the **-L** (for logical) option. This option is intended to make the entire name space look like the logical name space.

For consistency, implementors are encouraged to use the **-P** (for “physical”) flag to specify the physical walk in utilities that do logical walks by default for whatever reason.

When one or more of the **-H**, **-L**, and **-P** flags can be specified, the last one specified determines the behavior of the utility. This permits users to alias commands so that the default behavior is a logical walk and then override that behavior on the command line.

#### *Exceptions in the Third and Fourth Domains*

The *ls* and *rm* utilities are exceptions to these rules. The *rm* utility never follows symbolic links and does not support the **-H**, **-L**, or **-P** options. Some historical versions of *ls* always followed symbolic links given on the command line whether the **-L** option was specified or not. Historical versions of *ls* did not support the **-H** option. In POSIX.1-2008, unless one of the **-H** or **-L** options is specified, the *ls* utility only follows symbolic links to directories that are given as operands. The *ls* utility does not support the **-P** option.

The Shell and Utilities volume of POSIX.1-2008 requires that the standard utilities *ls*, *find*, and *pax* detect infinite loops when doing logical walks; that is, a directory, or more commonly a symbolic link, that refers to an ancestor in the current file hierarchy. If the file system itself is corrupted, causing the infinite loop, it may be impossible to recover. Because *find* and *ls* are often used in system administration and security applications, they should attempt to recover and

118151 continue as best as they can. The *pax* utility should terminate because the archive it was creating  
118152 is by definition corrupted. Other, less vital, utilities should probably simply terminate as well.  
118153 Implementations are strongly encouraged to detect infinite loops in all utilities.

118154 Historical practice is shown in [Table A-1](#) (on page 3497). The heading **SVID3** stands for the  
118155 Third Edition of the System V Interface Definition.

118156 Historically, several shells have had built-in versions of the *pwd* utility. In some of these shells,  
118157 *pwd* reported the physical path, and in others, the logical path. Implementations of the shell  
118158 corresponding to POSIX.1-2008 must report the logical path by default.

118159 The *cd* command is required, by default, to treat the filename dot-dot logically. Implementors are  
118160 required to support the **-P** flag in *cd* so that users can have their current environment handled  
118161 physically. In 4.3 BSD, *chgrp* during tree traversal changed the group of the symbolic link, not  
118162 the target. Symbolic links in 4.4 BSD did not have *owner*, *group*, *mode*, or other standard UNIX  
118163 system file attributes.

118164

**Table A-1** Historical Practice for Symbolic Links

Utility	SVID3	4.3 BSD	4.4 BSD	POSIX	Comments
118165 <i>cd</i>				–L	Treat " . . " logically.
118166 <i>cd</i>				–P	Treat " . . " physically.
118167 <i>chgrp</i>			–H	–H	Follow command line symlinks.
118168 <i>chgrp</i>			–h	–L	Follow symlinks.
118169 <i>chgrp</i>	–h			–h	Affect the symlink.
118170 <i>chmod</i>					Affect the symlink.
118171 <i>chmod</i>			–H		Follow command line symlinks.
118172 <i>chmod</i>			–h		Follow symlinks.
118173 <i>chown</i>			–H	–H	Follow command line symlinks.
118174 <i>chown</i>			–h	–L	Follow symlinks.
118175 <i>chown</i>	–h			–h	Affect the symlink.
118176 <i>cp</i>			–H	–H	Follow command line symlinks.
118177 <i>cp</i>			–h	–L	Follow symlinks.
118178 <i>cpio</i>	–L		–L		Follow symlinks.
118179 <i>du</i>			–H	–H	Follow command line symlinks.
118180 <i>du</i>			–h	–L	Follow symlinks.
118181 <i>file</i>	–h			–h	Affect the symlink.
118182 <i>find</i>			–H	–H	Follow command line symlinks.
118183 <i>find</i>			–h	–L	Follow symlinks.
118184 <i>find</i>	–follow		–follow		Follow symlinks.
118185 <i>ln</i>	–s	–s	–s	–s	Create a symbolic link.
118186 <i>ls</i>	–L	–L	–L	–L	Follow symlinks.
118187 <i>ls</i>				–H	Follow command line symlinks.
118188 <i>mv</i>					Operates on the symlink.
118189 <i>pax</i>			–H	–H	Follow command line symlinks.
118190 <i>pax</i>			–h	–L	Follow symlinks.
118191 <i>pwd</i>				–L	Printed path may contain symlinks.
118192 <i>pwd</i>				–P	Printed path will not contain symlinks.
118193 <i>rm</i>					Operates on the symlink.
118194 <i>tar</i>			–H		Follow command line symlinks.
118195 <i>tar</i>		–h	–h		Follow symlinks.
118196 <i>test</i>	–h		–h	–h	Affect the symlink.

118198

**Synchronously-Generated Signal**

118199

Those signals that may be generated synchronously include SIGABRT, SIGBUS, SIGILL, SIGFPE, SIGPIPE, and SIGSEGV.

118200

118201

Any signal sent via the *raise()* function or a *kill()* function targeting the current process is also considered synchronous.

118202

118203

**System Call\***

118204

The distinction between a “system call” and a “library routine” is an implementation detail that may differ between implementations and has thus been excluded from POSIX.1.

118205

118206

See “Interface, Not Implementation” in the Preface.

**118207 System Console**

118208 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/7 is applied, changing from “An  
118209 implementation-defined device” to “A device”.

**118210 System Databases**

118211 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/9 is applied, rewording the definition to  
118212 reference the existing definitions for “group database” and “user database”.

**118213 System Process**

118214 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/8 is applied, rewording the definition to  
118215 remove the requirement for an implementation to define the object.

**118216 System Reboot**

118217 A “system reboot” is an event initiated by an unspecified circumstance that causes all processes  
118218 (other than special system processes) to be terminated in an implementation-defined manner,  
118219 after which any changes to the state and contents of files created or written to by a Conforming  
118220 POSIX.1 Application prior to the event are implementation-defined.

118221 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/10 is applied, changing “An  
118222 implementation-defined sequence of events” to “An unspecified sequence of events”.

**118223 Synchronized I/O Data (and File) Integrity Completion**

118224 These terms specify that for synchronized read operations, pending writes must be successfully  
118225 completed before the read operation can complete. This is motivated by two circumstances.  
118226 Firstly, when synchronizing processes can access the same file, but not share common buffers  
118227 (such as for a remote file system), this requirement permits the reading process to guarantee that  
118228 it can read data written remotely. Secondly, having data written synchronously is insufficient to  
118229 guarantee the order with respect to a subsequent write by a reading process, and thus this extra  
118230 read semantic is necessary.

**118231 Text File**

118232 The term “text file” does not prevent the inclusion of control or other non-printable characters  
118233 (other than NUL). Therefore, standard utilities that list text files as inputs or outputs are either  
118234 able to process the special characters or they explicitly describe their limitations within their  
118235 individual descriptions. The definition of “text file” has caused controversy. The only difference  
118236 between text and binary files is that text files have lines of less than {LINE\_MAX} bytes, with no  
118237 NUL characters, each terminated by a <newline>. The definition allows a file with a single  
118238 <newline>, or a totally empty file, to be called a text file. If a file ends with an incomplete line it  
118239 is not strictly a text file by this definition. The <newline> referred to in POSIX.1-2008 is not some  
118240 generic line separator, but a single character; files created on systems where they use multiple  
118241 characters for ends of lines are not portable to all conforming systems without some translation  
118242 process unspecified by POSIX.1-2008.



**Thread**

POSIX.1-2008 defines a thread to be a flow of control within a process. Each thread has a minimal amount of private state; most of the state associated with a process is shared among all of the threads in the process. While most multi-thread extensions to POSIX have taken this approach, others have made different decisions.

**Note:** The choice to put threads within a process does not constrain implementations to implement threads in that manner. However, all functions have to behave as though threads share the indicated state information with the process from which they were created.

Threads need to share resources in order to cooperate. Memory has to be widely shared between threads in order for the threads to cooperate at a fine level of granularity. Threads keep data structures and the locks protecting those data structures in shared memory. For a data structure to be usefully shared between threads, such structures should not refer to any data that can only be interpreted meaningfully by a single thread. Thus, any system resources that might be referred to in data structures need to be shared between all threads. File descriptors, pathnames, and pointers to stack variables are all things that programmers want to share between their threads. Thus, the file descriptor table, the root directory, the current working directory, and the address space have to be shared.

Library implementations are possible as long as the effective behavior is as if system services invoked by one thread do not suspend other threads. This may be difficult for some library implementations on systems that do not provide asynchronous facilities.

See [Section B.2.9](#) (on page 3622) for additional rationale.

**Thread ID**

See [Section B.2.9.2](#) (on page 3639) for additional rationale.

**Thread-Safe Function**

All functions required by POSIX.1-2008 need to be thread-safe; see [Section A.4.17](#) (on page 3509) and [Section B.2.9.1](#) (on page 3636) for additional rationale.

**User Database**

There are no references in POSIX.1-2008 to a “passwd file” or a “group file”, and there is no requirement that the *group* or *passwd* databases be kept in files containing editable text. Many large timesharing systems use *passwd* databases that are hashed for speed. Certain security classifications prohibit certain information in the *passwd* database from being publicly readable.

The term “encoded” is used instead of “encrypted” in order to avoid the implementation connotations (such as reversibility or use of a particular algorithm) of the latter term.

The *getgrent()*, *setgrent()*, *endgrent()*, *getpwent()*, *setpwent()*, and *endpwent()* functions are not included as part of the base standard because they provide a linear database search capability that is not generally useful (the *getpwuid()*, *getpwnam()*, *getgrgid()*, and *getgrnam()* functions are provided for keyed lookup) and because in certain distributed systems, especially those with different authentication domains, it may not be possible or desirable to provide an application with the ability to browse the system databases indiscriminately. They are provided on XSI-conformant systems due to their historical usage by many existing applications.

A change from historical implementations is that the structures used by these functions have fields of the types **gid\_t** and **uid\_t**, which are required to be defined in the **<sys/types.h>** header. POSIX.1-2008 requires implementations to ensure that these types are defined by inclusion of **<grp.h>** and **<pwd.h>**, respectively, without imposing any name space pollution or errors from redefinition of types.

118288 POSIX.1-2008 is silent about the content of the strings containing user or group names. These  
 118289 could be digit strings. POSIX.1-2008 is also silent as to whether such digit strings bear any  
 118290 relationship to the corresponding (numeric) user or group ID.

118291 *Database Access*

118292 The thread-safe versions of the user and group database access functions return values in user-  
 118293 supplied buffers instead of possibly using static data areas that may be overwritten by each call.

118294 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/11 is applied, removing the words “of  
 118295 implementation-defined format”.

118296 **Virtual Processor\***

118297 The term “virtual processor” was chosen as a neutral term describing all kernel-level  
 118298 schedulable entities, such as processes, Mach tasks, or lightweight processes. Implementing  
 118299 threads using multiple processes as virtual processors, or implementing multiplexed threads  
 118300 above a virtual processor layer, should be possible, provided some mechanism has also been  
 118301 implemented for sharing state between processes or virtual processors. Many systems may also  
 118302 wish to provide implementations of threads on systems providing “shared processes” or  
 118303 “variable-weight processes”. It was felt that exposing such implementation details would  
 118304 severely limit the type of systems upon which the threads interface could be supported and  
 118305 prevent certain types of valid implementations. It was also determined that a virtual processor  
 118306 interface was out of the scope of the Rationale (Informative) volume of POSIX.1-2008.

118307 **XSI**

118308 This is included to allow POSIX.1-2008 to be adopted as an IEEE standard and an Open Group  
 118309 Standard, serving both the POSIX and the Single UNIX Specification in a core set of volumes.

118310 The term “XSI” has been used for 10 years in connection with the XPG series and the first and  
 118311 second versions of the base volumes of the Single UNIX Specification. The XSI margin code was  
 118312 introduced to denote the extended or more restrictive semantics beyond POSIX that are  
 118313 applicable to UNIX systems.

## 118314 **A.4 General Concepts**

118315 The general concepts are similar in nature to the definitions section, with the exception that a  
 118316 term defined in general concepts can contain normative requirements.

### 118317 **A.4.1 Concurrent Execution**

118318 There is no additional rationale provided for this section.



**118319 A.4.2 Directory Protection**

118320 There is no additional rationale provided for this section.

**118321 A.4.3 Extended Security Controls**

118322 Allowing an implementation to define extended security controls enables the use of  
 118323 POSIX.1-2008 in environments that require different or more rigorous security than that  
 118324 provided in POSIX.1. Extensions are allowed in two areas: privilege and file access permissions.  
 118325 The semantics of these areas have been defined to permit extensions with reasonable, but not  
 118326 exact, compatibility with all existing practices. For example, the elimination of the superuser  
 118327 definition precludes identifying a process as privileged or not by virtue of its effective user ID.

**118328 A.4.4 File Access Permissions**

118329 A process should not try to anticipate the result of an attempt to access data by *a priori* use of  
 118330 these rules. Rather, it should make the attempt to access data and examine the return value (and  
 118331 possibly *errno* as well), or use *access()*. An implementation may include other security  
 118332 mechanisms in addition to those specified in POSIX.1, and an access attempt may fail because of  
 118333 those additional mechanisms, even though it would succeed according to the rules given in this  
 118334 section. (For example, the user's security level might be lower than that of the object of the  
 118335 access attempt.) The supplementary group IDs provide another reason for a process to not  
 118336 attempt to anticipate the result of an access attempt.

118337 Since the current standard does not specify a method for opening a directory for searching, it is  
 118338 unspecified whether search permission on the *fd* argument to *openat()* and related functions is  
 118339 based on whether the directory was opened with search mode or on the current permissions  
 118340 allowed by the directory at the time a search is performed. When there is existing practice that  
 118341 supports opening directories for searching, it is expected that these functions will be modified to  
 118342 specify that the search permissions will be granted based on the file access modes of the  
 118343 directory's file descriptor identified by *fd*, and not on the mode of the directory at the time the  
 118344 directory is searched.

**118345 A.4.5 File Hierarchy**

118346 Though the file hierarchy is commonly regarded to be a tree, POSIX.1 does not define it as such  
 118347 for three reasons:

- 118348 1. Links may join branches.
- 118349 2. In some network implementations, there may be no single absolute root directory; see  
 118350 *pathname resolution*.
- 118351 3. With symbolic links, the file system need not be a tree or even a directed acyclic graph.

118352 **A.4.6 Filenames**

118353 Historically, certain filenames and pathnames have been reserved. This list includes **core**,  
118354 **/etc/passwd**, and so on. Conforming applications should avoid these.

118355 Most historical implementations prohibit case folding in filenames; that is, treating uppercase  
118356 and lowercase alphabetic characters as identical. However, some consider case folding desirable:

- 118357 • For user convenience
- 118358 • For ease-of-implementation of the POSIX.1 interface as a hosted system on some popular  
118359 operating systems

118360 Variants, such as maintaining case distinctions in filenames, but ignoring them in comparisons,  
118361 have been suggested. Methods of allowing escaped characters of the case opposite the default  
118362 have been proposed.

118363 Many reasons have been expressed for not allowing case folding, including:

- 118364 • No solid evidence has been produced as to whether case-sensitivity or case-insensitivity is  
118365 more convenient for users.
- 118366 • Making case-insensitivity a POSIX.1 implementation option would be worse than either  
118367 having it or not having it, because:
  - 118368 — More confusion would be caused among users.
  - 118369 — Application developers would have to account for both cases in their code.
  - 118370 — POSIX.1 implementors would still have other problems with native file systems, such  
118371 as short or otherwise constrained filenames or pathnames, and the lack of  
118372 hierarchical directory structure.
- 118373 • Case folding is not easily defined in many European languages, both because many of  
118374 them use characters outside the US ASCII alphabetic set, and because:
  - 118375 — In Spanish, the digraph "ll" is considered to be a single letter, the capitalized form  
118376 of which may be either "Ll" or "LL", depending on context.
  - 118377 — In French, the capitalized form of a letter with an accent may or may not retain the  
118378 accent, depending on the country in which it is written.
  - 118379 — In German, the sharp ess may be represented as a single character resembling a  
118380 Greek beta ( $\beta$ ) in lowercase, but as the digraph "SS" in uppercase.
  - 118381 — In Greek, there are several lowercase forms of some letters; the one to use depends on  
118382 its position in the word. Arabic has similar rules.
- 118383 • Many East Asian languages, including Japanese, Chinese, and Korean, do not distinguish  
118384 case and are sometimes encoded in character sets that use more than one byte per  
118385 character.
- 118386 • Multiple character codes may be used on the same machine simultaneously. There are  
118387 several ISO character sets for European alphabets. In Japan, several Japanese character  
118388 codes are commonly used together, sometimes even in filenames; this is evidently also the  
118389 case in China. To handle case insensitivity, the kernel would have to at least be able to  
118390 distinguish for which character sets the concept made sense.
- 118391 • The file system implementation historically deals only with bytes, not with characters.  
118392 Limitations on valid encodings ensure that the byte sequences for the <slash> character,  
118393 <period> character, and <NUL> character will not be confused with any other character in  
118394 any locale. However, there exist common single-shift encodings where other single-byte

118395 characters from the portable filename character set can also occur as a subset of a multi-  
 118396 byte character, making case folding of portable filename bytes dependent on the context of  
 118397 whether a shift-state is active.

118398 • The purpose of POSIX.1 is to standardize the common, existing definition, not to change it.  
 118399 Mandating case-insensitivity would make all historical implementations non-standard.

118400 • Not only the interface, but also application programs would need to change, counter to the  
 118401 purpose of having minimal changes to existing application code.

118402 • At least one of the original developers of the UNIX system has expressed objection in the  
 118403 strongest terms to either requiring case-insensitivity or making it an option, mostly on the  
 118404 basis that POSIX.1 should not hinder portability of application programs across related  
 118405 implementations in order to allow compatibility with unrelated operating systems.

118406 Two proposals were entertained regarding case folding in filenames:

118407 1. Remove all wording that previously permitted case folding.

118408 Rationale Case folding is inconsistent with the portable filename character set and  
 118409 filename definitions (all bytes except <slash> and null). No known  
 118410 implementations allowing all bytes except <slash> and null also do case  
 118411 folding.

118412 2. Change “though this practice is not recommended:” to “although this practice is strongly  
 118413 discouraged.”

118414 Rationale If case folding must be included in POSIX.1, the wording should be stronger  
 118415 to discourage the practice.

118416 The consensus selected the first proposal. Otherwise, a conforming application would have to  
 118417 assume that case folding would occur when it was not wanted, but that it would not occur when  
 118418 it was wanted.

#### 118419 **A.4.7 Filename Portability**

118420 Filenames should be constructed from the portable filename character set because the use of  
 118421 other characters can be confusing or ambiguous in certain contexts. (For example, the use of a  
 118422 <colon> (‘ : ’) in a pathname could cause ambiguity if that pathname were included in a *PATH*  
 118423 definition.)

118424 The constraint on use of the <hyphen-minus> character as the first character of a portable  
 118425 filename is a constraint on application behavior and not on implementations, since applications  
 118426 might not work as expected when such a filename is passed as a command line argument.

#### 118427 **A.4.8 File Times Update**

118428 This section reflects the actions of historical implementations. The times are not updated  
 118429 immediately, but are only marked for update by the functions. An implementation may update  
 118430 these times immediately.

118431 The accuracy of the time update values is intentionally left unspecified so that systems can  
 118432 control the bandwidth of a possible covert channel.

118433 The wording was carefully chosen to make it clear that there is no requirement that the  
 118434 conformance document contain information that might incidentally affect file timestamps. Any  
 118435 function that performs pathname resolution might update several last data access timestamps.

118436 Functions such as `getpwnam()` and `getgrnam()` might update the last data access timestamp of  
118437 some specific file or files. It is intended that these are not required to be documented in the  
118438 conformance document, but they should appear in the system documentation.

#### 118439 **A.4.9 Host and Network Byte Order**

118440 There is no additional rationale provided for this section.

#### 118441 **A.4.10 Measurement of Execution Time**

118442 The methods used to measure the execution time of processes and threads, and the precision of  
118443 these measurements, may vary considerably depending on the software architecture of the  
118444 implementation, and on the underlying hardware. Implementations can also make tradeoffs  
118445 between the scheduling overhead and the precision of the execution time measurements.  
118446 POSIX.1-2008 does not impose any requirement on the accuracy of the execution time; it instead  
118447 specifies that the measurement mechanism and its precision are implementation-defined.

#### 118448 **A.4.11 Memory Synchronization**

118449 In older multi-processors, access to memory by the processors was strictly multiplexed. This  
118450 meant that a processor executing program code interrogates or modifies memory in the order  
118451 specified by the code and that all the memory operation of all the processors in the system  
118452 appear to happen in some global order, though the operation histories of different processors are  
118453 interleaved arbitrarily. The memory operations of such machines are said to be sequentially  
118454 consistent. In this environment, threads can synchronize using ordinary memory operations. For  
118455 example, a producer thread and a consumer thread can synchronize access to a circular data  
118456 buffer as follows:

```
118457 int rdptr = 0;  
118458 int wrptr = 0;  
118459 data_t buf[BUFSIZE];  
  
118460 Thread 1:  
118461     while (work_to_do) {  
118462         int next;  
  
118463         buf[wrptr] = produce();  
118464         next = (wrptr + 1) % BUFSIZE;  
118465         while (rdptr == next)  
118466             ;  
118467         wrptr = next;  
118468     }  
  
118469 Thread 2:  
118470     while (work_to_do) {  
118471         while (rdptr == wrptr)  
118472             ;  
118473         consume(buf[rdptr]);  
118474         rdptr = (rdptr + 1) % BUFSIZE;  
118475     }
```

118476 In modern multi-processors, these conditions are relaxed to achieve greater performance. If one  
118477 processor stores values in location A and then location B, then other processors loading data

from location B and then location A may see the new value of B but the old value of A. The memory operations of such machines are said to be weakly ordered. On these machines, the circular buffer technique shown in the example will fail because the consumer may see the new value of *wrptr* but the old value of the data in the buffer. In such machines, synchronization can only be achieved through the use of special instructions that enforce an order on memory operations. Most high-level language compilers only generate ordinary memory operations to take advantage of the increased performance. They usually cannot determine when memory operation order is important and generate the special ordering instructions. Instead, they rely on the programmer to use synchronization primitives correctly to ensure that modifications to a location in memory are ordered with respect to modifications and/or access to the same location in other threads. Access to read-only data need not be synchronized. The resulting program is said to be data race-free.

Synchronization is still important even when accessing a single primitive variable (for example, an integer). On machines where the integer may not be aligned to the bus data width or be larger than the data width, a single memory load may require multiple memory cycles. This means that it may be possible for some parts of the integer to have an old value while other parts have a newer value. On some processor architectures this cannot happen, but portable programs cannot rely on this.

In summary, a portable multi-threaded program, or a multi-process program that shares writable memory between processes, has to use the synchronization primitives to synchronize data access. It cannot rely on modifications to memory being observed by other threads in the order written in the application or even on modification of a single variable being seen atomically.

Conforming applications may only use the functions listed to synchronize threads of control with respect to memory access. There are many other candidates for functions that might also be used. Examples are: signal sending and reception, or pipe writing and reading. In general, any function that allows one thread of control to wait for an action caused by another thread of control is a candidate. POSIX.1-2008 does not require these additional functions to synchronize memory access since this would imply the following:

- All these functions would have to be recognized by advanced compilation systems so that memory operations and calls to these functions are not reordered by optimization.
- All these functions would potentially have to have memory synchronization instructions added, depending on the particular machine.
- The additional functions complicate the model of how memory is synchronized and make automatic data race detection techniques impractical.

Formal definitions of the memory model were rejected as unreadable by the vast majority of programmers. In addition, most of the formal work in the literature has concentrated on the memory as provided by the hardware as opposed to the application programmer through the compiler and runtime system. It was believed that a simple statement intuitive to most programmers would be most effective. POSIX.1-2008 defines functions that can be used to synchronize access to memory, but it leaves open exactly how one relates those functions to the semantics of each function as specified elsewhere in POSIX.1-2008. POSIX.1-2008 also does not make a formal specification of the partial ordering in time that the functions can impose, as that is implied in the description of the semantics of each function. It simply states that the programmer has to ensure that modifications do not occur “simultaneously” with other access to a memory location.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/4 is applied, adding a new paragraph beneath the table of functions: “The *pthread\_once()* function shall synchronize memory for the first call in each thread for a given *pthread\_once\_t* object.”.

118527 **A.4.12 Pathname Resolution**

118528 It is necessary to differentiate between the definition of pathname and the concept of pathname  
118529 resolution with respect to the handling of trailing <slash> characters. By specifying the behavior  
118530 here, it is not possible to provide an implementation that is conforming but extends all interfaces  
118531 that handle pathnames to also handle strings that are not legal pathnames (because they have  
118532 trailing <slash> characters).

118533 Pathnames that end with one or more trailing <slash> characters must refer to directory paths.  
118534 Earlier versions of this standard were not specific about the distinction between trailing <slash>  
118535 characters on files and directories, and both were permitted.

118536 Two types of implementation have been prevalent; those that ignored trailing <slash> characters  
118537 on all pathnames regardless, and those that permitted them only on existing directories.

118538 An earlier version of this standard required that a pathname with a trailing <slash> character be  
118539 treated as if it had a trailing " / . " everywhere. This specification was ambiguous. In situations  
118540 where the intent was that the application wanted to require the implementation to accept the  
118541 pathname only if it named a directory (existing or to be created as a result of the call performing  
118542 pathname resolution), literally adding a ' . ' after the trailing <slash> could be interpreted to  
118543 require use of that pathname to fail. Some of the uses that created ambiguous requirements  
118544 included `mkdir("newdir/")` and `rmdir("existing-dir/")`. POSIX.1-2008 requires that a pathname  
118545 with a trailing <slash> be rejected unless it refers to a file that is a directory or to a file that is to  
118546 be created as a directory. The `rename()` function and the `mv` utility further specify that a trailing  
118547 <slash> cannot be used on a pathname naming a file that does not exist when used as the last  
118548 argument to `rename()` or `renameat()`, or as the last operand to `mv`.

118549 Note that this change does not break any conforming applications; since there were two different  
118550 types of implementation, no application could have portably depended on either behavior. This  
118551 change does however require some implementations to be altered to remain compliant.  
118552 Substantial discussion over a three-year period has shown that the benefits to application  
118553 developers outweighs the disadvantages for some vendors.

118554 On a historical note, some early applications automatically appended a '/' to every path.  
118555 Rather than fix the applications, the system implementation was modified to accept this  
118556 behavior by ignoring any trailing <slash>.

118557 Each directory has exactly one parent directory which is represented by the name **dot-dot** in the  
118558 first directory. No other directory, regardless of linkages established by symbolic links, is  
118559 considered the parent directory by POSIX.1-2008.

118560 There are two general categories of interfaces involving pathname resolution: those that follow  
118561 the symbolic link, and those that do not. There are several exceptions to this rule; for example,  
118562 `open(path, O_CREAT | O_EXCL)` will fail when `path` names a symbolic link. However, in all other  
118563 situations, the `open()` function will follow the link.

118564 What the filename **dot-dot** refers to relative to the root directory is implementation-defined. In  
118565 Version 7 it refers to the root directory itself; this is the behavior mentioned in POSIX.1-2008. In  
118566 some networked systems the construction `././hostname/` is used to refer to the root directory of  
118567 another host, and POSIX.1 permits this behavior.

118568 Other networked systems use the construct `//hostname` for the same purpose; that is, a double  
118569 initial <slash> is used. There is a potential problem with existing applications that create full  
118570 pathnames by taking a trunk and a relative pathname and making them into a single string  
118571 separated by ' / ', because they can accidentally create networked pathnames when the trunk is  
118572 ' / '. This practice is not prohibited because such applications can be made to conform by  
118573 simply changing to use " / / " as a separator instead of ' / ':



- If the trunk is `'/'`, the full pathname will begin with `"/"` (the initial `'/'` and the separator `"/"`). This is the same as `'/'`, which is what is desired. (This is the general case of making a relative pathname into an absolute one by prefixing with `"/"` instead of `'/'`.)
- If the trunk is `"/A"`, the result is `"/A/..."`; since non-leading sequences of two or more `<slash>` characters are treated as a single `<slash>`, this is equivalent to the desired `"/A/..."`.
- If the trunk is `"/A"`, the implementation-defined semantics will apply. (The multiple `<slash>` rule would apply.)

Application developers should avoid generating pathnames that start with `"/"`. Implementations are strongly encouraged to avoid using this special interpretation since a number of applications currently do not follow this practice and may inadvertently generate `"/..."`.

The term “root directory” is only defined in POSIX.1 relative to the process. In some implementations, there may be no absolute root directory. The initialization of the root directory of a process is implementation-defined.

When the standard says: “Pathname resolution for a given pathname shall yield the same results when used by any interface in POSIX.1-2008 as long as there are no changes to any files evaluated during pathname resolution for the given pathname between resolutions”, this applies to absolute pathnames or to relative pathnames from the same current working directory. Using the same relative pathname from two different working directories may yield different results.

Earlier versions of this standard were unclear as to whether a pathname was required to be a character string or just a string. This standard is now clear that filenames are just strings, and that pathname processing is locale-independent.

#### A.4.13 Process ID Reuse

There is no additional rationale provided for this section.

#### A.4.14 Scheduling Policy

There is no additional rationale provided for this section.

#### A.4.15 Seconds Since the Epoch

Coordinated Universal Time (UTC) includes leap seconds. However, in POSIX time (seconds since the Epoch), leap seconds are ignored (not applied) to provide an easy and compatible method of computing time differences. Broken-down POSIX time is therefore not necessarily UTC, despite its appearance.

As of December 2007, 23 leap seconds had been added to UTC since the Epoch, 1 January, 1970. Historically, one leap second is added every 15 months on average, so this offset can be expected to grow with time.

Most systems’ notion of “time” is that of a continuously increasing value, so this value should increase even during leap seconds. However, not only do most systems not keep track of leap seconds, but most systems are probably not synchronized to any standard time reference. Therefore, it is inappropriate to require that a time represented as seconds since the Epoch

precisely represent the number of seconds between the referenced time and the Epoch.

It is sufficient to require that applications be allowed to treat this time as if it represented the number of seconds between the referenced time and the Epoch. It is the responsibility of the vendor of the system, and the administrator of the system, to ensure that this value represents the number of seconds between the referenced time and the Epoch as closely as necessary for the application being run on that system.

It is important that the interpretation of time names and seconds since the Epoch values be consistent across conforming systems; that is, it is important that all conforming systems interpret “536 457 599 seconds since the Epoch” as 59 seconds, 59 minutes, 23 hours 31 December 1986, regardless of the accuracy of the system’s idea of the current time. The expression is given to ensure a consistent interpretation, not to attempt to specify the calendar. The relationship between *tm\_yday* and the day of week, day of month, and month is in accordance with the Gregorian calendar, and so is not specified in POSIX.1.

Consistent interpretation of seconds since the Epoch can be critical to certain types of distributed applications that rely on such timestamps to synchronize events. The accrual of leap seconds in a time standard is not predictable. The number of leap seconds since the Epoch will likely increase. POSIX.1 is more concerned about the synchronization of time between applications of astronomically short duration.

Note that *tm\_yday* is zero-based, not one-based, so the day number in the example above is 364. Note also that the division is an integer division (discarding remainder) as in the C language.

Note also that the meaning of *gmtime()*, *localtime()*, and *mktime()* is specified in terms of this expression. However, the ISO C standard computes *tm\_yday* from *tm\_mday*, *tm\_mon*, and *tm\_year* in *mktime()*. Because it is stated as a (bidirectional) relationship, not a function, and because the conversion between month-day-year and day-of-year dates is presumed well known and is also a relationship, this is not a problem.

Implementations that implement **time\_t** as a signed 32-bit integer will overflow in 2038. This standard requires that **time\_t** be an integer type with implementation-defined size, but does not mandate a particular size. The requirement that **time\_t** be integral is an additional constraint beyond the ISO C standard, which allows a real-floating **time\_t**. Implementation practice has shown that much existing code is unprepared to deal with a floating-point **time\_t**, and that use of **struct timespec** is a more uniform way to provide sub-second time manipulation within applications.

See also [Epoch](#) (on page 3482).

The topic of whether seconds since the Epoch should account for leap seconds has been debated on a number of occasions, and each time consensus was reached (with acknowledged dissent each time) that the majority of users are best served by treating all days identically. (That is, the majority of applications were judged to assume a single length—as measured in seconds since the Epoch—for all days. Thus, leap seconds are not applied to seconds since the Epoch.) Those applications which do care about leap seconds can determine how to handle them in whatever way those applications feel is best. This was particularly emphasized because there was disagreement about what the best way of handling leap seconds might be. It is a practical impossibility to mandate that a conforming implementation must have a fixed relationship to any particular official clock (consider isolated systems, or systems performing “reruns” by setting the clock to some arbitrary time).

Note that as a practical consequence of this, the length of a second as measured by some external standard is not specified. This unspecified second is nominally equal to an International System (SI) second in duration. Applications must be matched to a system that provides the particular handling of external time in the way required by the application.



118663 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/12 is applied, making an editorial  
118664 correction to the paragraph commencing “How any changes to the value of seconds ...”.

#### 118665 **A.4.16 Semaphore**

118666 There is no additional rationale provided for this section.

#### 118667 **A.4.17 Thread-Safety**

118668 Where the interface of a function required by POSIX.1-2008 precludes thread-safety, an alternate  
118669 thread-safe form is provided. The names of these thread-safe forms are the same as the non-  
118670 thread-safe forms with the addition of the suffix “\_r”. The suffix “\_r” is historical, where the  
118671 ‘r’ stood for “reentrant”.

118672 In some cases, thread-safety is provided by restricting the arguments to an existing function.

118673 See also [Section B.2.9.1](#) (on page 3636).

#### 118674 **A.4.18 Tracing**

118675 Refer to [Section B.2.11](#) (on page 3652).

#### 118676 **A.4.19 Treatment of Error Conditions for Mathematical Functions**

118677 It is intended that undeserved underflow and inexact floating-point exceptions are raised only if  
118678 avoiding them would be too costly.

#### 118679 **A.4.20 Treatment of NaN Arguments for Mathematical Functions**

118680 There is no additional rationale provided for this section.

#### 118681 **A.4.21 Utility**

118682 There is no additional rationale provided for this section.

#### 118683 **A.4.22 Variable Assignment**

118684 There is no additional rationale provided for this section.

## A.5 File Format Notation

The notation for spaces allows some flexibility for application output. Note that an empty character position in *format* represents one or more <blank> characters on the output (not *white space*, which can include <newline> characters). Therefore, another utility that reads that output as its input must be prepared to parse the data using *scanf()*, *awk*, and so on. The 'Δ' character is used when exactly one <space> is output.

The treatment of integers and spaces is different from the *printf()* function in that they can be surrounded with <blank> characters. This was done so that, given a format such as:

```
"%d\n" , <foo>
```

the implementation could use a *printf()* call such as:

```
printf("%6d\n" , foo);
```

and still conform. This notation is thus somewhat like *scanf()* in addition to *printf()*.

The *printf()* function was chosen as a model because most of the standard developers were familiar with it. One difference from the C function *printf()* is that the l and h conversion specifier characters are not used. As expressed by the Shell and Utilities volume of POSIX.1-2008, there is no differentiation between decimal values for type **int**, type **long**, or type **short**. The conversion specifications %d or %i should be interpreted as an arbitrary length sequence of digits. Also, no distinction is made between single precision and double precision numbers (**float** or **double** in C). These are simply referred to as floating-point numbers.

Many of the output descriptions in the Shell and Utilities volume of POSIX.1-2008 use the term "line", such as:

```
"%s" , <input line>
```

Since the definition of *line* includes the trailing <newline> already, there is no need to include a '\n' in the format; a double <newline> would otherwise result.

## A.6 Character Set

### A.6.1 Portable Character Set

The portable character set is listed in full so there is no dependency on the ISO/IEC 646: 1991 standard (or historically ASCII) encoded character set, although the set is identical to the characters defined in the International Reference version of the ISO/IEC 646: 1991 standard.

POSIX.1-2008 poses no requirement that multiple character sets or codesets be supported, leaving this as a marketing differentiation for implementors. Although multiple charmap files are supported, it is the responsibility of the implementation to provide the file(s); if only one is provided, only that one will be accessible using the *localedef -f* option.

The statement about invariance in codesets for the portable character set is worded to avoid precluding implementations where multiple incompatible codesets are available (for instance, ASCII and EBCDIC). The standard utilities cannot be expected to produce predictable results if they access portable characters that vary on the same implementation.

Not all character sets need include the portable character set, but each locale must include it. For

118723 example, a Japanese-based locale might be supported by a mixture of character sets: JIS X 0201  
 118724 Roman (a Japanese version of the ISO/IEC 646:1991 standard), JIS X 0208, and JIS X 0201  
 118725 Katakana. Not all of these character sets include the portable characters, but at least one does  
 118726 (JIS X 0201 Roman).

## 118727 **A.6.2 Character Encoding**

118728 Encoding mechanisms based on single shifts, such as the EUC encoding used in some Asian and  
 118729 other countries, can be supported via the current charmap mechanism. With single-shift  
 118730 encoding, each character is preceded by a shift code (SS2 or SS3). A complete EUC code,  
 118731 consisting of the portable character set (G0) and up to three additional character sets (G1, G2,  
 118732 G3), can be described using the current charmap mechanism; the encoding for each character in  
 118733 additional character sets G2 and G3 must then include their single-shift code. Other mechanisms  
 118734 to support locales based on encoding mechanisms such as locking shift are not addressed by this  
 118735 volume of POSIX.1-2008.

118736 The encodings for <slash> and <period> are required to be the same across all locales, in part  
 118737 because pathname resolution requires recognition of these bytes. It is a fortunate accident that all  
 118738 common shift-based encodings did not use either <slash> or <period> as a valid second byte in  
 118739 a multi-byte character.

118740 The encodings for <newline> and <carriage-return> are required to be the same across all +  
 118741 locales since they are special to the general terminal interface and cannot be changed (see XBD +  
 118742 [Section 11.1.9](#), on page 204). +

118743 Earlier versions of this standard did not state the requirement that the POSIX locale contains 256 +  
 118744 single-byte characters. This was an oversight; the intention was always that the POSIX locale +  
 118745 should have an 8-bit-clean single-byte encoding.

## 118746 **A.6.3 C Language Wide-Character Codes**

118747 The standard does not specify how wide characters are encoded or provide a method for  
 118748 defining wide characters in a charmap. It specifies ways of translating between wide characters  
 118749 and multi-byte characters. The standard does not prevent an extension from providing a method  
 118750 to define wide characters.

118751 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/13 is applied, adding a statement that the  
 118752 standard has no means of defining a wide-character codeset.

## 118753 **A.6.4 Character Set Description File**

118754 IEEE PASC Interpretation 1003.2 #196 is applied, removing three lines of text dealing with  
 118755 ranges of symbolic names using position constant values which had been erroneously included  
 118756 in the final IEEE P1003.2b draft standard.

118757 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/14 is applied, correcting the example and  
 118758 adding a statement that the standard provides no means of defining a wide-character codeset.

118759 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/15 is applied, allowing the value zero for  
 118760 the width value of **WIDTH** and **WIDTH\_DEFAULT**. This is required to cover some existing  
 118761 locales.

## 118762 A.6.4.1 State-Dependent Character Encodings

118763 A requirement was considered that would force utilities to eliminate any redundant locking  
 118764 shifts, but this was left as a quality of implementation issue.

118765 This change satisfies the following requirement from the ISO POSIX-2:1993 standard, Annex  
 118766 H.1:

118767 *The support of state-dependent (shift encoding) character sets should be addressed fully. See*  
 118768 *descriptions of these in XBD Section 6.2 (on page 128). If such character encodings are supported,*  
 118769 *it is expected that this will impact XBD Section 6.2 (on page 128), Chapter 7 (on page 135),*  
 118770 *Chapter 9 (on page 181), and the comm, cut, diff, grep, head, join, paste, and tail utilities.*

118771 The character set description file provides:

- 118772 • The capability to describe character set attributes (such as collation order or character  
 118773 classes) independent of character set encoding, and using only the characters in the  
 118774 portable character set. This makes it possible to create generic *localedef* source files for all  
 118775 codesets that share the portable character set (such as the ISO 8859 family or IBM Extended  
 118776 ASCII).
- 118777 • Standardized symbolic names for all characters in the portable character set, making it  
 118778 possible to refer to any such character regardless of encoding.

118779 Implementations are free to choose their own symbolic names, as long as the names identified  
 118780 by the Base Definitions volume of POSIX.1-2008 are also defined; this provides support for  
 118781 already existing “character names”.

118782 The names selected for the members of the portable character set follow the  
 118783 ISO/IEC 8859-1:1998 standard and the ISO/IEC 10646-1:2000 standard. However, several  
 118784 commonly used UNIX system names occur as synonyms in the list:

- 118785 • The historical UNIX system names are used for control characters.
- 118786 • The word “slash” is given in addition to “solidus”.
- 118787 • The word “backslash” is given in addition to “reverse-solidus”.
- 118788 • The word “hyphen” is given in addition to “hyphen-minus”.
- 118789 • The word “period” is given in addition to “full-stop”.
- 118790 • For digits, the word “digit” is eliminated.
- 118791 • For letters, the words “Latin Capital Letter” and “Latin Small Letter” are eliminated.
- 118792 • The words “left brace” and “right brace” are given in addition to “left-curly-bracket” and  
 118793 “right-curly-bracket”.
- 118794 • The names of the digits are preferred over the numbers to avoid possible confusion  
 118795 between ‘0’ and ‘O’, and between ‘1’ and ‘l’ (one and the letter ell).

118796 The names for the control characters in XBD Chapter 6 (on page 125) were taken from the  
 118797 ISO/IEC 4873:1991 standard.

118798 The charmap file was introduced to resolve problems with the portability of, especially, *localedef*  
 118799 sources. POSIX.1-2008 assumes that the portable character set is constant across all locales, but  
 118800 does not prohibit implementations from supporting two incompatible codings, such as both  
 118801 ASCII and EBCDIC. Such dual-support implementations should have all charmaps and *localedef*  
 118802 sources encoded using one portable character set, in effect cross-compiling for the other  
 118803 environment. Naturally, charmaps (and *localedef* sources) are only portable without  
 118804 transformation between systems using the same encodings for the portable character set. They

can, however, be transformed between two sets using only a subset of the actual characters (the portable character set). However, the particular coded character set used for an application or an implementation does not necessarily imply different characteristics or collation; on the contrary, these attributes should in many cases be identical, regardless of codeset. The charmap provides the capability to define a common locale definition for multiple codesets (the same *localedef* source can be used for codesets with different extended characters; the ability in the charmap to define empty names allows for characters missing in certain codesets).

The **<escape\_char>** declaration was added at the request of the international community to ease the creation of portable charmap files on terminals not implementing the default **<backslash>-escape**. The **<comment\_char>** declaration was added at the request of the international community to eliminate the potential confusion between the **<number-sign>** and the hash sign.

The octal number notation with no leading zero required was selected to match those of *awk* and *tr* and is consistent with that used by *localedef*. To avoid confusion between an octal constant and the back-references used in *localedef* source, the octal, hexadecimal, and decimal constants must contain at least two digits. As single-digit constants are relatively rare, this should not impose any significant hardship. Provision is made for more digits to account for systems in which the byte size is larger than 8 bits. For example, a Unicode (ISO/IEC 10646-1:2000 standard) system that has defined 16-bit bytes may require six octal, four hexadecimal, and five decimal digits.

The decimal notation is supported because some newer international standards define character values in decimal, rather than in the old column/row notation.

The charmap identifies the coded character sets supported by an implementation. At least one charmap must be provided, but no implementation is required to provide more than one. Likewise, implementations can allow users to generate new charmaps (for instance, for a new version of the ISO 8859 family of coded character sets), but does not have to do so. If users are allowed to create new charmaps, the system documentation describes the rules that apply (for instance, “only coded character sets that are supersets of the ISO/IEC 646:1991 standard IRV, no multi-byte characters”).

This addition of the **WIDTH** specification satisfies the following requirement from the ISO POSIX-2:1993 standard, Annex H.1:

- (9) *The definition of column position relies on the implementation's knowledge of the integral width of the characters. The charmap or LC\_CTYPE locale definitions should be enhanced to allow application specification of these widths.*

The character “width” information was first considered for inclusion under *LC\_CTYPE* but was moved because it is more closely associated with the information in the charmap than information in the locale source (cultural conventions information). Concerns were raised that formalizing this type of information is moving the locale source definition from the codeset-independent entity that it was designed to be to a repository of codeset-specific information. A similar issue occurred with the **<code\_set\_name>**, **<mb\_cur\_max>**, and **<mb\_cur\_min>** information, which was resolved to reside in the charmap definition.

The width definition was added to the IEEE P1003.2b draft standard with the intent that the *wcswidth()* and/or *wcwidth()* functions (currently specified in the System Interfaces volume of POSIX.1-2008) be the mechanism to retrieve the character width information.

118848 **A.7 Locale**118849 **A.7.1 General**

118850 The description of locales is based on work performed in the UniForum Technical Committee,  
 118851 Subcommittee on Internationalization. Wherever appropriate, keywords are taken from the  
 118852 ISO C standard or the X/Open Portability Guide.

118853 The value used to specify a locale with environment variables is the name specified as the *name*  
 118854 operand to the *localedef* utility when the locale was created. This provides a verifiable method to  
 118855 create and invoke a locale.

118856 The “object” definitions need not be portable, as long as “source” definitions are. Strictly  
 118857 speaking, source definitions are portable only between implementations using the same  
 118858 character set(s). Such source definitions, if they use symbolic names only, easily can be ported  
 118859 between systems using different codesets, as long as the characters in the portable character set  
 118860 (see XBD Section 6.1, on page 125) have common values between the codesets; this is frequently  
 118861 the case in historical implementations. Of source, this requires that the symbolic names used for  
 118862 characters outside the portable character set be identical between character sets. The definition  
 118863 of symbolic names for characters is outside the scope of POSIX.1-2008, but is certainly within the  
 118864 scope of other standards organizations.

118865 Applications can select the desired locale by invoking the *setlocale()* function (or equivalent)  
 118866 with the appropriate value. If the function is invoked with an empty string, the value of the  
 118867 corresponding environment variable is used. If the environment variable is not set or is set to the  
 118868 empty string, the implementation sets the appropriate environment as defined in XBD Chapter 8  
 118869 (on page 173).

118870 **A.7.2 POSIX Locale**

118871 On POSIX.1 implementations the POSIX locale is equal to the C locale, even though the |  
 118872 requirements for the POSIX locale are more extensive than the ISO C standard requirements for |  
 118873 the C locale. To avoid being classified as a C-language function, the name has been changed to |  
 118874 the POSIX locale; the environment variable value can be either "POSIX" or, for historical |  
 118875 reasons, "C".

118876 The POSIX definitions mirror the historical UNIX system behavior.

118877 The use of symbolic names for characters in the tables does not imply that the POSIX locale must  
 118878 be described using symbolic character names, but merely that it may be advantageous to do so.

118879 **A.7.3 Locale Definition**

118880 The decision to separate the file format from the *localedef* utility description was only partially  
 118881 editorial. Implementations may provide other interfaces than *localedef*. Requirements on “the  
 118882 utility”, mostly concerning error messages, are described in this way because they are meant to  
 118883 affect the other interfaces implementations may provide as well as *localedef*.

118884 The text about POSIX2\_LOCALEDEF does not mean that internationalization is optional; only  
 118885 that the functionality of the *localedef* utility is. REs, for instance, must still be able to recognize,  
 118886 for example, character class expressions such as "[[:alpha:]]". A possible analogy is with  
 118887 an applications development environment; while all conforming implementations must be



capable of executing applications, not all need to have the development environment installed. The assumption is that the capability to modify the behavior of utilities (and applications) via locale settings must be supported. If the *localedef* utility is not present, then the only choice is to select an existing (presumably implementation-documented) locale. An implementation could, for example, choose to support only the POSIX locale, which would in effect limit the amount of changes from historical implementations quite drastically. The *localedef* utility is still required, but would always terminate with an exit code indicating that no locale could be created. Supported locales must be documented using the syntax defined in this chapter. (This ensures that users can accurately determine what capabilities are provided. If the implementation decides to provide additional capabilities to the ones in this chapter, that is already provided for.)

If the option is present (that is, locales can be created), then the *localedef* utility must be capable of creating locales based on the syntax and rules defined in this chapter. This does not mean that the implementation cannot also provide alternate means for creating locales.

The octal, decimal, and hexadecimal notations are the same employed by the charmap facility (see XBD Section 6.4, on page 129). To avoid confusion between an octal constant and a back-reference, the octal, hexadecimal, and decimal constants must contain at least two digits. As single-digit constants are relatively rare, this should not impose any significant hardship. Provision is made for more digits to account for systems in which the byte size is larger than 8 bits. For example, a Unicode (see the ISO/IEC 10646-1:2000 standard) system that has defined 16-bit bytes may require six octal, four hexadecimal, and five decimal digits. As with the charmap file, multi-byte characters are described in the locale definition file using “big-endian” notation for reasons of portability. There is no requirement that the internal representation in the computer memory be in this same order.

One of the guidelines used for the development of this volume of POSIX.1-2008 is that characters outside the invariant part of the ISO/IEC 646:1991 standard should not be used in portable specifications. The <backslash> character is not in the invariant part; the <number-sign> is, but with multiple representations: as a <number-sign>, and as a hash sign. As far as general usage of these symbols, they are covered by the “grandfather clause”, but for newly defined interfaces, the WG15 POSIX working group has requested that POSIX provide alternate representations. Consequently, while the default escape character remains the <backslash> and the default comment character is the <number-sign>, implementations are required to recognize alternative representations, identified in the applicable source file via the <escape\_char> and <comment\_char> keywords.

#### A.7.3.1 LC\_CTYPE

The *LC\_CTYPE* category is primarily used to define the encoding-independent aspects of a character set, such as character classification. In addition, certain encoding-dependent characteristics are also defined for an application via the *LC\_CTYPE* category. POSIX.1-2008 does not mandate that the encoding used in the locale is the same as the one used by the application because an implementation may decide that it is advantageous to define locales in a system-wide encoding rather than having multiple, logically identical locales in different encodings, and to convert from the application encoding to the system-wide encoding on usage. Other implementations could require encoding-dependent locales.

In either case, the *LC\_CTYPE* attributes that are directly dependent on the encoding, such as <mb\_cur\_max> and the display width of characters, are not user-specifiable in a locale source and are consequently not defined as keywords.

Implementations may define additional keywords or extend the *LC\_CTYPE* mechanism to allow application-defined keywords.

The text “The ellipsis specification shall only be valid within a single encoded character set” is present because it is possible to have a locale supported by multiple character encodings, as explained in the rationale for XBD [Section 6.1](#) (on page 125). An example given there is of a possible Japanese-based locale supported by a mixture of the character sets JIS X 0201 Roman, JIS X 0208, and JIS X 0201 Katakana. Attempting to express a range of characters across these sets is not logical and the implementation is free to reject such attempts.

As the `LC_CTYPE` character classes are based on the ISO C standard character class definition, the category does not support multi-character elements. For instance, the German character <sharp-s> is traditionally classified as a lowercase letter. There is no corresponding uppercase letter; in proper capitalization of German text, the <sharp-s> will be replaced by "SS"; that is, by two characters. This kind of conversion is outside the scope of the **toupper** and **tolower** keywords.

Where POSIX.1-2008 specifies that only certain characters can be specified, as for the keywords **digit** and **xdigit**, the specified characters must be from the portable character set, as shown. As an example, only the Arabic digits 0 through 9 are acceptable as digits.

The character classes **digit**, **xdigit**, **lower**, **upper**, and **space** have a set of automatically included characters. These only need to be specified if the character values (that is, encoding) differs from the implementation default values. It is not possible to define a locale without these automatically included characters unless some implementation extension is used to prevent their inclusion. Such a definition would not be a proper superset of the C locale, and thus, it might not be possible for the standard utilities to be implemented as programs conforming to the ISO C standard.

The definition of character class **digit** requires that only ten characters—the ones defining digits—can be specified; alternate digits (for example, Hindi or Kanji) cannot be specified here. However, the encoding may vary if an implementation supports more than one encoding.

The definition of character class **xdigit** requires that the characters included in character class **digit** are included here also and allows for different symbols for the hexadecimal digits 10 through 15.

The inclusion of the **charclass** keyword satisfies the following requirement from the ISO POSIX-2: 1993 standard, Annex H.1:

(3) *The LC\_CTYPE (2.5.2.1) locale definition should be enhanced to allow user-specified additional character classes, similar in concept to the ISO C standard Multibyte Support Extension (MSE) `iswctype()` function.*

This keyword was previously included in The Open Group specifications and is now mandated in the Shell and Utilities volume of POSIX.1-2008.

The symbolic constant `{CHARCLASS_NAME_MAX}` was also adopted from The Open Group specifications. Applications portability is enhanced by the use of symbolic constants.

#### A.7.3.2 LC\_COLLATE

The rules governing collation depend to some extent on the use. At least five different levels of increasingly complex collation rules can be distinguished:

1. *Byte/machine code order*: This is the historical collation order in the UNIX system and many proprietary operating systems. Collation is here performed character by character, without any regard to context. The primary virtue is that it usually is quite fast and also completely deterministic; it works well when the native machine collation sequence matches the user expectations.



2. *Character order*: On this level, collation is also performed character by character, without regard to context. The order between characters is, however, not determined by the code values, but on the expectations by the user of the “correct” order between characters. In addition, such a (simple) collation order can specify that certain characters collate equally (for example, uppercase and lowercase letters).
3. *String ordering*: On this level, entire strings are compared based on relatively straightforward rules. Several “passes” may be required to determine the order between two strings. Characters may be ignored in some passes, but not in others; the strings may be compared in different directions; and simple string substitutions may be performed before strings are compared. This level is best described as “dictionary” ordering; it is based on the spelling, not the pronunciation, or meaning, of the words.
4. *Text search ordering*: This is a further refinement of the previous level, best described as “telephone book ordering”; some common homonyms (words spelled differently but with the same pronunciation) are collated together; numbers are collated as if they were spelled out, and so on.
5. *Semantic-level ordering*: Words and strings are collated based on their meaning; entire words (such as “the”) are eliminated; the ordering is not deterministic. This usually requires special software and is highly dependent on the intended use.

While the historical collation order formally is at level 1, for the English language it corresponds roughly to elements at level 2. The user expects to see the output from the *ls* utility sorted very much as it would be in a dictionary. While telephone book ordering would be an optimal goal for standard collation, this was ruled out as the order would be language-dependent. Furthermore, a requirement was that the order must be determined solely from the text string and the collation rules; no external information (for example, “pronunciation dictionaries”) could be required.

As a result, the goal for the collation support is at level 3. This also matches the requirements for the Canadian collation order, as well as other, known collation requirements for alphabetic scripts. It specifically rules out collation based on pronunciation rules or based on semantic analysis of the text.

The syntax for the *LC\_COLLATE* category source meets the requirements for level 3 and has been verified to produce the correct result with examples based on French, Canadian, and Danish collation order. Because it supports multi-character collating elements, it is also capable of supporting collation in codesets where a character is expressed using non-spacing characters followed by the base character (such as the ISO/IEC 6937: 2001 standard).

The directives that can be specified in an operand to the **order\_start** keyword are based on the requirements specified in several proposed standards and in customary use. The following is a rephrasing of rules defined for “lexical ordering in English and French” by the Canadian Standards Association (the text in square brackets is rephrased):

- Once special characters [punctuation] have been removed from original strings, the ordering is determined by scanning forwards (left to right) [disregarding case and diacriticals].
- In case of equivalence, special characters are once again removed from original strings and the ordering is determined by scanning backwards (starting from the rightmost character of the string and back), character by character [disregarding case but considering diacriticals].
- In case of repeated equivalence, special characters are removed again from original strings and the ordering is determined by scanning forwards, character by character [considering both case and diacriticals].

- If there is still an ordering equivalence after the first three rules have been applied, then only special characters and the position they occupy in the string are considered to determine ordering. The string that has a special character in the lowest position comes first. If two strings have a special character in the same position, the character [with the lowest collation value] comes first. In case of equality, the other special characters are considered until there is a difference or until all special characters have been exhausted.

It is estimated that this part of POSIX.1-2008 covers the requirements for all European languages, and no particular problems are anticipated with Slavic or Middle East character sets.

The Far East (particularly Japanese/Chinese) collations are often based on contextual information and pronunciation rules (the same ideogram can have different meanings and different pronunciations). Such collation, in general, falls outside the desired goal of POSIX.1-2008. There are, however, several other collation rules (stroke/radical or “most common pronunciation”) that can be supported with the mechanism described here.

The character order is defined by the order in which characters and elements are specified between the **order\_start** and **order\_end** keywords. Weights assigned to the characters and elements define the collation sequence; in the absence of weights, the character order is also the collation sequence.

The **position** keyword provides the capability to consider, in a compare, the relative position of characters not subject to **IGNORE**. As an example, consider the two strings “o-ring” and “or-ing”. Assuming the <hyphen-minus> is subject to **IGNORE** on the first pass, the two strings compare equal, and the position of the <hyphen-minus> is immaterial. On second pass, all characters except the <hyphen-minus> are subject to **IGNORE**, and in the normal case the two strings would again compare equal. By taking position into account, the first collates before the second.

This standard recommends (by the use of “should” in the normative text) that all implementation-provided locales define a collation sequence that has a total ordering of all characters unless the locale name has an ‘@’ modifier indicating that it has a special collation sequence. Defining locales in this way eliminates unexpected behavior when non-identical strings can collate equally (for example, `sort -u` and `sort | uniq` are not equivalent). The exception for locales with a suitable ‘@’ modifier in the name allows implementations to supply locales which do not have a total ordering of all characters provided that they draw attention to it in the modifier name. For example, `@icase` could indicate that each upper and lowercase character pair collates equally. Even with an ‘@’ modifier, total ordering is preferred when possible; for example, characters that are “ignored” in dictionary order need not be completely ignored (by using **IGNORE** for all collation weights), but can instead be given a unique weight after one or more **IGNORE** weights.

#### A.7.3.3 LC\_MONETARY

The currency symbol does not appear in `LC_MONETARY` because it is not defined in the C locale of the ISO C standard.

The ISO C standard limits the size of decimal points and thousands delimiters to single-byte values. In locales based on multi-byte coded character sets, this cannot be enforced; POSIX.1-2008 does not prohibit such characters, but makes the behavior unspecified (in the text “In contexts where other standards ...”).

The grouping specification is based on, but not identical to, the ISO C standard. The `-1` indicates that no further grouping is performed; the equivalent of `{CHAR_MAX}` in the ISO C standard.

The text “the value is not available in the locale” is taken from the ISO C standard and is used instead of the “unspecified” text in early proposals. There is no implication that omitting these

keywords or assigning them values of " " or -1 produces unspecified results; such omissions or assignments eliminate the effects described for the keyword or produce zero-length strings, as appropriate.

The locale definition is an extension of the ISO C standard *localeconv()* specification. In particular, rules on how **currency\_symbol** is treated are extended to also cover **int\_curr\_symbol**, and **p\_sep\_by\_space** and **n\_sep\_by\_space** have been augmented with the value 2, which places a <space> between the sign and the symbol. This has been updated to match the ISO/IEC 9899:1999 standard requirements and is an incompatible change from UNIX 98 and the ISO POSIX-2 standard and the ISO POSIX-1:1996 standard requirements. The following table shows the result of various combinations:

		<b>p_sep_by_space</b>		
		<b>2</b>	<b>1</b>	<b>0</b>
<b>p_cs_precedes = 1</b>	<b>p_sign_posn = 0</b>	(\$1.25)	(\$ 1.25)	(\$1.25)
	<b>p_sign_posn = 1</b>	+ \$1.25	+\$ 1.25	+\$1.25
	<b>p_sign_posn = 2</b>	\$1.25 +	\$ 1.25+	\$1.25+
	<b>p_sign_posn = 3</b>	+ \$1.25	+\$ 1.25	+\$1.25
	<b>p_sign_posn = 4</b>	\$ +1.25	\$+ 1.25	+\$1.25
<b>p_cs_precedes = 0</b>	<b>p_sign_posn = 0</b>	(1.25 \$)	(1.25 \$)	(1.25\$)
	<b>p_sign_posn = 1</b>	+1.25 \$	+1.25 \$	+1.25\$
	<b>p_sign_posn = 2</b>	1.25\$ +	1.25 \$+	1.25\$+
	<b>p_sign_posn = 3</b>	1.25+ \$	1.25 +\$	1.25+\$
	<b>p_sign_posn = 4</b>	1.25\$ +	1.25 \$+	1.25\$+

The following is an example of the interpretation of the **mon\_grouping** keyword. Assuming that the value to be formatted is 123 456 789 and the **mon\_thousands\_sep** is <apostrophe>, then the following table shows the result. The third column shows the equivalent string in the ISO C standard that would be used by the *localeconv()* function to accommodate this grouping.

<b>mon_grouping</b>	<b>Formatted Value</b>	<b>ISO C String</b>
3;-1	123456'789	"\3\177"
3	123'456'789	"\3"
3;2;-1	1234'56'789	"\3\2\177"
3;2	12'34'56'789	"\3\2"
-1	123456789	"\177"

In these examples, the octal value of {CHAR\_MAX} is 177.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/6 adds a correction that permits the Euro currency symbol and addresses extensibility. The correction is stated using the term "should" intentionally, in order to make this a recommendation rather than a restriction on implementations. This allows for flexibility in implementations on how they handle future currency symbol additions.

IEEE Std 1003.1-2001/Cor 1-2002, tem XBD/TC1/D6/5 is applied, adding the **int\_[np]\_\*** values to the POSIX locale definition of *LC\_MONETARY*.

IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/16 is applied, updating the descriptions of **p\_sep\_by\_space**, **n\_sep\_by\_space**, **int\_p\_sep\_by\_space**, and **int\_n\_sep\_by\_space** to match the description of these keywords in the ISO C standard and the System Interfaces volume of POSIX.1-2008, *localeconv()*.

## 119120 A.7.3.4 LC\_NUMERIC

119121 See the rationale for LC\_MONETARY for a description of the behavior of grouping.

## 119122 A.7.3.5 LC\_TIME

119123 Although certain of the conversion specifications in the POSIX locale (such as the name of the  
119124 month) are shown with initial capital letters, this need not be the case in other locales. Programs  
119125 using these conversion specifications may need to adjust the capitalization if the output is going  
119126 to be used at the beginning of a sentence.

119127 The LC\_TIME descriptions of **abday**, **day**, **mon**, and **abmon** imply a Gregorian style calendar  
119128 (7-day weeks, 12-month years, leap years, and so on). Formatting time strings for other types of  
119129 calendars is outside the scope of POSIX.1-2008.

119130 While the ISO 8601:2004 standard numbers the weekdays starting with Monday, historical  
119131 practice is to use the Sunday as the first day. Rather than change the order and introduce  
119132 potential confusion, the days must be specified beginning with Sunday; previous references to  
119133 "first day" have been removed. Note also that the Shell and Utilities volume of POSIX.1-2008  
119134 *date* utility supports numbering compliant with the ISO 8601:2004 standard.

119135 As specified under *date* in the Shell and Utilities volume of POSIX.1-2008 and *strftime()* in the  
119136 System Interfaces volume of POSIX.1-2008, the conversion specifications corresponding to the  
119137 optional keywords consist of a modifier followed by a traditional conversion specification (for  
119138 instance, %Ex). If the optional keywords are not supported by the implementation or are  
119139 unspecified for the current locale, these modified conversion specifications are treated as the  
119140 traditional conversion specifications. For example, assume the following keywords:

```
119141 alt_digits    "0th"; "1st"; "2nd"; "3rd"; "4th"; "5th"; \
119142              "6th"; "7th"; "8th"; "9th"; "10th"
```

```
119143 d_fmt        "The %Od day of %B in %Y"
```

119144 On July 4th 1776, the %x conversion specifications would result in "The 4th day of July  
119145 in 1776", while on July 14th 1789 it would result in "The 14 day of July in 1789". It  
119146 can be noted that the above example is for illustrative purposes only; the %O modifier is  
119147 primarily intended to provide for Kanji or Hindi digits in *date* formats.

119148 The following is an example for Japan that supports the current plus last three Emperors and  
119149 reverts to Western style numbering for years prior to the Meiji era. The example also allows for  
119150 the custom of using a special name for the first year of an era instead of using 1. (The examples  
119151 substitute romaji where kanji should be used.)

```
119152 era_d_fmt    "%EY%mgatsu%dnichi (%a)"
```

```
119153 era          "+:2:1990/01/01:+*:Heisei:%EC%Eynen"; \
119154             "+:1:1989/01/08:1989/12/31:Heisei:%ECgannen"; \
119155             "+:2:1927/01/01:1989/01/07:Shouwa:%EC%Eynen"; \
119156             "+:1:1926/12/25:1926/12/31:Shouwa:%ECgannen"; \
119157             "+:2:1913/01/01:1926/12/24:Taishou:%EC%Eynen"; \
119158             "+:1:1912/07/30:1912/12/31:Taishou:%ECgannen"; \
119159             "+:2:1869/01/01:1912/07/29:Meiji:%EC%Eynen"; \
119160             "+:1:1868/09/08:1868/12/31:Meiji:%ECgannen"; \
119161             "-:1868:1868/09/07:-*::%Ey"
```

119162 Assuming that the current date is September 21, 1991, a request to *date* or *strftime()* would yield  
119163 the following results:

```
119164 %Ec - Heisei3nen9gatsu2lnichi (Sat) 14:39:26
```

```

119165 %EC - Heisei
119166 %Ex - Heisei3nen9gatsu2lnichi (Sat)
119167 %Ey - 3
119168 %EY - Heisei3nen

119169 Example era definitions for the Republic of China:

119170 era      "+:2:1913/01/01:+*:ChungHwaMingGuo:%EC%EyNen";\
119171          "+:1:1912/1/1:1912/12/31:ChungHwaMingGuo:%ECYuenNen";\
119172          "+:1:1911/12/31:-*:MingChien:%EC%EyNen"

119173 Example definitions for the Christian Era:

119174 era      "+:1:0001/01/01:+*:AD:%EC %Ey";\
119175          "+:1:-0001/12/31:-*:BC:%Ey %EC"

```

#### 119176 A.7.3.6 LC\_MESSAGES

119177 The **yesstr** and **nostr** locale keywords and the YESSTR and NOSTR *langinfo* items were formerly  
 119178 used to match user affirmative and negative responses. In POSIX.1-2008, the **yesexpr**, **noexpr**,  
 119179 YESEXPR, and NOEXPR extended regular expressions have replaced them. Applications  
 119180 should use the general locale-based messaging facilities to issue prompting messages which  
 119181 include sample desired responses.

119182 Affirmative responses like:

```

119183 y
119184 Yes
119185 Yes!

```

119186 and negative responses like:

```

119187 N
119188 No
119189 Never
119190 No way!

```

119191 should all be recognized as affirmative and negative responses, respectively, by the EREs  
 119192 identified by the **yesexpr** and **noexpr** keywords for English language-based locales. There is no  
 119193 requirement that multi-line responses nor ambiguous responses like:

```

119194 no or yes
119195 yes or no
119196 maybe

```

119197 be correctly classified by either of these EREs. Application writers are encouraged to include  
 119198 locale-specific suggestions for affirmative and negative responses in prompts.

#### 119199 A.7.4 Locale Definition Grammar

119200 There is no additional rationale provided for this section.

119201 A.7.4.1 *Locale Lexical Conventions*

119202 There is no additional rationale provided for this section.

119203 A.7.4.2 *Locale Grammar*

119204 There is no additional rationale provided for this section.

## 119205 A.7.5 Locale Definition Example

119206 The following is an example of a locale definition file that could be used as input to the *localedf*  
119207 utility. It assumes that the utility is executed with the *-f* option, naming a charmap file with (at  
119208 least) the following content:

```
119209 CHARMAP
119210 <space>      \x20
119211 <dollar>     \x24
119212 <A>          \101
119213 <a>          \141
119214 <A-acute>    \346
119215 <a-acute>    \365
119216 <A-grave>    \300
119217 <a-grave>    \366
119218 <b>          \142
119219 <C>          \103
119220 <c>          \143
119221 <c-cedilla>  \347
119222 <d>          \x64
119223 <H>          \110
119224 <h>          \150
119225 <eszet>     \xb7
119226 <s>          \x73
119227 <z>          \x7a
119228 END CHARMAP
```

119229 It should not be taken as complete or to represent any actual locale, but only to illustrate the  
119230 syntax.

```
119231 #
119232 LC_CTYPE
119233 lower  <a>;<b>;<c>;<c-cedilla>;<d>;...;<z>
119234 upper  A;B;C;Ç;...;Z
119235 space  \x20;\x09;\x0a;\x0b;\x0c;\x0d
119236 blank  \040;\011
119237 toupper (<a>,<A>);(<b>,B);(<c>,C);(<ç>,Ç);(<d>,D);(<z>,Z)
119238 END LC_CTYPE
119239 #
119240 LC_COLLATE
119241 #
119242 # The following example of collation is based on
119243 # Canadian standard Z243.4.1-1998, "Canadian Alphanumeric
119244 # Ordering Standard for Character Sets of CSA Z234.4 Standard".
119245 # (Other parts of this example locale definition file do not
119246 # purport to relate to Canada, or to any other real culture.)
```



```

119247 # The proposed standard defines a 4-weight collation, such that
119248 # in the first pass, characters are compared without regard to
119249 # case or accents; in the second pass, backwards-compare without
119250 # regard to case; in the third pass, forwards-compare without
119251 # regard to diacriticals. In the 3 first passes, non-alphabetic
119252 # characters are ignored; in the fourth pass, only special
119253 # characters are considered, such that "The string that has a
119254 # special character in the lowest position comes first. If two
119255 # strings have a special character in the same position, the
119256 # collation value of the special character determines ordering.
119257 #
119258 # Only a subset of the character set is used here; mostly to
119259 # illustrate the set-up.
119260 #
119261 collating-symbol <NULL>
119262 collating-symbol <LOW_VALUE>
119263 collating-symbol <LOWER-CASE>
119264 collating-symbol <SUBSCRIPT-LOWER>
119265 collating-symbol <SUPERScript-LOWER>
119266 collating-symbol <UPPER-CASE>
119267 collating-symbol <NO-ACCENT>
119268 collating-symbol <PECULIAR>
119269 collating-symbol <LIGATURE>
119270 collating-symbol <ACUTE>
119271 collating-symbol <GRAVE>
119272 # Further collating-symbols follow.
119273 #
119274 # Properly, the standard does not include any multi-character
119275 # collating elements; the one below is added for completeness.
119276 #
119277 collating_element <ch> from "<c><h>"
119278 collating_element <CH> from "<C><H>"
119279 collating_element <Ch> from "<C><h>"
119280 #
119281 order_start forward;backward;forward;forward,position
119282 #
119283 # Collating symbols are specified first in the sequence to allocate
119284 # basic collation values to them, lower than that of any character.
119285 <NULL>
119286 <LOW_VALUE>
119287 <LOWER-CASE>
119288 <SUBSCRIPT-LOWER>
119289 <SUPERScript-LOWER>
119290 <UPPER-CASE>
119291 <NO-ACCENT>
119292 <PECULIAR>
119293 <LIGATURE>
119294 <ACUTE>
119295 <GRAVE>
119296 <RING-ABOVE>
119297 <DIAERESIS>
119298 <TILDE>
119299 # Further collating symbols are given a basic collating value here.

```

```

119300      #
119301      # Here follow special characters.
119302      <space>          IGNORE;IGNORE;IGNORE;<space>
119303      # Other special characters follow here.
119304      #
119305      # Here follow the regular characters.
119306      <a>               <a>;<NO-ACCENT>;<LOWER-CASE>;IGNORE
119307      <A>               <a>;<NO-ACCENT>;<UPPER-CASE>;IGNORE
119308      <a-acute>         <a>;<ACUTE>;<LOWER-CASE>;IGNORE
119309      <A-acute>         <a>;<ACUTE>;<UPPER-CASE>;IGNORE
119310      <a-grave>         <a>;<GRAVE>;<LOWER-CASE>;IGNORE
119311      <A-grave>        <a>;<GRAVE>;<UPPER-CASE>;IGNORE
119312      <ae>              " <a><e> " ; " <LIGATURE><LIGATURE> " ; \
119313                      " <LOWER-CASE><LOWER-CASE> " ; IGNORE
119314      <AE>              " <a><e> " ; " <LIGATURE><LIGATURE> " ; \
119315                      " <UPPER-CASE><UPPER-CASE> " ; IGNORE
119316      <b>               <b>;<NO-ACCENT>;<LOWER-CASE>;IGNORE
119317      <B>               <b>;<NO-ACCENT>;<UPPER-CASE>;IGNORE
119318      <c>               <c>;<NO-ACCENT>;<LOWER-CASE>;IGNORE
119319      <C>               <c>;<NO-ACCENT>;<UPPER-CASE>;IGNORE
119320      <ch>              <ch>;<NO-ACCENT>;<LOWER-CASE>;IGNORE
119321      <Ch>              <ch>;<NO-ACCENT>;<PECULIAR>;IGNORE
119322      <CH>              <ch>;<NO-ACCENT>;<UPPER-CASE>;IGNORE
119323      #
119324      # As an example, the strings "Bach" and "bach" could be encoded (for
119325      # compare purposes) as:
119326      # "Bach"          <b>;<a>;<ch>;<LOW_VALUE>;<NO-ACCENT>;<NO-ACCENT>;\
119327                      <NO-ACCENT>;<LOW_VALUE>;<UPPER-CASE>;<LOWER-CASE>;\
119328                      <LOWER-CASE>;<NULL>
119329      # "bach"          <b>;<a>;<ch>;<LOW_VALUE>;<NO-ACCENT>;<NO-ACCENT>;\
119330                      <NO-ACCENT>;<LOW_VALUE>;<LOWER-CASE>;<LOWER-CASE>;\
119331                      <LOWER-CASE>;<NULL>
119332      #
119333      # The two strings are equal in pass 1 and 2, but differ in pass 3.
119334      #
119335      # Further characters follow.
119336      #
119337      UNDEFINED          IGNORE;IGNORE;IGNORE;IGNORE
119338      #
119339      order_end
119340      #
119341      END LC_COLLATE
119342      #
119343      LC_MONETARY
119344      int_curr_symbol    "USD "
119345      currency_symbol    "$ "
119346      mon_decimal_point  "."
119347      mon_grouping       3;0
119348      positive_sign      ""
119349      negative_sign      "- "
119350      p_cs_precedes       1
119351      n_sign_posn         0
119352      END LC_MONETARY

```



```

119353      #
119354      LC_NUMERIC
119355      copy "US_en.ASCII"
119356      END LC_NUMERIC
119357      #
119358      LC_TIME
119359      abday  "Sun";"Mon";"Tue";"Wed";"Thu";"Fri";"Sat"
119360      #
119361      day    "Sunday";"Monday";"Tuesday";"Wednesday";\
119362           "Thursday";"Friday";"Saturday"
119363      #
119364      abmon  "Jan";"Feb";"Mar";"Apr";"May";"Jun";\
119365           "Jul";"Aug";"Sep";"Oct";"Nov";"Dec"
119366      #
119367      mon    "January";"February";"March";"April";\
119368           "May";"June";"July";"August";"September";\
119369           "October";"November";"December"
119370      #
119371      d_t_fmt "%a %b %d %T %Z %Y\n"
119372      END LC_TIME
119373      #
119374      LC_MESSAGES
119375      yesexpr "^[yY][[:alpha:]]*"|(OK)"
119376      #
119377      noexpr  "^[nN][[:alpha:]]*"
119378      END LC_MESSAGES

```

## 119379 A.8 Environment Variables

### 119380 A.8.1 Environment Variable Definition

119381 The variable *environ* is not intended to be declared in any header, but rather to be declared by  
 119382 the user for accessing the array of strings that is the environment. This is the traditional usage of  
 119383 the symbol. Putting it into a header could break some programs that use the symbol for their  
 119384 own purposes.

119385 The decision to restrict conforming systems to the use of digits, uppercase letters, and  
 119386 underscores for environment variable names allows applications to use lowercase letters in their  
 119387 environment variable names without conflicting with any conforming system.

119388 In addition to the obvious conflict with the shell syntax for positional parameter substitution,  
 119389 some historical applications (including some shells) exclude names with leading digits from the  
 119390 environment.

## 119391 **A.8.2 Internationalization Variables**

119392 Utilities conforming to the Shell and Utilities volume of POSIX.1-2008 and written in standard C  
119393 can access the locale variables by issuing the following call:

```
119394 setlocale(LC_ALL, " ")
```

119395 If this were omitted, the ISO C standard specifies that the C (or POSIX) locale would be used.

119396 The DESCRIPTION of *setlocale()* requires that when setting all categories of a locale, if the value  
119397 of any of the environment variable searches yields a locale that is not supported (and non-null),  
119398 the *setlocale()* function returns a null pointer and the global locale is unchanged.

119399 For the standard utilities, if any of the environment variables are invalid, it makes sense to  
119400 default to an implementation-defined, consistent locale environment. It is more confusing for a  
119401 user to have partial settings occur in case of a mistake. All utilities would then behave in one  
119402 language/cultural environment. Furthermore, it provides a way of forcing the whole  
119403 environment to be the implementation-defined default. Disastrous results could occur if a  
119404 pipeline of utilities partially uses the environment variables in different ways. In this case, it  
119405 would be appropriate for utilities that use *LANG* and related variables to exit with an error if  
119406 any of the variables are invalid. For example, users typing individual commands at a terminal  
119407 might want *date* to work if *LC\_MONETARY* is invalid as long as *LC\_TIME* is valid. Since these  
119408 are conflicting reasonable alternatives, POSIX.1-2008 leaves the results unspecified if the locale  
119409 environment variables would not produce a complete locale matching the specification of the  
119410 user.

119411 The locale settings of individual categories cannot be truly independent and still guarantee  
119412 correct results. For example, when collating two strings, characters must first be extracted from  
119413 each string (governed by *LC\_CTYPE*) before being mapped to collating elements (governed by  
119414 *LC\_COLLATE*) for comparison. That is, if *LC\_CTYPE* is causing parsing according to the rules of  
119415 a large, multi-byte code set (potentially returning 20 000 or more distinct character codeset  
119416 values), but *LC\_COLLATE* is set to handle only an 8-bit codeset with 256 distinct characters,  
119417 meaningful results are obviously impossible.

119418 The *LC\_MESSAGES* variable affects the language of messages generated by the standard  
119419 utilities.

119420 The description of the environment variable names starting with the characters “LC\_”  
119421 acknowledges the fact that the interfaces presented may be extended as new international  
119422 functionality is required. In the ISO C standard, names preceded by “LC\_” are reserved in the  
119423 name space for future categories.

119424 To avoid name clashes, new categories and environment variables are divided into two  
119425 classifications: “implementation-independent” and “implementation-defined”.

119426 Implementation-independent names will have the following format:

```
119427 LC_NAME
```

119428 where *NAME* is the name of the new category and environment variable. Capital letters must be  
119429 used for implementation-independent names.

119430 Implementation-defined names must be in lowercase letters, as below:

```
119431 LC_name
```

### A.8.3 Other Environment Variables

#### COLUMNS, LINES

The default values for the number of column positions, *COLUMNS*, and screen height, *LINES*, are unspecified because historical implementations use different methods to determine values corresponding to the size of the screen in which the utility is run. This size is typically known to the implementation through the value of *TERM*, or by more elaborate methods such as extensions to the *stty* utility or knowledge of how the user is dynamically resizing windows on a bit-mapped display terminal. Users should not need to set these variables in the environment unless there is a specific reason to override the default behavior of the implementation, such as to display data in an area arbitrarily smaller than the terminal or window. Values for these variables that are not decimal integers greater than zero are implicitly undefined values; it is unnecessary to enumerate all of the possible values outside of the acceptable set.

#### LOGNAME

In most implementations, the value of such a variable is easily forged, so security-critical applications should rely on other means of determining user identity. *LOGNAME* is required to be constructed from the portable filename character set for reasons of interchange. No diagnostic condition is specified for violating this rule, and no requirement for enforcement exists. The intent of the requirement is that if extended characters are used, the “guarantee” of portability implied by a standard is void.

#### PATH

Many historical implementations of the Bourne shell do not interpret a trailing <colon> to represent the current working directory and are thus non-conforming. The C Shell and the KornShell conform to POSIX.1-2008 on this point. The usual name of dot may also be used to refer to the current working directory.

Many implementations historically have used a default value of */bin* and */usr/bin* for the *PATH* variable. POSIX.1-2008 does not mandate this default path be identical to that retrieved from *getconf PATH* because it is likely that the standardized utilities may be provided in another directory separate from the directories used by some historical applications.

#### SHELL

The *SHELL* variable names the preferred shell of the user; it is a guide to applications. There is no direct requirement that that shell conform to POSIX.1-2008; that decision should rest with the user. It is the intention of the standard developers that alternative shells be permitted, if the user chooses to develop or acquire one. An operating system that builds its shell into the “kernel” in such a manner that alternative shells would be impossible does not conform to the spirit of POSIX.1-2008.

#### TZ

The quoted form of the timezone variable allows timezone names of the form UTC+1 (or any name that contains the <plus-sign> (‘+’), the <hyphen-minus> (‘-’), or digits), which may be appropriate for countries that do not have an official timezone name. It would be coded as <UTC+1>+1<UTC+2>, which would cause *std* to have a value of UTC+1 and *dst* a value of UTC+2, each with a length of 5 characters. This does not appear to conflict with any existing usage. The characters ‘<’ and ‘>’ were chosen for quoting because they are easier to parse visually than a quoting character that does not provide some sense of bracketing (and in a string like this, such bracketing is helpful). They were also chosen because they do not need special treatment when assigning to the *TZ* variable. Users are often confused by embedding quotes in a

string. Because '`<`' and '`>`' are meaningful to the shell, the whole string would have to be quoted, but that is easily explained. (Parentheses would have presented the same problems.) Although the '`>`' symbol could have been permitted in the string by either escaping it or doubling it, it seemed of little value to require that. This could be provided as an extension if there was a need. Timezone names of this new form lead to a requirement that the value of `{_POSIX_TZNAME_MAX}` change from 3 to 6.

Since the *TZ* environment variable is usually inherited by all applications started by a user after the value of the *TZ* environment variable is changed and since many applications run using the C or POSIX locale, using characters that are not in the portable character set in the *std* and *dst* fields could cause unexpected results.

The format of the *TZ* environment variable is changed in Issue 6 to allow for the quoted form, as defined in earlier versions of the ISO POSIX-1 standard.

IEEE Std 1003.1-2001/Cor 1-2002, item XBD/TC1/D6/7 is applied, adding the *ctime\_r()* and *localtime\_r()* functions to the list of functions that use the *TZ* environment variable.

## A.9 Regular Expressions

Rather than repeating the description of REs for each utility supporting REs, the standard developers preferred a common, comprehensive description of regular expressions in one place. The most common behavior is described here, and exceptions or extensions to this are documented for the respective utilities, as appropriate.

The BRE corresponds to the *ed* or historical *grep* type, and the ERE corresponds to the historical *egrep* type (now *grep -E*).

The text is based on the *ed* description and substantially modified, primarily to aid developers and others in the understanding of the capabilities and limitations of REs. Much of this was influenced by internationalization requirements.

It should be noted that the definitions in this section do not cover the *tr* utility; the *tr* syntax does not employ REs.

The specification of REs is particularly important to internationalization because pattern matching operations are very basic operations in business and other operations. The syntax and rules of REs are intended to be as intuitive as possible to make them easy to understand and use. The historical rules and behavior do not provide that capability to non-English language users, and do not provide the necessary support for commonly used characters and language constructs. It was necessary to provide extensions to the historical RE syntax and rules to accommodate other languages.

As they are limited to bracket expressions, the rationale for these modifications is in XBD [Section 9.3.5](#) (on page 184).

## 119512 **A.9.1 Regular Expression Definitions**

119513 It is possible to determine what strings correspond to subexpressions by recursively applying  
 119514 the leftmost longest rule to each subexpression, but only with the proviso that the overall match  
 119515 is leftmost longest. For example, matching "\(ac\*\\)c\*d[ac]\*\1" against *acdacaaa* matches  
 119516 *acdacaaa* (with \1=*a*); simply matching the longest match for "\(ac\*\\)" would yield \1=*ac*, but  
 119517 the overall match would be smaller (*acdac*). Conceptually, the implementation must examine  
 119518 every possible match and among those that yield the leftmost longest total matches, pick the one  
 119519 that does the longest match for the leftmost subexpression, and so on. Note that this means that  
 119520 matching by subexpressions is context-dependent: a subexpression within a larger RE may  
 119521 match a different string from the one it would match as an independent RE, and two instances of  
 119522 the same subexpression within the same larger RE may match different lengths even in similar  
 119523 sequences of characters. For example, in the ERE "(a.\*b)(a.\*b)", the two identical  
 119524 subexpressions would match four and six characters, respectively, of *accbaccccb*.

119525 The definition of single character has been expanded to include also collating elements  
 119526 consisting of two or more characters; this expansion is applicable only when a bracket  
 119527 expression is included in the BRE or ERE. An example of such a collating element may be the  
 119528 Dutch *ij*, which collates as a 'y'. In some encodings, a ligature "i with j" exists as a character  
 119529 and would represent a single-character collating element. In another encoding, no such ligature  
 119530 exists, and the two-character sequence *ij* is defined as a multi-character collating element.  
 119531 Outside brackets, the *ij* is treated as a two-character RE and matches the same characters in a  
 119532 string. Historically, a bracket expression only matched a single character. The ISO POSIX-2: 1993  
 119533 standard required bracket expressions like "[^[:lower:]]" to match multi-character collating  
 119534 elements such as "*ij*". However, this requirement led to behavior that many users did not  
 119535 expect and that could not feasibly be mimicked in user code, and it was rarely if ever  
 119536 implemented correctly. The current standard leaves it unspecified whether a bracket expression  
 119537 matches a multi-character collating element, allowing both historical and ISO POSIX-2: 1993  
 119538 standard implementations to conform.

119539 Also, in the current standard, it is unspecified whether character class expressions like  
 119540 "[[:lower:]]" can include multi-character collating elements like "*ij*"; hence  
 119541 "[[:lower:]]" can match "*ij*", and "[^[:lower:]]" can fail to match "*ij*". Common  
 119542 practice is for a character class expression to match a collating element if it matches the collating  
 119543 element's first character.

## 119544 **A.9.2 Regular Expression General Requirements**

119545 The definition of which sequence is matched when several are possible is based on the leftmost-  
 119546 longest rule historically used by deterministic recognizers. This rule is easier to define and  
 119547 describe, and arguably more useful, than the first-match rule historically used by non-  
 119548 deterministic recognizers. It is thought that dependencies on the choice of rule are rare; carefully  
 119549 contrived examples are needed to demonstrate the difference.

119550 A formal expression of the leftmost-longest rule is:

119551       The search is performed as if all possible suffixes of the string were tested for a prefix  
 119552       matching the pattern; the longest suffix containing a matching prefix is chosen, and the  
 119553       longest possible matching prefix of the chosen suffix is identified as the matching  
 119554       sequence.

119555 Historically, most RE implementations only match lines, not strings. However, that is more an  
 119556 effect of the usage than of an inherent feature of REs themselves. Consequently, POSIX.1-2008  
 119557 does not regard <newline> characters as special; they are ordinary characters, and both a

119558 <period> and a non-matching list can match them. Those utilities (like *grep*) that do not allow  
 119559 <newline> characters to match are responsible for eliminating any <newline> from strings  
 119560 before matching against the RE. The *regcomp()* function, however, can provide support for such  
 119561 processing without violating the rules of this section.

119562 Some implementations of *egrep* have had very limited flexibility in handling complex EREs.  
 119563 POSIX.1-2008 does not attempt to define the complexity of a BRE or ERE, but does place a lower  
 119564 limit on it—any RE must be handled, as long as it can be expressed in 256 bytes or less. (Of  
 119565 course, this does not place an upper limit on the implementation.) There are historical programs  
 119566 using a non-deterministic-recognizer implementation that should have no difficulty with this  
 119567 limit. It is possible that a good approach would be to attempt to use the faster, but more limited,  
 119568 deterministic recognizer for simple expressions and to fall back on the non-deterministic  
 119569 recognizer for those expressions requiring it. Non-deterministic implementations must be  
 119570 careful to observe the rules on which match is chosen; the longest match, not the first match,  
 119571 starting at a given character is used.

119572 The term “invalid” highlights a difference between this section and some others: POSIX.1-2008  
 119573 frequently avoids mandating of errors for syntax violations because they can be used by  
 119574 implementors to trigger extensions. However, the authors of the internationalization features of  
 119575 REs wanted to mandate errors for certain conditions to identify usage problems or non-portable  
 119576 constructs. These are identified within this rationale as appropriate. The remaining syntax  
 119577 violations have been left implicitly or explicitly undefined. For example, the BRE construct  
 119578 “\{1, 2, 3\}” does not comply with the grammar. A conforming application cannot rely on it  
 119579 producing an error nor matching the literal characters “\{1, 2, 3\}”.

119580 The term “undefined” was used in favor of “unspecified” because many of the situations are  
 119581 considered errors on some implementations, and the standard developers considered that  
 119582 consistency throughout the section was preferable to mixing undefined and unspecified.

### 119583 **A.9.3 Basic Regular Expressions**

119584 There is no additional rationale provided for this section.

#### 119585 *A.9.3.1 BREs Matching a Single Character or Collating Element*

119586 There is no additional rationale provided for this section.

#### 119587 *A.9.3.2 BRE Ordinary Characters*

119588 There is no additional rationale provided for this section.

#### 119589 *A.9.3.3 BRE Special Characters*

119590 There is no additional rationale provided for this section.

#### 119591 *A.9.3.4 Periods in BREs*

119592 There is no additional rationale provided for this section.



## 119593 A.9.3.5 RE Bracket Expression

119594 Range expressions are, historically, an integral part of REs. However, the requirements of  
 119595 “natural language behavior” and portability do conflict. In the POSIX locale, ranges must be  
 119596 treated according to the collating sequence and include such characters that fall within the range  
 119597 based on that collating sequence, regardless of character values. In other locales, ranges have  
 119598 unspecified behavior.

119599 Some historical implementations allow range expressions where the ending range point of one  
 119600 range is also the starting point of the next (for instance, "[a-m-o]"). This behavior should not  
 119601 be permitted, but to avoid breaking historical implementations, it is now *undefined* whether it is  
 119602 a valid expression and how it should be interpreted.

119603 Current practice in *awk* and *lex* is to accept escape sequences in bracket expressions as per XBD  
 119604 Table 5-1 (on page 121), while the normal ERE behavior is to regard such a sequence as  
 119605 consisting of two characters. Allowing the *awk/lex* behavior in EREs would change the normal  
 119606 behavior in an unacceptable way; it is expected that *awk* and *lex* will decode escape sequences in  
 119607 EREs before passing them to *regcomp()* or comparable routines. Each utility describes the escape  
 119608 sequences it accepts as an exception to the rules in this section; the list is not the same, for  
 119609 historical reasons.

119610 As noted previously, the new syntax and rules have been added to accommodate other  
 119611 languages than English. The remainder of this section describes the rationale for these  
 119612 modifications.

119613 In the POSIX locale, a regular expression that starts with a range expression matches a set of  
 119614 strings that are contiguously sorted, but this is not necessarily true in other locales. For example,  
 119615 a French locale might have the following behavior:

```
119616 $ ls
119617 alpha  Alpha  estimé  ESTIMÉ  été  eurêka
119618 $ ls [a-e]*
119619 alpha  Alpha  estimé  eurêka
```

119620 Such disagreements between matching and contiguous sorting are unavoidable because POSIX  
 119621 sorting cannot be implemented in terms of a deterministic finite-state automaton (DFA), but  
 119622 range expressions by design are implementable in terms of DFAs.

119623 Historical implementations used native character order to interpret range expressions. The  
 119624 ISO POSIX-2:1993 standard instead required collating element order (CEO): the order that  
 119625 collating elements were specified between the **order\_start** and **order\_end** keywords in the  
 119626 *LC\_COLLATE* category of the current locale. CEO had some advantages in portability over the  
 119627 native character order, but it also had some disadvantages:

- 119628 • CEO could not feasibly be mimicked in user code, leading to inconsistencies between  
 119629 POSIX matchers and matchers in popular user programs like Emacs, *ksh*, and Perl.
- 119630 • CEO caused range expressions to match accented and capitalized letters contrary to many  
 119631 users' expectations. For example, "[a-e]" typically matched both 'E' and 'á' but  
 119632 neither 'A' nor 'é'.
- 119633 • CEO was not consistent across implementations. In practice, CEO was often less portable  
 119634 than native character order. For example, it was common for the CEOs of two  
 119635 implementation-supplied locales to disagree, even if both locales were named "da\_DK".

119636 Because of these problems, some implementations of regular expressions continued to use native  
 119637 character order. Others used the collation sequence, which is more consistent with sorting than  
 119638 either CEO or native order, but which departs further from the traditional POSIX semantics  
 119639 because it generally requires "[a-e]" to match either 'A' or 'E' but not both. As a result of

119640 this kind of implementation variation, programmers who wanted to write portable regular  
 119641 expressions could not rely on the ISO POSIX-2: 1993 standard guarantees in practice.

119642 While revising the standard, lengthy consideration was given to proposals to attack this problem  
 119643 by adding an API for querying the CEO to allow user-mode matchers, but none of these  
 119644 proposals had implementation experience and none achieved consensus. Leaving the standard  
 119645 alone was also considered, but rejected due to the problems described above.

119646 The current standard leaves unspecified the behavior of a range expression outside the POSIX  
 119647 locale. This makes it clearer that conforming applications should avoid range expressions  
 119648 outside the POSIX locale, and it allows implementations and compatible user-mode matchers to  
 119649 interpret range expressions using native order, CEO, collation sequence, or other, more  
 119650 advanced techniques. The concerns which led to this change were raised in IEEE PASC  
 119651 interpretation 1003.2 #43 and others, and related to ambiguities in the specification of how  
 119652 multi-character collating elements should be handled in range expressions. These ambiguities  
 119653 had led to multiple interpretations of the specification, in conflicting ways, which led to varying  
 119654 implementations. As noted above, efforts were made to resolve the differences, but no solution  
 119655 has been found that would be specific enough to allow for portable software while not  
 119656 invalidating existing implementations.

119657 The standard developers recognize that collating elements are important, such elements being  
 119658 common in several European languages; for example, 'ch' or 'll' in traditional Spanish;  
 119659 'aa' in several Scandinavian languages. Existing internationalized implementations have  
 119660 processed, and continue to process, these elements in range expressions. Efforts are expected to  
 119661 continue in the future to find a way to define the behavior of these elements precisely and  
 119662 portably.

119663 The ISO POSIX-2: 1993 standard required "[b-a]" to be an invalid expression in the POSIX  
 119664 locale, but this requirement has been relaxed in this version of the standard so that "[b-a]" can  
 119665 instead be treated as a valid expression that does not match any string.

119666 The standard specifies three possible behaviors for regular expressions such as "[ :alpha: ]".  
 119667 One behavior is the traditional implementation, which behaves like "[ :a-hlp]". Another, for  
 119668 alignment with the *tr* utility, is to treat it like "[ [ :alpha: ] ]". And finally, the standard allows  
 119669 rejecting the regular expression as invalid, as a means of alerting a user to the non-portable  
 119670 aspect of that regular expression. The set of regular expressions with this undefined behavior is  
 119671 limited solely to the expressions where the outer '[' and ']' of the bracket expression can be  
 119672 confused with the missing bracket pair '[' and ']' necessary to form a collating symbol,  
 119673 equivalence class, or character class; thus "[\_ :alpha:]" or "[::]" do not trigger the  
 119674 unspecified behavior.

#### 119675 A.9.3.6 BREs Matching Multiple Characters

119676 The limit of nine back-references to subexpressions in the RE is based on the use of a single-digit  
 119677 identifier; increasing this to multiple digits would break historical applications. This does not  
 119678 imply that only nine subexpressions are allowed in REs. The following is a valid BRE with ten  
 119679 subexpressions:

119680 `\(\(\(ab\) *c\) *d\)\(ef\) *\(\(gh\)\{2\}\(ij\) *\(\(kl\) *\(\(mn\) *\(\(op\) *\(\(qr\) *`

119681 The standard developers regarded the common historical behavior, which supported "\n\*", but  
 119682 not "\n{min,max}", "\(\...\)\*", or "\(\...\)\{min,max\}", as a non-intentional  
 119683 result of a specific implementation, and they supported both duplication and interval  
 119684 expressions following subexpressions and back-references.

119685 The changes to the processing of the back-reference expression remove an unspecified or  
 119686 ambiguous behavior in the Shell and Utilities volume of POSIX.1-2008, aligning it with the



119687 requirements specified for the *regcomp()* expression, and is the result of PASC Interpretation  
119688 1003.2-92 #43 submitted for the ISO POSIX-2: 1993 standard.

#### 119689 A.9.3.7 BRE Precedence

119690 There is no additional rationale provided for this section.

#### 119691 A.9.3.8 BRE Expression Anchoring

119692 Often, the <dollar-sign> is viewed as matching the ending <newline> in text files. This is not  
119693 strictly true; the <newline> is typically eliminated from the strings to be matched, and the  
119694 <dollar-sign> matches the terminating null character.

119695 The ability of ' ^ ', ' \$ ', and ' \* ' to be non-special in certain circumstances may be confusing to  
119696 some programmers, but this situation was changed only in a minor way from historical practice  
119697 to avoid breaking many historical scripts. Some consideration was given to making the use of  
119698 the anchoring characters undefined if not escaped and not at the beginning or end of strings.  
119699 This would cause a number of historical BREs, such as " 2 ^ 10 ", " \$HOME ", and " \$1 . 35 ", that  
119700 relied on the characters being treated literally, to become invalid.

119701 However, one relatively uncommon case was changed to allow an extension used on some  
119702 implementations. Historically, the BREs " ^foo " and " \ ( ^foo \ ) " did not match the same  
119703 string, despite the general rule that subexpressions and entire BREs match the same strings. To  
119704 increase consensus, POSIX.1-2008 has allowed an extension on some implementations to treat  
119705 these two cases in the same way by declaring that anchoring *may* occur at the beginning or end  
119706 of a subexpression. Therefore, portable BREs that require a literal <circumflex> at the beginning  
119707 or a <dollar-sign> at the end of a subexpression must escape them. Note that a BRE such as  
119708 " a \ ( ^bc \ ) " will either match " a ^bc " or nothing on different systems under the rules.

119709 ERE anchoring has been different from BRE anchoring in all historical systems. An unescaped  
119710 anchor character has never matched its literal counterpart outside a bracket expression. Some  
119711 implementations treated " foo\$bar " as a valid expression that never matched anything; others  
119712 treated it as invalid. POSIX.1-2008 mandates the former, valid unmatched behavior.

119713 Some implementations have extended the BRE syntax to add alternation. For example, the  
119714 subexpression " \ ( foo\$ | bar \ ) " would match either " foo " at the end of the string or " bar "  
119715 anywhere. The extension is triggered by the use of the undefined " \ | " sequence. Because the  
119716 BRE is undefined for portable scripts, the extending system is free to make other assumptions,  
119717 such that the ' \$ ' represents the end-of-line anchor in the middle of a subexpression. If it were  
119718 not for the extension, the ' \$ ' would match a literal <dollar-sign> under the rules.

### 119719 A.9.4 Extended Regular Expressions

119720 As with BREs, the standard developers decided to make the interpretation of escaped ordinary  
119721 characters undefined.

119722 The <right-parenthesis> is not listed as an ERE special character because it is only special in the  
119723 context of a preceding <left-parenthesis>. If found without a preceding <left-parenthesis>, the  
119724 <right-parenthesis> has no special meaning.

119725 The interval expression, " { m , n } ", has been added to EREs. Historically, the interval expression  
119726 has only been supported in some ERE implementations. The standard developers estimated that  
119727 the addition of interval expressions to EREs would not decrease consensus and would also make  
119728 BREs more of a subset of EREs than in many historical implementations.

- 119729 It was suggested that, in addition to interval expressions, back-references ( '\n' ) should also be  
 119730 added to EREs. This was rejected by the standard developers as likely to decrease consensus.
- 119731 In historical implementations, multiple duplication symbols are usually interpreted from left to  
 119732 right and treated as additive. As an example, "a+\*b" matches zero or more instances of 'a'  
 119733 followed by a 'b'. In POSIX.1-2008, multiple duplication symbols are undefined; that is, they  
 119734 cannot be relied upon for conforming applications. One reason for this is to provide some scope  
 119735 for future enhancements.
- 119736 The precedence of operations differs between EREs and those in *lex*; in *lex*, for historical reasons,  
 119737 interval expressions have a lower precedence than concatenation.
- 119738 A.9.4.1 *EREs Matching a Single Character or Collating Element*  
 119739 There is no additional rationale provided for this section.
- 119740 A.9.4.2 *ERE Ordinary Characters*  
 119741 There is no additional rationale provided for this section.
- 119742 A.9.4.3 *ERE Special Characters*  
 119743 There is no additional rationale provided for this section.
- 119744 A.9.4.4 *Periods in EREs*  
 119745 There is no additional rationale provided for this section.
- 119746 A.9.4.5 *ERE Bracket Expression*  
 119747 There is no additional rationale provided for this section.
- 119748 A.9.4.6 *EREs Matching Multiple Characters*  
 119749 There is no additional rationale provided for this section.
- 119750 A.9.4.7 *ERE Alternation*  
 119751 There is no additional rationale provided for this section.
- 119752 A.9.4.8 *ERE Precedence*  
 119753 There is no additional rationale provided for this section.
- 119754 A.9.4.9 *ERE Expression Anchoring*  
 119755 There is no additional rationale provided for this section.

119756 **A.9.5 Regular Expression Grammar**

119757 The grammars are intended to represent the range of acceptable syntaxes available to  
 119758 conforming applications. There are instances in the text where undefined constructs are  
 119759 described; as explained previously, these allow implementation extensions. There is no intended  
 119760 requirement that an implementation extension must somehow fit into the grammars shown  
 119761 here.

119762 The BRE grammar does not permit `L_ANCHOR` or `R_ANCHOR` inside `"\("` and `"\)"` (which  
 119763 implies that `'^'` and `'$'` are ordinary characters). This reflects the semantic limits on the  
 119764 application, as noted in XBD [Section 9.3.8](#) (on page 188). Implementations are permitted to  
 119765 extend the language to interpret `'^'` and `'$'` as anchors in these locations, and as such,  
 119766 conforming applications cannot use unescaped `'^'` and `'$'` in positions inside `"\("` and `"\)"`  
 119767 that might be interpreted as anchors.

119768 The ERE grammar does not permit several constructs that XBD [Section 9.4.2](#) (on page 188) and  
 119769 [Section 9.4.3](#) (on page 189) specify as having undefined results:

- 119770 • `ORD_CHAR` preceded by `<backslash>`
- 119771 • `ERE_dupl_symbol(s)` appearing first in an ERE, or immediately following `'|'`, `'^'`, or `'('`
- 119772 • `'{'` not part of a valid `ERE_dupl_symbol`
- 119773 • `'|'` appearing first or last in an ERE, or immediately following `'|'` or `'('`, or  
 119774 immediately preceding `')`

119775 Implementations are permitted to extend the language to allow these. Conforming applications  
 119776 cannot use such constructs.

119777 *A.9.5.1 BRE/ERE Grammar Lexical Conventions*

119778 There is no additional rationale provided for this section.

119779 *A.9.5.2 RE and Bracket Expression Grammar*

119780 The removal of the `Back_open_paren Back_close_paren` option from the `nondupl_RE` specification is  
 119781 the result of PASC Interpretation 1003.2-92 #43 submitted for the ISO POSIX-2:1993 standard.  
 119782 Although the grammar required support for null subexpressions, this section does not describe  
 119783 the meaning of, and historical practice did not support, this construct.

119784 *A.9.5.3 ERE Grammar*

119785 There is no additional rationale provided for this section.

## 119786 A.10 Directory Structure and Devices

### 119787 A.10.1 Directory Structure and Files

119788 A description of the historical `/usr/tmp` was omitted, removing any concept of differences in  
119789 emphasis between the `/` and `/usr` directories. The descriptions of `/bin`, `/usr/bin`, `/lib`, and `/usr/lib`  
119790 were omitted because they are not useful for applications. In an early draft, a distinction was  
119791 made between system and application directory usage, but this was not found to be useful.

119792 The directories `/` and `/dev` are included because the notion of a hierarchical directory structure is  
119793 key to other information presented elsewhere in POSIX.1-2008. In early drafts, it was argued that  
119794 special devices and temporary files could conceivably be handled without a directory structure  
119795 on some implementations. For example, the system could treat the characters `" /tmp "` as a  
119796 special token that would store files using some non-POSIX file system structure. This notion was  
119797 rejected by the standard developers, who required that all the files in this section be  
119798 implemented via POSIX file systems.

119799 The `/tmp` directory is retained in POSIX.1-2008 to accommodate historical applications that  
119800 assume its availability. Implementations are encouraged to provide suitable directory names in  
119801 the environment variable `TMPDIR` and applications are encouraged to use the contents of  
119802 `TMPDIR` for creating temporary files.

119803 The standard files `/dev/null` and `/dev/tty` are required to be both readable and writable to allow  
119804 applications to have the intended historical access to these files.

119805 The standard file `/dev/console` has been added for alignment with the Single UNIX  
119806 Specification.

### 119807 A.10.2 Output Devices and Terminal Types

119808 IEEE Std 1003.1-2001/Cor 2-2004, item XBD/TC2/D6/17 is applied, making it clear that the  
119809 requirements for documenting terminal support are in the system documentation.

## 119810 A.11 General Terminal Interface

119811 If the implementation does not support this interface on any device types, it should behave as if  
119812 it were being used on a device that is not a terminal device (in most cases `errno` will be set to  
119813 `[ENOTTY]` on return from functions defined by this interface). This is based on the fact that  
119814 many applications are written to run both interactively and in some non-interactive mode, and  
119815 they adapt themselves at runtime. Requiring that they all be modified to test an environment  
119816 variable to determine whether they should try to adapt is unnecessary. On a system that  
119817 provides no general terminal interface, providing all the entry points as stubs that return  
119818 `[ENOTTY]` (or an equivalent, as appropriate) has the same effect and requires no changes to the  
119819 application.

119820 Although the needs of both interface implementors and application developers were addressed  
119821 throughout POSIX.1-2008, this section pays more attention to the needs of the latter. This is  
119822 because, while many aspects of the programming interface can be hidden from the user by the  
119823 application developer, the terminal interface is usually a large part of the user interface.  
119824 Although to some extent the application developer can build missing features or work around

inappropriate ones, the difficulties of doing that are greater in the terminal interface than elsewhere. For example, efficiency prohibits the average program from interpreting every character passing through it in order to simulate character erase, line kill, and so on. These functions should usually be done by the operating system, possibly at the interrupt level.

The *tc\**() functions were introduced as a way of avoiding the problems inherent in the traditional *ioctl*() function and in variants of it that were proposed. For example, *tcsetattr*() is specified in place of the use of the TCSETA *ioctl*() command function. This allows specification of all the arguments in a manner consistent with the ISO C standard unlike the varying third argument of *ioctl*(), which is sometimes a pointer (to any of many different types) and sometimes an **int**.

The advantages of this new method include:

- It allows strict type checking.
- The direction of transfer of control data is explicit.
- Portable capabilities are clearly identified.
- The need for a general interface routine is avoided.
- Size of the argument is well-defined (there is only one type).

The disadvantages include:

- No historical implementation used the new method.
- There are many small routines instead of one general-purpose one.
- The historical parallel with *fcntl*() is broken.

The issue of modem control was excluded from POSIX.1-2008 on the grounds that:

- It was concerned with setting and control of hardware timers.
- The appropriate timers and settings vary widely internationally.
- Feedback from European computer manufacturers indicated that this facility was not consistent with European needs and that specification of such a facility was not a requirement for portability.

## A.11.1 Interface Characteristics

### A.11.1.1 Opening a Terminal Device File

The **O\_TTY\_INIT** flag for *open*() has been added to POSIX.1-2008 to solve a problem encountered by applications written for earlier versions of this standard which need to open a modem or similar device and initialize all of the parameter settings. Using the *tcgetattr*()-*modify-tcsetattr*() method mandated by the standard could result in non-conforming behavior if the device had previously been used with non-conforming parameter settings, on implementations which do not reset the parameter settings in between the last close of the device by one application and the first open by another application. To avoid this problem, some application developers were resorting to using *memset*() to zero the **termios** structure before setting all of the standard parameters, but this risks non-conforming behavior on systems where some non-standard parameter needs a non-zero value in order for the terminal to behave in a conforming manner.

119864 On systems which do reset the parameter settings to defaults between uses of a terminal device,  
119865 it is expected that either `O_TTY_INIT` will have the value zero or `open(ttypath,`  
119866 `O_RDWR|O_TTY_INIT)` will do nothing additional.

119867 The standard developers considered an alternative solution of a special *fildev* argument for the  
119868 `tcgetattr()` call to obtain default parameters. However, this would not be adequate if a system  
119869 supports several different types of terminal device and the default settings need to differ  
119870 between the different types. With the `O_TTY_INIT` open flag, the implementor can determine  
119871 which device type is being opened.

119872 The standard developers also considered a special `POSIX_TTY_INIT` value for the **termios**  
119873 structure used in `tcsetattr()`, which would reset the values if used immediately after an `open()`  
119874 call. However, it was felt that this would lead to confusion amongst application developers who  
119875 wanted to reset the parameters at other points, and implementations might diverge.

#### 119876 A.11.1.2 Process Groups

119877 There is a potential race when the members of the foreground process group on a terminal leave  
119878 that process group, either by exit or by changing process groups. After the last process exits the  
119879 process group, but before the foreground process group ID of the terminal is changed (usually  
119880 by a job control shell), it would be possible for a new process to be created with its process ID  
119881 equal to the terminal's foreground process group ID. That process might then become the  
119882 process group leader and accidentally be placed into the foreground on a terminal that was not  
119883 necessarily its controlling terminal. As a result of this problem, the controlling terminal is  
119884 defined to not have a foreground process group during this time.

119885 The cases where a controlling terminal has no foreground process group occur when all  
119886 processes in the foreground process group either terminate and are waited for or join other  
119887 process groups via `setpgid()` or `setsid()`. If the process group leader terminates, this is the first  
119888 case described; if it leaves the process group via `setpgid()`, this is the second case described (a  
119889 process group leader cannot successfully call `setsid()`). When one of those cases causes a  
119890 controlling terminal to have no foreground process group, it has two visible effects on  
119891 applications. The first is the value returned by `tcgetpgrp()`. The second (which occurs only in the  
119892 case where the process group leader terminates) is the sending of signals in response to special  
119893 input characters. The intent of POSIX.1-2008 is that no process group be wrongly identified as  
119894 the foreground process group by `tcgetpgrp()` or unintentionally receive signals because of  
119895 placement into the foreground.

119896 In 4.3 BSD, the old process group ID continues to be used to identify the foreground process  
119897 group and is returned by the function equivalent to `tcgetpgrp()`. In that implementation it is  
119898 possible for a newly created process to be assigned the same value as a process ID and then form  
119899 a new process group with the same value as a process group ID. The result is that the new  
119900 process group would receive signals from this terminal for no apparent reason, and  
119901 POSIX.1-2008 precludes this by forbidding a process group from entering the foreground in this  
119902 way. It would be more direct to place part of the requirement made by the last sentence under  
119903 `fork()`, but there is no convenient way for that section to refer to the value that `tcgetpgrp()`  
119904 returns, since in this case there is no process group and thus no process group ID.

119905 One possibility for a conforming implementation is to behave similarly to 4.3 BSD, but to  
119906 prevent this reuse of the ID, probably in the implementation of `fork()`, as long as it is in use by  
119907 the terminal.

119908 Another possibility is to recognize when the last process stops using the terminal's foreground  
119909 process group ID, which is when the process group lifetime ends, and to change the terminal's  
119910 foreground process group ID to a reserved value that is never used as a process ID or process  
119911 group ID. (See the definition of *process group lifetime* in the definitions section.) The process ID



119912 can then be reserved until the terminal has another foreground process group.

119913 The 4.3 BSD implementation permits the leader (and only member) of the foreground process  
119914 group to leave the process group by calling the equivalent of *setpgid()* and to later return,  
119915 expecting to return to the foreground. There are no known application needs for this behavior,  
119916 and POSIX.1-2008 neither requires nor forbids it (except that it is forbidden for session leaders)  
119917 by leaving it unspecified.

119918 *A.11.1.3 The Controlling Terminal*

119919 POSIX.1-2008 does not specify a mechanism by which to allocate a controlling terminal. This is  
119920 normally done by a system utility (such as *getty*) and is considered an administrative feature  
119921 outside the scope of POSIX.1-2008.

119922 Historical implementations allocate controlling terminals on certain *open()* calls. Since *open()* is  
119923 part of POSIX.1, its behavior had to be dealt with. The traditional behavior is not required  
119924 because it is not very straightforward or flexible for either implementations or applications.  
119925 However, because of its prevalence, it was not practical to disallow this behavior either. Thus, a  
119926 mechanism was standardized to ensure portable, predictable behavior in *open()*.

119927 Some historical implementations deallocate a controlling terminal on the last system-wide close.  
119928 This behavior is neither required nor prohibited. Even on implementations that do provide this  
119929 behavior, applications generally cannot depend on it due to its system-wide nature.

119930 *A.11.1.4 Terminal Access Control*

119931 The access controls described in this section apply only to a process that is accessing its  
119932 controlling terminal. A process accessing a terminal that is not its controlling terminal is  
119933 effectively treated the same as a member of the foreground process group. While this may seem  
119934 unintuitive, note that these controls are for the purpose of job control, not security, and job  
119935 control relates only to the controlling terminal of a process. Normal file access permissions  
119936 handle security.

119937 If the process calling *read()* or *write()* is in a background process group that is orphaned, it is not  
119938 desirable to stop the process group, as it is no longer under the control of a job control shell that  
119939 could put it into the foreground again. Accordingly, calls to *read()* or *write()* functions by such  
119940 processes receive an immediate error return. This is different from 4.2 BSD, which kills orphaned  
119941 processes that receive terminal stop signals.

119942 The foreground/background/orphaned process group check performed by the terminal driver  
119943 must be repeatedly performed until the calling process moves into the foreground or until the  
119944 process group of the calling process becomes orphaned. That is, when the terminal driver  
119945 determines that the calling process is in the background and should receive a job control signal,  
119946 it sends the appropriate signal (SIGTTIN or SIGTTOU) to every process in the process group of  
119947 the calling process and then it allows the calling process to immediately receive the signal. The  
119948 latter is typically performed by blocking the process so that the signal is immediately noticed.  
119949 Note, however, that after the process finishes receiving the signal and control is returned to the  
119950 driver, the terminal driver must re-execute the foreground/background/orphaned process  
119951 group check. The process may still be in the background, either because it was continued in the  
119952 background by a job control shell, or because it caught the signal and did nothing.

119953 The terminal driver repeatedly performs the foreground/background/orphaned process group  
119954 checks whenever a process is about to access the terminal. In the case of *write()* or the control  
119955 *tc\*()* functions, the check is performed at the entry of the function. In the case of *read()*, the  
119956 check is performed not only at the entry of the function, but also after blocking the process to

119957 wait for input characters (if necessary). That is, once the driver has determined that the process  
119958 calling the *read()* function is in the foreground, it attempts to retrieve characters from the input  
119959 queue. If the queue is empty, it blocks the process waiting for characters. When characters are  
119960 available and control is returned to the driver, the terminal driver must return to the repeated  
119961 foreground/background/orphaned process group check again. The process may have moved  
119962 from the foreground to the background while it was blocked waiting for input characters.

119963 A.11.1.5 *Input Processing and Reading Data*

119964 There is no additional rationale provided for this section.

119965 A.11.1.6 *Canonical Mode Input Processing*

119966 The term “character” is intended here. ERASE should erase the last character, not the last byte.  
119967 In the case of multi-byte characters, these two may be different.

119968 4.3 BSD has a WERASE character that erases the last “word” typed (but not any preceding  
119969 <blank> or <tab> characters). A word is defined as a sequence of non-<blank> characters, with  
119970 <tab> characters counted as <blank> characters. Like ERASE, WERASE does not erase beyond  
119971 the beginning of the line. This WERASE feature has not been specified in POSIX.1 because it is  
119972 difficult to define in the international environment. It is only useful for languages where words  
119973 are delimited by <blank> characters. In some ideographic languages, such as Japanese and  
119974 Chinese, words are not delimited at all. The WERASE character should presumably go back to  
119975 the beginning of a sentence in those cases; practically, this means it would not be used much for  
119976 those languages.

119977 It should be noted that there is a possible inherent deadlock if the application and  
119978 implementation conflict on the value of {MAX\_CANON}. With ICANON set (if IXOFF is  
119979 enabled) and more than {MAX\_CANON} characters transmitted without a <linefeed>,  
119980 transmission will be stopped, the <linefeed> (or <carriage-return> when ICRLF is set) will never  
119981 arrive, and the *read()* will never be satisfied.

119982 An application should not set IXOFF if it is using canonical mode unless it knows that (even in  
119983 the face of a transmission error) the conditions described previously cannot be met or unless it  
119984 is prepared to deal with the possible deadlock in some other way, such as timeouts.

119985 It should also be noted that this can be made to happen in non-canonical mode if the trigger  
119986 value for sending IXOFF is less than VMIN and VTIME is zero.

119987 A.11.1.7 *Non-Canonical Mode Input Processing*

119988 Some points to note about MIN and TIME:

- 119989 1. The interactions of MIN and TIME are not symmetric. For example, when MIN>0 and  
119990 TIME=0, TIME has no effect. However, in the opposite case where MIN=0 and TIME>0,  
119991 both MIN and TIME play a role in that MIN is satisfied with the receipt of a single  
119992 character.
- 119993 2. Also note that in case A (MIN>0, TIME>0), TIME represents an inter-character timer,  
119994 while in case C (MIN=0, TIME>0), TIME represents a read timer.

119995 These two points highlight the dual purpose of the MIN/TIME feature. Cases A and B, where  
119996 MIN>0, exist to handle burst-mode activity (for example, file transfer programs) where a  
119997 program would like to process at least MIN characters at a time. In case A, the inter-character  
119998 timer is activated by a user as a safety measure; in case B, it is turned off.



119999 Cases C and D exist to handle single-character timed transfers. These cases are readily adaptable  
 120000 to screen-based applications that need to know if a character is present in the input queue before  
 120001 refreshing the screen. In case C, the read is timed; in case D, it is not.

120002 Another important note is that MIN is always just a minimum. It does not denote a record  
 120003 length. That is, if a program does a read of 20 bytes, MIN is 10, and 25 characters are present, 20  
 120004 characters are returned to the user. In the special case of MIN=0, this still applies: if more than  
 120005 one character is available, they all will be returned immediately.

#### 120006 A.11.1.8 Writing Data and Output Processing

120007 There is no additional rationale provided for this section.

#### 120008 A.11.1.9 Special Characters

120009 There is no additional rationale provided for this section.

#### 120010 A.11.1.10 Modem Disconnect

120011 There is no additional rationale provided for this section.

#### 120012 A.11.1.11 Closing a Terminal Device File

120013 POSIX.1-2008 does not specify that a *close()* on a terminal device file include the equivalent of a  
 120014 call to *tcflow(fd,TCOON)*.

120015 An implementation that discards output at the time *close()* is called after reporting the return  
 120016 value to the *write()* call that data was written does not conform with POSIX.1-2008. An  
 120017 application has functions such as *tcdrain()*, *tcflush()*, and *tcflow()* available to obtain the detailed  
 120018 behavior it requires with respect to flushing of output.

120019 At the time of the last close on a terminal device, an application relinquishes any ability to exert  
 120020 flow control via *tcflow()*.

### 120021 A.11.2 Parameters that Can be Set

#### 120022 A.11.2.1 The termios Structure

120023 This structure is part of an interface that, in general, retains the historic grouping of flags.  
 120024 Although a more optimal structure for implementations may be possible, the degree of change  
 120025 to applications would be significantly larger.

#### 120026 A.11.2.2 Input Modes

120027 Some historical implementations treated a long break as multiple events, as many as one per  
 120028 character time. The wording in POSIX.1 explicitly prohibits this.

120029 Although the ISTRIP flag is normally superfluous with today's terminal hardware and software,  
 120030 it is historically supported. Therefore, applications may be using ISTRIP, and there is no  
 120031 technical problem with supporting this flag. Also, applications may wish to receive only 7-bit  
 120032 input bytes and may not be connected directly to the hardware terminal device (for example,

120033 when a connection traverses a network).

120034 Also, there is no requirement in general that the terminal device ensures that high-order bits  
120035 beyond the specified character size are cleared. ISTRIP provides this function for 7-bit  
120036 characters, which are common.

120037 In dealing with multi-byte characters, the consequences of a parity error in such a character, or  
120038 in an escape sequence affecting the current character set, are beyond the scope of POSIX.1 and  
120039 are best dealt with by the application processing the multi-byte characters.

#### 120040 *A.11.2.3 Output Modes*

120041 POSIX.1 does not describe post-processing of output to a terminal or detailed control of that  
120042 from a conforming application. (That is, translation of <newline> to <carriage-return> followed  
120043 by <linefeed> or <tab> processing.) There is nothing that a conforming application should do to  
120044 its output for a terminal because that would require knowledge of the operation of the terminal.  
120045 It is the responsibility of the operating system to provide post-processing appropriate to the  
120046 output device, whether it is a terminal or some other type of device.

120047 Extensions to POSIX.1 to control the type of post-processing already exist and are expected to  
120048 continue into the future. The control of these features is primarily to adjust the interface between  
120049 the system and the terminal device so the output appears on the display correctly. This should  
120050 be set up before use by any application.

120051 In general, both the input and output modes should not be set absolutely, but rather modified  
120052 from the inherited state.

#### 120053 *A.11.2.4 Control Modes*

120054 This section could be misread that the symbol "CSIZE" is a title in the **termios** *c\_flag* field.  
120055 Although it does serve that function, it is also a required symbol, as a literal reading of POSIX.1  
120056 (and the caveats about typography) would indicate.

#### 120057 *A.11.2.5 Local Modes*

120058 Non-canonical mode is provided to allow fast bursts of input to be read efficiently while still  
120059 allowing single-character input.

120060 The ECHONL function historically has been in many implementations. Since there seems to be  
120061 no technical problem with supporting ECHONL, it is included in POSIX.1 to increase consensus.

120062 The alternate behavior possible when ECHOK or ECHOE are specified with ICANON is  
120063 permitted as a compromise depending on what the actual terminal hardware can do. Erasing  
120064 characters and lines is preferred, but is not always possible.

#### 120065 *A.11.2.6 Special Control Characters*

120066 Permitting VMIN and VTIME to overlap with VEOF and VEOL was a compromise for historical  
120067 implementations. Only when backwards-compatibility of object code is a serious concern to an  
120068 implementor should an implementation continue this practice. Correct applications that work  
120069 with the overlap (at the source level) should also work if it is not present, but not the reverse.

## A.12 Utility Conventions

### A.12.1 Utility Argument Syntax

The standard developers considered that recent trends toward diluting the SYNOPSIS sections of historical reference pages to the equivalent of:

```
command [options][operands]
```

were a disservice to the reader. Therefore, considerable effort was placed into rigorous definitions of all the command line arguments and their interrelationships. The relationships depicted in the synopses are normative parts of POSIX.1-2008; this information is sometimes repeated in textual form, but that is only for clarity within context.

The use of “undefined” for conflicting argument usage and for repeated usage of the same option is meant to prevent conforming applications from using conflicting arguments or repeated options unless specifically allowed (as is the case with *ls*, which allows simultaneous, repeated use of the *-C*, *-l*, and *-1* options). Many historical implementations will tolerate this usage, choosing either the first or the last applicable argument. This tolerance can continue, but conforming applications cannot rely upon it. (Other implementations may choose to print usage messages instead.)

The use of “undefined” for conflicting argument usage also allows an implementation to make reasonable extensions to utilities where the implementor considers mutually-exclusive options according to POSIX.1-2008 to have a sensible meaning and result.

POSIX.1-2008 does not define the result of a command when an option-argument or operand is not followed by ellipses and the application specifies more than one of that option-argument or operand. This allows an implementation to define valid (although non-standard) behavior for the utility when more than one such option or operand is specified.

The requirements for option-arguments are summarized as follows:

	SYNOPSIS Shows:	
	<i>-a arg</i>	<i>-c[ arg]</i>
Conforming application uses:	<i>-a arg</i>	<i>-carg</i> or <i>-c</i>
System supports:	<i>-a arg</i> and <i>-aarg</i>	<i>-carg</i> and <i>-c</i>
Non-conforming applications may use:	<i>-aarg</i>	N/A

Earlier versions of this standard included obsolescent syntax which showed some options with (mandatory) adjacent option-arguments in the SYNOPSIS for some utilities. These have since been removed. For all options with mandatory option-arguments, the SYNOPSIS now shows <blank> characters between the option and the option-argument; however, historical usage has not been consistent in this area; therefore, <blank> characters are required to be used by conforming applications and to be handled by all implementations, but implementations are also required to handle an adjacent option-argument in order to preserve backwards-compatibility for old scripts. One of the justifications for selecting the multiple-argument method was that the single-argument case is inherently ambiguous when the option-argument can legitimately be a null string.

POSIX.1-2008 explicitly states that digits are permitted as operands and option-arguments. The lower and upper bounds for the values of the numbers used for operands and option-arguments were derived from the ISO C standard values for {LONG\_MIN} and {LONG\_MAX}. The

requirement on the standard utilities is that numbers in the specified range do not cause a syntax error, although the specification of a number need not be semantically correct for a particular operand or option-argument of a utility. For example, the specification of:

```
dd obs=3000000000
```

would yield undefined behavior for the application and could be a syntax error because the number 3 000 000 000 is outside of the range -2 147 483 647 to +2 147 483 647. On the other hand:

```
dd obs=2000000000
```

may cause some error, such as “blocksize too large”, rather than a syntax error.

## A.12.2 Utility Syntax Guidelines

This section is based on the rules listed in the SVID. It was included for two reasons:

1. The individual utility descriptions in XCU [Chapter 4](#) (on page 2443) needed a set of common (although not universal) actions on which they could anchor their descriptions of option and operand syntax. Most of the standard utilities actually do use these guidelines, and many of their historical implementations use the *getopt()* function for their parsing. Therefore, it was simpler to cite the rules and merely identify exceptions.
2. Developers of conforming applications need suggested guidelines if the POSIX community is to avoid the chaos of historical UNIX system command syntax.

It is recommended that all *future* utilities and applications use these guidelines to enhance “user portability”. The fact that some historical utilities could not be changed (to avoid breaking historical applications) should not deter this future goal.

The voluntary nature of the guidelines is highlighted by repeated uses of the word *should* throughout. This usage should not be misinterpreted to imply that utilities that claim conformance in their OPTIONS sections do not always conform.

Guidelines 1 and 2 encourage utility writers to use only characters from the portable character set because use of locale-specific characters may make the utility inaccessible from other locales. Use of uppercase letters is discouraged due to problems associated with porting utilities to systems that do not distinguish between uppercase and lowercase characters in filenames. Use of non-alphanumeric characters is discouraged due to the number of utilities that treat non-alphanumeric characters in “special” ways depending on context (such as the shell using white-space characters to delimit arguments, various quote characters for quoting, the <dollar-sign> to introduce variable expansion, etc.).

In XCU [Section 2.9.1](#) (on page 2355), it is further stated that a command used in the Shell Command Language cannot be named with a trailing <colon>.

Guideline 3 was changed to allow alphanumeric characters (letters and digits) from the character set to allow compatibility with historical usage. Historical practice allows the use of digits wherever practical, and there are no portability issues that would prohibit the use of digits. In fact, from an internationalization viewpoint, digits (being non-language-dependent) are preferable over letters (a -2 is intuitively self-explanatory to any user, while in the -f *filename* the letter ‘f’ is a mnemonic aid only to speakers of Latin-based languages where “filename” happens to translate to a word that begins with ‘f’). Since Guideline 3 still retains the word “single”, multi-digit options are not allowed. Instances of historical utilities that used them have been marked obsolescent, with the numbers being changed from option names to option-arguments.

It was difficult to achieve a satisfactory solution to the problem of name space in option

characters. When the standard developers desired to extend the historical *cc* utility to accept ISO C standard programs, they found that all of the portable alphabet was already in use by various vendors. Thus, they had to devise a new name, *c89* (now superseded by *c99*), rather than something like *cc -X*. There were suggestions that implementors be restricted to providing extensions through various means (such as using a <plus-sign> as the option delimiter or using option characters outside the alphanumeric set) that would reserve all of the remaining alphanumeric characters for future POSIX standards. These approaches were resisted because they lacked the historical style of UNIX systems. Furthermore, if a vendor-provided option should become commonly used in the industry, it would be a candidate for standardization. It would be desirable to standardize such a feature using historical practice for the syntax (the semantics can be standardized with any syntax). This would not be possible if the syntax was one reserved for the vendor. However, since the standardization process may lead to minor changes in the semantics, it may prove to be better for a vendor to use a syntax that will not be affected by standardization.

Guideline 8 includes the concept of <comma>-separated lists in a single argument. It is up to the utility to parse such a list itself because *getopt()* just returns the single string. This situation was retained so that certain historical utilities would not violate the guidelines. Applications preparing for international use should be aware of an occasional problem with <comma>-separated lists: in some locales, the <comma> is used as the radix character. Thus, if an application is preparing operands for a utility that expects a <comma>-separated list, it should avoid generating non-integer values through one of the means that is influenced by setting the *LC\_NUMERIC* variable (such as *awk*, *bc*, *printf*, or *printf()*).

Unless explicitly stated otherwise in the utility description, Guideline 9 requires applications to put options before operands, and requires utilities to accept any such usage without misinterpreting operands as options. For example, if an implementation of the *printf* utility supports a *-e* option as an extension, the command:

```
printf %s -e
```

must output the string "-e" without interpreting the *-e* as an option. Similarly, the command:

```
ls myfile -l
```

must interpret the *-l* argument as a second file operand, not as a *-l* option.

Applications calling any utility with a first operand starting with '-' should usually specify *--*, as indicated by Guideline 10, to mark the end of the options. This is true even if the SYNOPSIS in the Shell and Utilities volume of POSIX.1-2008 does not specify any options; implementations may provide options as extensions to the Shell and Utilities volume of POSIX.1-2008. The standard utilities that do not support Guideline 10 indicate that fact in the OPTIONS section of the utility description.

Guideline 7 allows any string to be an option-argument; an option-argument can begin with any character, can be - or --, and can be an empty string. For example, the commands *pr -h -*, *pr -h --*, *pr -h -d*, *pr -h +2*, and *pr -h ''* contain the option-arguments -, --, -d, +2, and an empty string, respectively. Conversely, the command *pr -h -- -d* treats -d as an option, not as an argument, because the -- is an option-argument here, not a delimiter.

Guideline 11 was modified to clarify that the order of different options should not matter relative to one another. However, the order of repeated options that also have option-arguments may be significant; therefore, such options are required to be interpreted in the order that they are specified. The *make* utility is an instance of a historical utility that uses repeated options in which the order is significant. Multiple files are specified by giving multiple instances of the *-f* option; for example:

```
make -f common_header -f specific_rules target
```

Guideline 13 does not imply that all of the standard utilities automatically accept the operand `'-'` to mean standard input or output, nor does it specify the actions of the utility upon encountering multiple `'-'` operands. It simply says that, by default, `'-'` operands are not used for other purposes in the file reading or writing (but not when using `stat()`, `unlink()`, `touch`, and so on) utilities. In earlier versions of this standard, all information concerning actual treatment of the `'-'` operand is found in the individual utility sections. Many implementations, however, treated `'-'` as standard input or output and many applications depended on this behavior even though it was not standard. This behavior is now implementation-defined. Portable applications should not use `'-'` to mean standard input or output unless it is explicitly stated to do so in the utility description and they should always use `'./-'` if they intend to refer to a file named `-` in the current working directory.

Guideline 14 is intended to prohibit implementations that would treat the command `ls -l -d` as if it were `ls -- -l -d` or `ls -l -- -d`.

The standard permits implementations to have extensions that violate the Utility Syntax Guidelines so long as when the utility is used in line with the forms defined by the standard it follows the Utility Syntax Guidelines. Thus, `head-42file` and `ls--help` are permitted extensions. The intent is to allow extensions so long as the standard form is accepted and follows the guidelines.

An area of concern was that as implementations mature, implementation-defined utilities and implementation-defined utility options will result. The idea was expressed that there needed to be a standard way, say an environment variable or some such mechanism, to identify implementation-defined utilities separately from standard utilities that may have the same name. It was decided that there already exist several ways of dealing with this situation and that it is outside of the scope to attempt to standardize in the area of non-standard items. A method that exists on some historical implementations is the use of the so-called `/local/bin` or `/usr/local/bin` directory to separate local or additional copies or versions of utilities. Another method that is also used is to isolate utilities into completely separate domains. Still another method to ensure that the desired utility is being used is to request the utility by its full pathname. There are many approaches to this situation; the examples given above serve to illustrate that there is more than one.

## A.13 Headers

### A.13.1 Format of Entries

Each header reference page has a common layout of sections describing the interface. This layout is similar to the manual page or “man” page format shipped with most UNIX systems, and each header has sections describing the SYNOPSIS and DESCRIPTION. These are the two sections that relate to conformance.

Additional sections are informative, and add considerable information for the application developer. APPLICATION USAGE sections provide additional caveats, issues, and recommendations to the developer. RATIONALE sections give additional information on the decisions made in defining the interface.

FUTURE DIRECTIONS sections act as pointers to related work that may impact the interface in the future, and often cautions the developer to architect the code to account for a change in this area. Note that a future directions statement should not be taken as a commitment to adopt a



- 120247 feature or interface in the future.
- 120248 The CHANGE HISTORY section describes when the interface was introduced, and how it has  
120249 changed.
- 120250 Option labels and margin markings in the page can be useful in guiding the application  
120251 developer.

120252 **A.13.2 Removed Headers in Issue 7**

- 120253 The headers removed in Issue 7 (from the Issue 6 base document) are as follows:

120254

120255

Removed Headers in Issue 7	
<sys/timeb.h>	<ucontext.h>





120256

# *Rationale (Informative)*

120257

## **Part B:**

120258

## **System Interfaces**

120259

*The Open Group*

120260

*The Institute of Electrical and Electronics Engineers, Inc.*



120261

Appendix B

120262

# *Rationale for System Interfaces*

## **B.1 Introduction**

### **B.1.1 Change History**

The change history is provided as an informative section, to track changes from earlier versions of this standard.

The following sections describe changes made to the System Interfaces volume of POSIX.1-2008 since Issue 6 of the base document. The CHANGE HISTORY section for each entry details the technical changes that have been made to that entry from Issue 5. Changes between earlier versions of the base document and Issue 5 are not included.

#### **Changes from Issue 6 to Issue 7 (POSIX.1-2008)**

The following list summarizes the major changes that were made in the System Interfaces volume of POSIX.1-2008 from Issue 6 to Issue 7:

- The Open Group Technical Standard, 2006, Extended API Set Part 1 is incorporated.
- The Open Group Technical Standard, 2006, Extended API Set Part 2 is incorporated.
- The Open Group Technical Standard, 2006, Extended API Set Part 3 is incorporated.
- The Open Group Technical Standard, 2006, Extended API Set Part 4 is incorporated.
- Existing functionality is aligned with ISO/IEC 9899:1999, Programming Languages — C, ISO/IEC 9899:1999/Cor.2:2004(E)
- Austin Group defect reports, IEEE Interpretations against IEEE Std 1003.1, and responses to ISO/IEC defect reports against ISO/IEC 9945 are applied.
- The Open Group corrigenda and resolutions are applied.
- Features, marked legacy or obsolescent in the base document, have been considered for removal in this version.
- The options within the standard have been revised.

**New Features in Issue 7**

The functions first introduced in Issue 7 (over the Issue 6 base document) are as follows:

**New Functions in Issue 7**

<i>alphasort()</i>	<i>iswdigit_l()</i>	<i>strfmon_l()</i>
<i>dirfd()</i>	<i>iswgraph_l()</i>	<i>strncascmp_l()</i>
<i>dprintf()</i>	<i>iswlower_l()</i>	<i>strndup()</i>
<i>duplocale()</i>	<i>iswprint_l()</i>	<i>strnlen()</i>
<i>faccessat()</i>	<i>iswpunct_l()</i>	<i>strsignal()</i>
<i>fchmodat()</i>	<i>iswspace_l()</i>	<i>strxfrm_l()</i>
<i>fchownat()</i>	<i>iswupper_l()</i>	<i>symlinkat()</i>
<i>fdopendir()</i>	<i>iswxdigit_l()</i>	<i>tolower_l()</i>
<i>execve()</i>	<i>isxdigit_l()</i>	<i>toupper_l()</i>
<i>fnmopen()</i>	<i>linkat()</i>	<i>towctrans_l()</i>
<i>freelocale()</i>	<i>mbsnrtowcs()</i>	<i>tolower_l()</i>
<i>fstatat()</i>	<i>mkdirat()</i>	<i>towupper_l()</i>
<i>futimens()</i>	<i>mkdtemp()</i>	<i>unlinkat()</i>
<i>getdelim()</i>	<i>mkffloat()</i>	<i>uselocale()</i>
<i>getline()</i>	<i>mknodat()</i>	<i>utimensat()</i>
<i>isalnum_l()</i>	<i>newlocale()</i>	<i>vdprintf()</i>
<i>isalpha_l()</i>	<i>openat()</i>	<i>wcpcpy()</i>
<i>isblank_l()</i>	<i>open_memstream()</i>	<i>wcpncpy()</i>
<i>iscntrl_l()</i>	<i>open_wmemstream()</i>	<i>wscascmp_l()</i>
<i>isdigit_l()</i>	<i>psiginfo()</i>	<i>wscascmp_l()</i>
<i>isgraph_l()</i>	<i>psignal()</i>	<i>wscoll_l()</i>
<i>islower_l()</i>	<i>pthread_mutexattr_getrobust()</i>	<i>wcsdup()</i>
<i>isprint_l()</i>	<i>pthread_mutexattr_setrobust()</i>	<i>wcsncascmp_l()</i>
<i>ispunct_l()</i>	<i>pthread_mutex_consistent()</i>	<i>wcsncascmp_l()</i>
<i>isspace_l()</i>	<i>readlinkat()</i>	<i>wcsnlen()</i>
<i>isupper_l()</i>	<i>renameat()</i>	<i>wcsnrtombs()</i>
<i>iswalnum_l()</i>	<i>scandir()</i>	<i>wcsxfrm_l()</i>
<i>iswalphal_l()</i>	<i>stpcpy()</i>	<i>wctrans_l()</i>
<i>iswblank_l()</i>	<i>stpncpy()</i>	<i>wctype_l()</i>
<i>iswcntrl_l()</i>	<i>strcascmp_l()</i>	
<i>iswctype_l()</i>	<i>strcoll_l()</i>	

**Newly Mandated Functions in Issue 7**

The functions that were previously part of an option group but are now mandatory in Issue 7 are as follows:

Newly Mandated Functions in Issue 7		
120323		
120324	<i>aiocancel()</i>	<i>pthread_atfork()</i>
120325	<i>aioerror()</i>	<i>pthread_attr_destroy()</i>
120326	<i>aiofsync()</i>	<i>pthread_attr_getdetachstate()</i>
120327	<i>aio_read()</i>	<i>pthread_attr_getguardsize()</i>
120328	<i>aio_return()</i>	<i>pthread_attr_getschedparam()</i>
120329	<i>aio_suspend()</i>	<i>pthread_attr_init()</i>
120330	<i>aio_write()</i>	<i>pthread_attr_setdetachstate()</i>
120331	<i>asctime_r()</i>	<i>pthread_attr_setguardsize()</i>
120332	<i>catclose()</i>	<i>pthread_attr_setschedparam()</i>
120333	<i>catgets()</i>	<i>pthread_barrier_destroy()</i>
120334	<i>catopen()</i>	<i>pthread_barrier_init()</i>
120335	<i>clock_getres()</i>	<i>pthread_barrier_wait()</i>
120336	<i>clock_gettime()</i>	<i>pthread_barrierattr_destroy()</i>
120337	<i>clock_nanosleep()</i>	<i>pthread_barrierattr_init()</i>
120338	<i>clock_settime()</i>	<i>pthread_cancel()</i>
120339	<i>ctime_r()</i>	<i>pthread_cleanup_pop()</i>
120340	<i>dlclose()</i>	<i>pthread_cleanup_push()</i>
120341	<i>dlderror()</i>	<i>pthread_cond_broadcast()</i>
120342	<i>dlopen()</i>	<i>pthread_cond_destroy()</i>
120343	<i>dlsym()</i>	<i>pthread_cond_init()</i>
120344	<i>fchdir()</i>	<i>pthread_cond_signal()</i>
120345	<i>flockfile()</i>	<i>pthread_cond_timedwait()</i>
120346	<i>fstatvfs()</i>	<i>pthread_cond_wait()</i>
120347	<i>ftrylockfile()</i>	<i>pthread_condattr_destroy()</i>
120348	<i>funlockfile()</i>	<i>pthread_condattr_getclock()</i>
120349	<i>getc_unlocked()</i>	<i>pthread_condattr_init()</i>
120350	<i>getchar_unlocked()</i>	<i>pthread_condattr_setclock()</i>
120351	<i>getgrgid_r()</i>	<i>pthread_create()</i>
120352	<i>getgrnam_r()</i>	<i>pthread_detach()</i>
120353	<i>getlogin_r()</i>	<i>pthread_equal()</i>
120354	<i>getpgid()</i>	<i>pthread_exit()</i>
120355	<i>getpwnam_r()</i>	<i>pthread_getspecific()</i>
120356	<i>getpwuid_r()</i>	<i>pthread_join()</i>
120357	<i>getsid()</i>	<i>pthread_key_create()</i>
120358	<i>getsubopt()</i>	<i>pthread_key_delete()</i>
120359	<i>gmtime_r()</i>	<i>pthread_mutex_destroy()</i>
120360	<i>iconv()</i>	<i>pthread_mutex_init()</i>
120361	<i>iconv_close()</i>	<i>pthread_mutex_lock()</i>
120362	<i>iconv_open()</i>	<i>pthread_mutex_timedlock()</i>
120363	<i>lchown()</i>	<i>pthread_mutex_trylock()</i>
120364	<i>lio_listio()</i>	<i>pthread_mutex_unlock()</i>
120365	<i>localtime_r()</i>	<i>pthread_mutexattr_destroy()</i>
120366	<i>mkstemp()</i>	<i>pthread_mutexattr_gettype()</i>
120367	<i>mmap()</i>	<i>pthread_mutexattr_init()</i>
120368	<i>mprotect()</i>	<i>pthread_mutexattr_settype()</i>
120369	<i>munmap()</i>	<i>pthread_once()</i>
120370	<i>nanosleep()</i>	<i>pthread_rwlock_destroy()</i>
120371	<i>nl_langinfo()</i>	<i>pthread_rwlock_init()</i>
120372	<i>poll()</i>	<i>pthread_rwlock_rdlock()</i>
120373	<i>posix_trace_timedgetnext_event()</i>	<i>pthread_rwlock_timedrdlock()</i>
120374	<i>pread()</i>	<i>pthread_rwlock_timedwrlock()</i>
		<i>pthread_rwlock_tryrdlock()</i>
		<i>pthread_rwlock_trywrlock()</i>
		<i>pthread_rwlock_unlock()</i>
		<i>pthread_rwlock_wrlock()</i>
		<i>pthread_rwlockattr_destroy()</i>
		<i>pthread_rwlockattr_init()</i>
		<i>pthread_self()</i>
		<i>pthread_setcancelstate()</i>
		<i>pthread_setcanceltype()</i>
		<i>pthread_setspecific()</i>
		<i>pthread_spin_destroy()</i>
		<i>pthread_spin_init()</i>
		<i>pthread_spin_lock()</i>
		<i>pthread_spin_trylock()</i>
		<i>pthread_spin_unlock()</i>
		<i>pthread_testcancel()</i>
		<i>putc_unlocked()</i>
		<i>putchar_unlocked()</i>
		<i>pwrite()</i>
		<i>rand_r()</i>
		<i>readdir_r()</i>
		<i>sem_close()</i>
		<i>sem_destroy()</i>
		<i>sem_getvalue()</i>
		<i>sem_init()</i>
		<i>sem_open()</i>
		<i>sem_post()</i>
		<i>sem_timedwait()</i>
		<i>sem_trywait()</i>
		<i>sem_unlink()</i>
		<i>sem_wait()</i>
		<i>sigqueue()</i>
		<i>sigqueue()</i>
		<i>sigtimedwait()</i>
		<i>sigwaitinfo()</i>
		<i>statvfs()</i>
		<i>strcasecmp()</i>
		<i>strdup()</i>
		<i>strerror_r()</i>
		<i>strfmon()</i>
		<i>strncasecmp()</i>
		<i>strtok_r()</i>
		<i>tcgetsid()</i>
		<i>timer_create()</i>
		<i>timer_delete()</i>
		<i>timer_getoverrun()</i>
		<i>timer_gettime()</i>
		<i>timer_settime()</i>
		<i>truncate()</i>
		<i>ttynam_r()</i>
		<i>waitid()</i>

**Obsolescent Functions in Issue 7**

The base functions moved to obsolescent status in Issue 7 (from the Issue 6 base document) are as follows:

**Obsolescent Base Functions in Issue 7**

<i>asctime()</i>	<i>gets()</i>
<i>asctime_r()</i>	<i>rand_r()</i>
<i>ctime()</i>	<i>tmpnam()</i>
<i>ctime_r()</i>	<i>utime()</i>

The XSI functions moved to obsolescent status in Issue 7 (from the Issue 6 base document) are as follows:

**Obsolescent XSI Functions in Issue 7**

<i>_longjmp()</i>	<i>pthread_getconcurrency()</i>	<i>sigset()</i>
<i>_setjmp()</i>	<i>pthread_setconcurrency()</i>	<i>siginterrupt()</i>
<i>_tolower()</i>	<i>setitimer()</i>	<i>tempnam()</i>
<i>_toupper()</i>	<i>setpggrp()</i>	<i>toascii()</i>
<i>ftw()</i>	<i>sighold()</i>	<i>ulimit()</i>
<i>getitimer()</i>	<i>sigignore()</i>	
<i>gettimeofday()</i>	<i>sigpause()</i>	
<i>isascii()</i>	<i>sigrelse()</i>	

**Removed Functions and Symbols in Issue 7**

The functions and symbols removed in Issue 7 (from the Issue 6 base document) are as follows:

**Removed Functions and Symbols in Issue 7**

<i>bcmp()</i>	<i>gethostbyaddr()</i>	<i>rindex()</i>
<i>bcopy()</i>	<i>gethostbyname()</i>	<i>scalb()</i>
<i>bsd_signal()</i>	<i>getwd()</i>	<i>setcontext()</i>
<i>bzero()</i>	<i>h_errno</i>	<i>swapcontext()</i>
<i>ecvt()</i>	<i>index()</i>	<i>ualarm()</i>
<i>fcvt()</i>	<i>makecontext()</i>	<i>usleep()</i>
<i>ftime()</i>	<i>mktemp()</i>	<i>vfork()</i>
<i>gcvt()</i>	<i>pthread_attr_getstackaddr()</i>	<i>wcswcs()</i>
<i>getcontext()</i>	<i>pthread_attr_setstackaddr()</i>	

**B.1.2 Relationship to Other Formal Standards**

There is no additional rationale provided for this section.

**B.1.3 Format of Entries**

Each system interface reference page has a common layout of sections describing the interface. This layout is similar to the manual page or “man” page format shipped with most UNIX systems, and each header has sections describing the SYNOPSIS, DESCRIPTION, RETURN VALUE, and ERRORS. These are the four sections that relate to conformance.

Additional sections are informative, and add considerable information for the application developer. EXAMPLES sections provide example usage. APPLICATION USAGE sections

120415 provide additional caveats, issues, and recommendations to the developer. RATIONALE  
 120416 sections give additional information on the decisions made in defining the interface.

120417 FUTURE DIRECTIONS sections act as pointers to related work that may impact the interface in  
 120418 the future, and often cautions the developer to architect the code to account for a change in this  
 120419 area. Note that a future directions statement should not be taken as a commitment to adopt a  
 120420 feature or interface in the future.

120421 The CHANGE HISTORY section describes when the interface was introduced, and how it has  
 120422 changed.

120423 Option labels and margin markings in the page can be useful in guiding the application  
 120424 developer.

## 120425 B.2 General Information

### 120426 B.2.1 Use and Implementation of Interfaces

120427 The information concerning the use of functions was adapted from a description in the ISO C  
 120428 standard. Here is an example of how an application program can protect itself from functions  
 120429 that may or may not be macros, rather than true functions:

120430 The `atoi()` function may be used in any of several ways:

- 120431 • By use of its associated header (possibly generating a macro expansion):

```
120432 #include <stdlib.h>
120433 /* ... */
120434 i = atoi(str);
```

- 120435 • By use of its associated header (assuredly generating a true function call):

```
120436 #include <stdlib.h>
120437 #undef atoi
120438 /* ... */
120439 i = atoi(str);
```

120440 or:

```
120441 #include <stdlib.h>
120442 /* ... */
120443 i = (atoi) (str);
```

- 120444 • By explicit declaration:

```
120445 extern int atoi (const char *);
120446 /* ... */
120447 i = atoi(str);
```

- 120448 • By implicit declaration:

```
120449 /* ... */
120450 i = atoi(str);
```

120451 (Assuming no function prototype is in scope. This is not allowed by the ISO C standard for  
 120452 functions with variable arguments; furthermore, parameter type conversion “widening” is

120453 subject to different rules in this case.)

120454 Note that the ISO C standard reserves names starting with ‘\_’ for the compiler. Therefore, the  
 120455 compiler could, for example, implement an intrinsic, built-in function `_asm_builtin_atoi()`, which  
 120456 it recognized and expanded into inline assembly code. Then, in `<stdlib.h>`, there could be the  
 120457 following:

120458 `#define atoi(X) _asm_builtin_atoi(X)`

120459 The user’s “normal” call to `atoi()` would then be expanded inline, but the implementor would  
 120460 also be required to provide a callable function named `atoi()` for use when the application  
 120461 requires it; for example, if its address is to be stored in a function pointer variable.

120462 Implementors should note that since applications can **#undef** a macro in order to ensure that the  
 120463 function is used, this means that it is not safe for implementations to use the names of any  
 120464 standard functions in macro values, since the application could use **#undef** to ensure that no  
 120465 macro exists and then use the same name for an identifier with local scope. For example,  
 120466 historically it was common for a `getchar()` macro to be defined in `<stdio.h>` as:

120467 `#define getchar() getc(stdin)`

120468 This definition does not conform, because an application is allowed to use the identifier `getc`  
 120469 with local scope, and the expansion of the `getchar()` macro would then pick up the local `getc`.  
 120470 The following is conforming code, but would not compile with the above definition of `getchar()`:

120471 `#include <stdio.h>`

120472 `#undef getc`

120473 `int main(void)`

120474 `{`

120475  `int getc;`

120476  `getc = getchar();`

120477  `return getc;`

120478 `}`

120479 This does not only affect function-like macros. For example, the following definition does not  
 120480 conform because there could be a local `sysconf` variable in scope when `SIGRTMIN` is expanded:

120481 `#define SIGRTMIN ((int)sysconf(_SC_SIGRT_MIN))`

120482 Implementors can avoid the problem by using aliases for standard functions instead of the  
 120483 actual function, with names that conforming applications cannot use for local variables. For  
 120484 example:

120485 `#define SIGRTMIN ((int)__sysconf(_SC_SIGRT_MIN))`

## 120486 **B.2.2 The Compilation Environment**

### 120487 *B.2.2.1 POSIX.1 Symbols*

120488 This and the following section address the issue of “name space pollution”. The ISO C standard  
 120489 requires that the name space beyond what it reserves not be altered except by explicit action of  
 120490 the application developer. This section defines the actions to add the POSIX.1 symbols for those  
 120491 headers where both the ISO C standard and POSIX.1 need to define symbols, and also where the



120492 XSI option extends the base standard.

120493 When headers are used to provide symbols, there is a potential for introducing symbols that the  
120494 application developer cannot predict. Ideally, each header should only contain one set of  
120495 symbols, but this is not practical for historical reasons. Thus, the concept of feature test macros is  
120496 included. Two feature test macros are explicitly defined by POSIX.1-2008; it is expected that  
120497 future versions may add to this.

120498 **Note:** Feature test macros allow an application to announce to the implementation its desire to have  
120499 certain symbols and prototypes exposed. They should not be confused with the version test  
120500 macros and constants for options in `<unistd.h>` which are the implementation's way of  
120501 announcing functionality to the application.

120502 It is further intended that these feature test macros apply only to the headers specified by  
120503 POSIX.1-2008. Implementations are expressly permitted to make visible symbols not specified  
120504 by POSIX.1-2008, within both POSIX.1 and other headers, under the control of feature test  
120505 macros that are not defined by POSIX.1-2008.

#### 120506 **The `_POSIX_C_SOURCE` Feature Test Macro**

120507 The POSIX.1-1990 standard specified a macro called `_POSIX_SOURCE`. This has been  
120508 superseded by `_POSIX_C_SOURCE`. This symbol will allow implementations to support various  
120509 versions of this standard simultaneously. For instance, when `_POSIX_C_SOURCE` is defined as  
120510 200809L, the system should make visible the same name space as permitted and required by the  
120511 POSIX.1-2008 standard. A special case is the one where the implementation wishes to make  
120512 available support for the 1990 version of the POSIX standard, in which instance when either  
120513 `_POSIX_SOURCE` is defined or `_POSIX_C_SOURCE` is defined as 1, the system should make  
120514 visible the same name space as permitted and required by the POSIX.1-1990 standard.

120515 It is expected that C bindings to future POSIX standards will define new values for  
120516 `_POSIX_C_SOURCE`, with each new value reserving the name space for that new standard.

#### 120517 **The `_XOPEN_SOURCE` Feature Test Macro**

120518 The feature test macro `_XOPEN_SOURCE` is provided as the announcement mechanism for the  
120519 application that it requires functionality from the Single UNIX Specification. `_XOPEN_SOURCE`  
120520 must be defined to the value 700 before the inclusion of any header to enable the functionality in  
120521 the Single UNIX Specification Version 4. Its definition subsumes the use of `_POSIX_C_SOURCE`.

120522 An extract of code from a conforming application, that appears before any **#include** statements,  
120523 is given below:

```
120524 #define _XOPEN_SOURCE 700 /* Single UNIX Specification, Version 4 */  
120525 #include ...
```

120526 Note that the definition of `_XOPEN_SOURCE` with the value 700 makes the definition of  
120527 `_POSIX_C_SOURCE` redundant and it can safely be omitted.

#### 120528 *B.2.2.2 The Name Space*

120529 The reservation of identifiers is paraphrased from the ISO C standard. The text is included  
120530 because it needs to be part of POSIX.1-2008, regardless of possible changes in future versions of  
120531 the ISO C standard.

120532 These identifiers may be used by implementations, particularly for feature test macros.  
120533 Implementations should not use feature test macro names that might be reasonably used by a  
120534 standard.

Including headers more than once is a reasonably common practice, and it should be carried forward from the ISO C standard. More significantly, having definitions in more than one header is explicitly permitted. Where the potential declaration is “benign” (the same definition twice) the declaration can be repeated, if that is permitted by the compiler. (This is usually true of macros, for example.) In those situations where a repetition is not benign (for example, **typedefs**), conditional compilation must be used. The situation actually occurs both within the ISO C standard and within POSIX.1: **time\_t** should be in `<sys/types.h>`, and the ISO C standard mandates that it be in `<time.h>`.

The area of name space pollution *versus* additions to structures is difficult because of the macro structure of C. The following discussion summarizes all the various problems with and objections to the issue.

Note the phrase “user-defined macro”. Users are not permitted to define macro names (or any other name) beginning with “\_**[A-Z\_]**”. Thus, the conflict cannot occur for symbols reserved to the vendor’s name space, and the permission to add fields automatically applies, without qualification, to those symbols.

1. Data structures (and unions) need to be defined in headers by implementations to meet certain requirements of POSIX.1 and the ISO C standard.

2. The structures defined by POSIX.1 are typically minimal, and any practical implementation would wish to add fields to these structures either to hold additional related information or for backwards-compatibility (or both). Future standards (and *de facto* standards) would also wish to add to these structures. Issues of field alignment make it impractical (at least in the general case) to simply omit fields when they are not defined by the particular standard involved.

The **dirent** structure is an example of such a minimal structure (although one could argue about whether the other fields need visible names). The *st\_rdev* field of most implementations’ **stat** structure is a common example where extension is needed and where a conflict could occur.

3. Fields in structures are in an independent name space, so the addition of such fields presents no problem to the C language itself in that such names cannot interact with identically named user symbols because access is qualified by the specific structure name.

4. There is an exception to this: macro processing is done at a lexical level. Thus, symbols added to a structure might be recognized as user-provided macro names at the location where the structure is declared. This only can occur if the user-provided name is declared as a macro before the header declaring the structure is included. The user’s use of the name after the declaration cannot interfere with the structure because the symbol is hidden and only accessible through access to the structure. Presumably, the user would not declare such a macro if there was an intention to use that field name.

5. Macros from the same or a related header might use the additional fields in the structure, and those field names might also collide with user macros. Although this is a less frequent occurrence, since macros are expanded at the point of use, no constraint on the order of use of names can apply.

6. An “obvious” solution of using names in the reserved name space and then redefining them as macros when they should be visible does not work because this has the effect of exporting the symbol into the general name space. For example, given a (hypothetical) system-provided header `<h.h>`, and two parts of a C program in **a.c** and **b.c**, in header `<h.h>`:

```
struct foo {
    int __i;
```

```

120583     }
120584     #ifdef _FEATURE_TEST
120585     #define i __i;
120586     #endif

```

120587 In file **a.c**:

```

120588     #include h.h
120589     extern int i;
120590     ...

```

120591 In file **b.c**:

```

120592     extern int i;
120593     ...

```

120594 The symbol that the user thinks of as *i* in both files has an external name of `__i` in **a.c**; the  
 120595 same symbol *i* in **b.c** has an external name *i* (ignoring any hidden manipulations the  
 120596 compiler might perform on the names). This would cause a mysterious name resolution  
 120597 problem when **a.o** and **b.o** are linked.

120598 Simply avoiding definition then causes alignment problems in the structure.

120599 A structure of the form:

```

120600     struct foo {
120601         union {
120602             int __i;
120603             #ifdef _FEATURE_TEST
120604                 int i;
120605             #endif
120606         } __ii;
120607     }

```

120608 does not work because the name of the logical field *i* is `__ii.i`, and introduction of a macro  
 120609 to restore the logical name immediately reintroduces the problem discussed previously  
 120610 (although its manifestation might be more immediate because a syntax error would result  
 120611 if a recursive macro did not cause it to fail first).

120612 7. A more workable solution would be to declare the structure:

```

120613     struct foo {
120614         #ifdef _FEATURE_TEST
120615             int i;
120616         #else
120617             int __i;
120618         #endif
120619     }

```

120620 However, if a macro (particularly one required by a standard) is to be defined that uses  
 120621 this field, two must be defined: one that uses *i*, the other that uses `__i`. If more than one  
 120622 additional field is used in a macro and they are conditional on distinct combinations of  
 120623 features, the complexity goes up as  $2^n$ .

120624 All this leaves a difficult situation: vendors must provide very complex headers to deal with  
 120625 what is conceptually simple and safe—adding a field to a structure. It is the possibility of user-  
 120626 provided macros with the same name that makes this difficult.

120627 Several alternatives were proposed that involved constraining the user's access to part of the

name space available to the user (as specified by the ISO C standard). In some cases, this was only until all the headers had been included. There were two proposals discussed that failed to achieve consensus:

1. Limiting it for the whole program.
2. Restricting the use of identifiers containing only uppercase letters until after all system headers had been included. It was also pointed out that because macros might wish to access fields of a structure (and macro expansion occurs totally at point of use) restricting names in this way would not protect the macro expansion, and thus the solution was inadequate.

It was finally decided that reservation of symbols would occur, but as constrained.

The current wording also allows the addition of fields to a structure, but requires that user macros of the same name not interfere. This allows vendors to do one of the following:

- Not create the situation (do not extend the structures with user-accessible names or use the solution in (7) above)
- Extend their compilers to allow some way of adding names to structures and macros safely

There are at least two ways that the compiler might be extended: add new preprocessor directives that turn off and on macro expansion for certain symbols (without changing the value of the macro) and a function or lexical operation that suppresses expansion of a word. The latter seems more flexible, particularly because it addresses the problem in macros as well as in declarations.

The following seems to be a possible implementation extension to the C language that will do this: any token that during macro expansion is found to be preceded by three '#' symbols shall not be further expanded in exactly the same way as described for macros that expand to their own name as in Section 3.8.3.4 of the ISO C standard. A vendor may also wish to implement this as an operation that is lexically a function, which might be implemented as:

```
#define __safe_name(x) ###x
```

Using a function notation would insulate vendors from changes in standards until such a functionality is standardized (if ever). Standardization of such a function would be valuable because it would then permit third parties to take advantage of it portably in software they may supply.

The symbols that are “explicitly permitted, but not required by POSIX.1-2008” include those classified below. (That is, the symbols classified below might, but are not required to, be present when `_POSIX_C_SOURCE` is defined to have the value 200809L.)

- Symbols in `<limits.h>` and `<unistd.h>` that are defined to indicate support for options or limits that are constant at compile-time
- Symbols in the name space reserved for the implementation by the ISO C standard
- Symbols in a name space reserved for a particular type of extension (for example, type names ending with `_t` in `<sys/types.h>`)
- Additional members of structures or unions whose names do not reduce the name space reserved for applications

Since both implementations and future versions of this standard and other POSIX standards may use symbols in the reserved spaces described in these tables, there is a potential for name space clashes. To avoid future name space clashes when adding symbols, implementations should not use the `posix_`, `POSIX_`, or `_POSIX_` prefixes.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/2 is applied, deleting the entries `POSIX_`, `_POSIX_`, and `posix_` from the column of allowed name space prefixes for use by an implementation in the first table. The presence of these prefixes was contradicting later text which states that: “The prefixes `posix_`, `POSIX_`, and `_POSIX_` are reserved for use by XCU Chapter 2 (on page 2335) and other POSIX standards. Implementations may add symbols to the headers shown in the following table, provided the identifiers ... do not use the reserved prefixes `posix_`, `POSIX_`, or `_POSIX_`.”.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/3 is applied, correcting the reserved macro prefix from: “`PRI[a-z]`, `SCN[a-z]`” to: “`PRI[Xa-z]`, `SCN[Xa-z]`” in the second table. The change was needed since the ISO C standard allows implementations to define macros of the form `PRI` or `SCN` followed by any lowercase letter or ‘`x`’ in `<inttypes.h>`. (The ISO/IEC 9899:1999 standard, Subclause 7.26.4.)

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/4 is applied, adding a new section listing reserved names for the `<stdint.h>` header. This change is for alignment with the ISO C standard.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/2 is applied, making it clear that implementations are permitted to have symbols with the prefix `_POSIX_` visible in any header.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/3 is applied, updating the table of allowed macro prefixes to include the prefix `FP_[A-Z]` for `<math.h>`. This text is added for consistency with the `<math.h>` reference page in the Base Definitions volume of POSIX.1-2008 which permits additional implementation-defined floating-point classifications.

Austin Group Interpretation 1003.1-2001 #048 is applied, reserving `SEEK_` in the name space.

### 120693 B.2.3 Error Numbers

It was the consensus of the standard developers that to allow the conformance document to state that an error occurs and under what conditions, but to disallow a statement that it never occurs, does not make sense. It could be implied by the current wording that this is allowed, but to reduce the possibility of future interpretation requests, it is better to make an explicit statement.

The ISO C standard requires that `errno` be an assignable lvalue. Originally, the definition in POSIX.1 was stricter than that in the ISO C standard, `extern int errno`, in order to support historical usage. In a multi-threaded environment, implementing `errno` as a global variable results in non-deterministic results when accessed. It is required, however, that `errno` work as a per-thread error reporting mechanism. In order to do this, a separate `errno` value has to be maintained for each thread. The following section discusses the various alternative solutions that were considered.

In order to avoid this problem altogether for new functions, these functions avoid using `errno` and, instead, return the error number directly as the function return value; a return value of zero indicates that no error was detected.

For any function that can return errors, the function return value is not used for any purpose other than for reporting errors. Even when the output of the function is scalar, it is passed through a function argument. While it might have been possible to allow some scalar outputs to be coded as negative function return values and mixed in with positive error status returns, this was rejected—using the return value for a mixed purpose was judged to be of limited use and error prone.

Checking the value of `errno` alone is not sufficient to determine the existence or type of an error, since it is not required that a successful function call clear `errno`. The variable `errno` should only be examined when the return value of a function indicates that the value of `errno` is meaningful. In that case, the function is required to set the variable to something other than zero.



120718 The variable *errno* is never set to zero by any function call; to do so would contradict the ISO C  
120719 standard.

120720 POSIX.1 requires (in the ERRORS sections of function descriptions) certain error values to be set  
120721 in certain conditions because many existing applications depend on them. Some error numbers,  
120722 such as [EFAULT], are entirely implementation-defined and are noted as such in their  
120723 description in the ERRORS section. This section otherwise allows wide latitude to the  
120724 implementation in handling error reporting.

120725 Some of the ERRORS sections in POSIX.1-2008 have two subsections. The first:

120726 “The function shall fail if:”

120727 could be called the “mandatory” section.

120728 The second:

120729 “The function may fail if:”

120730 could be informally known as the “optional” section.

120731 Attempting to infer the quality of an implementation based on whether it detects optional error  
120732 conditions is not useful.

120733 Following each one-word symbolic name for an error, there is a description of the error. The  
120734 rationale for some of the symbolic names follows:

120735 [ECANCELED] This spelling was chosen as being more common.

120736 [EFAULT] Most historical implementations do not catch an error and set *errno* when an  
120737 invalid address is given to the functions *wait()*, *time()*, or *times()*. Some  
120738 implementations cannot reliably detect an invalid address. And most systems  
120739 that detect invalid addresses will do so only for a system call, not for a library  
120740 routine.

120741 [EFTYPE] This error code was proposed in earlier proposals as “Inappropriate operation  
120742 for file type”, meaning that the operation requested is not appropriate for the  
120743 file specified in the function call. This code was proposed, although the same  
120744 idea was covered by [ENOTTY], because the connotations of the name would  
120745 be misleading. It was pointed out that the *fcntl()* function uses the error code  
120746 [EINVAL] for this notion, and hence all instances of [EFTYPE] were changed  
120747 to this code.

120748 [EINTR] POSIX.1 prohibits conforming implementations from restarting interrupted  
120749 system calls of conforming applications unless the SA\_RESTART flag is in  
120750 effect for the signal. However, it does not require that [EINTR] be returned  
120751 when another legitimate value may be substituted; for example, a partial  
120752 transfer count when *read()* or *write()* are interrupted. This is only given when  
120753 the signal-catching function returns normally as opposed to returns by  
120754 mechanisms like *longjmp()* or *siglongjmp()*.

120755 [ELOOP] In specifying conditions under which implementations would generate this  
120756 error, the following goals were considered:

120757 • To ensure that actual loops are detected, including loops that result from  
120758 symbolic links across distributed file systems.

120759 • To ensure that during pathname resolution an application can rely on  
120760 the ability to follow at least {SYMLOOP\_MAX} symbolic links in the  
120761 absence of a loop.

120762		• To allow implementations to provide the capability of traversing more
120763		than {SYMLOOP_MAX} symbolic links in the absence of a loop.
120764		• To allow implementations to detect loops and generate the error prior to
120765		encountering {SYMLOOP_MAX} symbolic links.
120766	[ENAMETOOLONG]	
120767		When a symbolic link is encountered during pathname resolution, the
120768		contents of that symbolic link are used to create a new pathname. The
120769		standard developers intended to allow, but not require, that implementations
120770		enforce the restriction of {PATH_MAX} on the result of this pathname
120771		substitution.
120772		Implementations are allowed, but not required, to treat a pathname longer
120773		than {PATH_MAX} passed into the system as an error. Implementations are
120774		required to return a pathname (even if it is longer than {PATH_MAX}) when
120775		the user supplies a buffer with an interface that specifies the buffer size, as
120776		long as the user-supplied buffer is large enough to hold the entire pathname
120777		(see XSH <i>getcwd()</i> for an example of this type of interface). Implementations
120778		are required to treat a request to pass a pathname longer than {PATH_MAX}
120779		from the system to a user-supplied buffer of an unspecified size (usually
120780		assumed to be of size {PATH_MAX}) as an error (see XSH <i>realpath()</i> for an
120781		example of this type of interface).
120782	[ENOMEM]	The term “main memory” is not used in POSIX.1 because it is
120783		implementation-defined.
120784	[ENOTSUP]	This error code is to be used when an implementation chooses to implement
120785		the required functionality of POSIX.1-2008 but does not support optional
120786		facilities defined by POSIX.1-2008. In some earlier versions of this standard,
120787		the difference between [ENOTSUP] and [ENOSYS] was that [ENOSYS]
120788		indicated that the function was not supported at all. This is no longer the case
120789		as [ENOSYS] can also be used to indicate non-support of optional
120790		functionality for a function that has some required functionality. (See XSH
120791		<i>encrypt()</i> .)
120792	[ENOTTY]	The symbolic name for this error is derived from a time when device control
120793		was done by <i>ioctl()</i> and that operation was only permitted on a terminal
120794		interface. The term “TTY” is derived from “teletypewriter”, the devices to
120795		which this error originally applied.
120796	[EOVERFLOW]	Most of the uses of this error code are related to large file support. Typically,
120797		these cases occur on systems which support multiple programming
120798		environments with different sizes for <b>off_t</b> , but they may also occur in
120799		connection with remote file systems.
120800		In addition, when different programming environments have different widths
120801		for types such as <b>int</b> and <b>uid_t</b> , several functions may encounter a condition
120802		where a value in a particular environment is too wide to be represented. In
120803		that case, this error should be raised. For example, suppose the currently
120804		running process has 64-bit <b>int</b> , and file descriptor 9 223 372 036 854 775 807 is
120805		open and does not have the close-on-exec flag set. If the process then uses
120806		<i>execl()</i> to <i>exec</i> a file compiled in a programming environment with 32-bit <b>int</b> ,
120807		the call to <i>execl()</i> can fail with <i>errno</i> set to [EOVERFLOW]. A similar failure
120808		can occur with <i>execl()</i> if any of the user IDs or any of the group IDs to be
120809		assigned to the new process image are out of range for the executed file’s
120810		programming environment.

120811 Note, however, that this condition cannot occur for functions that are  
120812 explicitly described as always being successful, such as *getpid()*.

120813 [EPIPE] This condition normally generates the signal SIGPIPE; the error is returned if  
120814 the generation of the signal is suppressed or the signal does not terminate the  
120815 process.

120816 [EROFS] In historical implementations, attempting to *unlink()* or *rmdir()* a mount point  
120817 would generate an [EBUSY] error. An implementation could be envisioned  
120818 where such an operation could be performed without error. In this case, if  
120819 *either* the directory entry or the actual data structures reside on a read-only file  
120820 system, [EROFS] is the appropriate error to generate. (For example, changing  
120821 the link count of a file on a read-only file system could not be done, as is  
120822 required by *unlink()*, and thus an error should be reported.)

120823 Three error numbers, [EDOM], [EILSEQ], and [ERANGE], were added to this section primarily  
120824 for consistency with the ISO C standard.

#### 120825 **Alternative Solutions for Per-Thread *errno***

120826 The usual implementation of *errno* as a single global variable does not work in a multi-threaded  
120827 environment. In such an environment, a thread may make a POSIX.1 call and get a -1 error  
120828 return, but before that thread can check the value of *errno*, another thread might have made a  
120829 second POSIX.1 call that also set *errno*. This behavior is unacceptable in robust programs. There  
120830 were a number of alternatives that were considered for handling the *errno* problem:

- 120831 • Implement *errno* as a per-thread integer variable.
- 120832 • Implement *errno* as a service that can access the per-thread error number.
- 120833 • Change all POSIX.1 calls to accept an extra status argument and avoid setting *errno*.
- 120834 • Change all POSIX.1 calls to raise a language exception.

120835 The first option offers the highest level of compatibility with existing practice but requires  
120836 special support in the linker, compiler, and/or virtual memory system to support the new  
120837 concept of thread private variables. When compared with current practice, the third and fourth  
120838 options are much cleaner, more efficient, and encourage a more robust programming style, but  
120839 they require new versions of all of the POSIX.1 functions that might detect an error. The second  
120840 option offers compatibility with existing code that uses the **<errno.h>** header to define the  
120841 symbol *errno*. In this option, *errno* may be a macro defined:

```
120842 #define errno  (*__errno())
120843 extern int      *__errno();
```

120844 This option may be implemented as a per-thread variable whereby an *errno* field is allocated in  
120845 the user space object representing a thread, and whereby the function *\_\_errno()* makes a system  
120846 call to determine the location of its user space object and returns the address of the *errno* field of  
120847 that object. Another implementation, one that avoids calling the kernel, involves allocating  
120848 stacks in chunks. The stack allocator keeps a side table indexed by chunk number containing a  
120849 pointer to the thread object that uses that chunk. The *\_\_errno()* function then looks at the stack  
120850 pointer, determines the chunk number, and uses that as an index into the chunk table to find its  
120851 thread object and thus its private value of *errno*. On most architectures, this can be done in four  
120852 to five instructions. Some compilers may wish to implement *\_\_errno()* inline to improve  
120853 performance.



## Disallowing Return of the [EINTR] Error Code

Many blocking interfaces defined by POSIX.1-2008 may return [EINTR] if interrupted during their execution by a signal handler. Blocking interfaces introduced under the threads functionality do not have this property. Instead, they require that the interface appear to be atomic with respect to interruption. In particular, clients of blocking interfaces need not handle any possible [EINTR] return as a special case since it will never occur. If it is necessary to restart operations or complete incomplete operations following the execution of a signal handler, this is handled by the implementation, rather than by the application.

Requiring applications to handle [EINTR] errors on blocking interfaces has been shown to be a frequent source of often unreproducible bugs, and it adds no compelling value to the available functionality. Thus, blocking interfaces introduced for use by multi-threaded programs do not use this paradigm. In particular, in none of the functions *flockfile()*, *pthread\_cond\_timedwait()*, *pthread\_cond\_wait()*, *pthread\_join()*, *pthread\_mutex\_lock()*, and *sigwait()* did providing [EINTR] returns add value, or even particularly make sense. Thus, these functions do not provide for an [EINTR] return, even when interrupted by a signal handler. The same arguments can be applied to *sem\_wait()*, *sem\_trywait()*, *sigwaitinfo()*, and *sigtimedwait()*, but implementations are permitted to return [EINTR] error codes for these functions for compatibility with earlier versions of this standard. Applications cannot rely on calls to these functions returning [EINTR] error codes when signals are delivered to the calling thread, but they should allow for the possibility.

Austin Group Interpretation 1003.1-2001 #050 is applied, allowing [ENOTSUP] and [EOPNOTSUPP] to be the same values.

### B.2.3.1 Additional Error Numbers

The ISO C standard defines the name space for implementations to add additional error numbers.

## B.2.4 Signal Concepts

Historical implementations of signals, using the *signal()* function, have shortcomings that make them unreliable for many application uses. Because of this, a new signal mechanism, based very closely on the one of 4.2 BSD and 4.3 BSD, was added to POSIX.1.

### Signal Names

The restriction on the actual type used for **sigset\_t** is intended to guarantee that these objects can always be assigned, have their address taken, and be passed as parameters by value. It is not intended that this type be a structure including pointers to other data structures, as that could impact the portability of applications performing such operations. A reasonable implementation could be a structure containing an array of some integer type.

The signals described in POSIX.1-2008 must have unique values so that they may be named as parameters of **case** statements in the body of a C-language **switch** clause. However, implementation-defined signals may have values that overlap with each other or with signals specified in POSIX.1-2008. An example of this is SIGABRT, which traditionally overlaps some other signal, such as SIGIOT.

SIGKILL, SIGTERM, SIGUSR1, and SIGUSR2 are ordinarily generated only through the explicit use of the *kill()* function, although some implementations generate SIGKILL under extraordinary circumstances. SIGTERM is traditionally the default signal sent by the *kill* command.

The signals SIGBUS, SIGEMT, SIGIOT, SIGTRAP, and SIGSYS were omitted from POSIX.1 because their behavior is implementation-defined and could not be adequately categorized. Conforming implementations may deliver these signals, but must document the circumstances under which they are delivered and note any restrictions concerning their delivery. The signals SIGFPE, SIGILL, and SIGSEGV are similar in that they also generally result only from programming errors. They were included in POSIX.1 because they do indicate three relatively well-categorized conditions. They are all defined by the ISO C standard and thus would have to be defined by any system with an ISO C standard binding, even if not explicitly included in POSIX.1.

There is very little that a Conforming POSIX.1 Application can do by catching, ignoring, or masking any of the signals SIGILL, SIGTRAP, SIGIOT, SIGEMT, SIGBUS, SIGSEGV, SIGSYS, or SIGFPE. They will generally be generated by the system only in cases of programming errors. While it may be desirable for some robust code (for example, a library routine) to be able to detect and recover from programming errors in other code, these signals are not nearly sufficient for that purpose. One portable use that does exist for these signals is that a command interpreter can recognize them as the cause of termination of a process (with *wait()*) and print an appropriate message. The mnemonic tags for these signals are derived from their PDP-11 origin.

The signals SIGSTOP, SIGTSTP, SIGTTIN, SIGTTOU, and SIGCONT are provided for job control and are unchanged from 4.2 BSD. The signal SIGCHLD is also typically used by job control shells to detect children that have terminated or, as in 4.2 BSD, stopped.

Some implementations, including System V, have a signal named SIGCLD, which is similar to SIGCHLD in 4.2 BSD. POSIX.1 permits implementations to have a single signal with both names. POSIX.1 carefully specifies ways in which conforming applications can avoid the semantic differences between the two different implementations. The name SIGCHLD was chosen for POSIX.1 because most current application usages of it can remain unchanged in conforming applications. SIGCLD in System V has more cases of semantics that POSIX.1 does not specify, and thus applications using it are more likely to require changes in addition to the name change.

The signals SIGUSR1 and SIGUSR2 are commonly used by applications for notification of exceptional behavior and are described as “reserved as application-defined” so that such use is not prohibited. Implementations should not generate SIGUSR1 or SIGUSR2, except when explicitly requested by *kill()*. It is recommended that libraries not use these two signals, as such use in libraries could interfere with their use by applications calling the libraries. If such use is unavoidable, it should be documented. It is prudent for non-portable libraries to use non-standard signals to avoid conflicts with use of standard signals by portable libraries.

There is no portable way for an application to catch or ignore non-standard signals. Some implementations define the range of signal numbers, so applications can install signal-catching functions for all of them. Unfortunately, implementation-defined signals often cause problems when caught or ignored by applications that do not understand the reason for the signal. While the desire exists for an application to be more robust by handling all possible signals (even those only generated by *kill()*), no existing mechanism was found to be sufficiently portable to include in POSIX.1. The value of such a mechanism, if included, would be diminished given that SIGKILL would still not be catchable.

A number of new signal numbers are reserved for applications because the two user signals defined by POSIX.1 are insufficient for many realtime applications. A range of signal numbers is specified, rather than an enumeration of additional reserved signal names, because different applications and application profiles will require a different number of application signals. It is not desirable to burden all application domains and therefore all implementations with the maximum number of signals required by all possible applications. Note that in this context, signal numbers are essentially different signal priorities.

120948 The relatively small number of required additional signals, `{_POSIX_RTSIG_MAX}`, was chosen  
 120949 so as not to require an unreasonably large signal mask/set. While this number of signals  
 120950 defined in POSIX.1 will fit in a single 32-bit word signal mask, it is recognized that most existing  
 120951 implementations define many more signals than are specified in POSIX.1 and, in fact, many  
 120952 implementations have already exceeded 32 signals (including the “null signal”). Support of  
 120953 `{_POSIX_RTSIG_MAX}` additional signals may push some implementation over the single 32-bit  
 120954 word line, but is unlikely to push any implementations that are already over that line beyond  
 120955 the 64-signal line.

#### 120956 B.2.4.1 Signal Generation and Delivery

120957 The terms defined in this section are not used consistently in documentation of historical  
 120958 systems. Each signal can be considered to have a lifetime beginning with generation and ending  
 120959 with delivery or acceptance. The POSIX.1 definition of “delivery” does not exclude ignored  
 120960 signals; this is considered a more consistent definition. This revised text in several parts of  
 120961 POSIX.1-2008 clarifies the distinct semantics of asynchronous signal delivery and synchronous  
 120962 signal acceptance. The previous wording attempted to categorize both under the term  
 120963 “delivery”, which led to conflicts over whether the effects of asynchronous signal delivery  
 120964 applied to synchronous signal acceptance.

120965 Signals generated for a process are delivered to only one thread. Thus, if more than one thread  
 120966 is eligible to receive a signal, one has to be chosen. The choice of threads is left entirely up to the  
 120967 implementation both to allow the widest possible range of conforming implementations and to  
 120968 give implementations the freedom to deliver the signal to the “easiest possible” thread should  
 120969 there be differences in ease of delivery between different threads.

120970 Note that should multiple delivery among cooperating threads be required by an application,  
 120971 this can be trivially constructed out of the provided single-delivery semantics. The construction  
 120972 of a *sigwait\_multiple()* function that accomplishes this goal is presented with the rationale for  
 120973 *sigwaitinfo()*.

120974 Implementations should deliver unblocked signals as soon after they are generated as possible.  
 120975 However, it is difficult for POSIX.1 to make specific requirements about this, beyond those in  
 120976 *kill()* and *sigprocmask()*. Even on systems with prompt delivery, scheduling of higher priority  
 120977 processes is always likely to cause delays.

120978 In general, the interval between the generation and delivery of unblocked signals cannot be  
 120979 detected by an application. Thus, references to pending signals generally apply to blocked,  
 120980 pending signals. An implementation registers a signal as pending on the process when no  
 120981 thread has the signal unblocked and there are no threads blocked in a *sigwait()* function for that  
 120982 signal. Thereafter, the implementation delivers the signal to the first thread that unblocks the  
 120983 signal or calls a *sigwait()* function on a signal set containing this signal rather than choosing the  
 120984 recipient thread at the time the signal is sent.

120985 In the 4.3 BSD system, signals that are blocked and set to `SIG_IGN` are discarded immediately  
 120986 upon generation. For a signal that is ignored as its default action, if the action is `SIG_DFL` and  
 120987 the signal is blocked, a generated signal remains pending. In the 4.1 BSD system and in  
 120988 System V Release 3 (two other implementations that support a somewhat similar signal  
 120989 mechanism), all ignored blocked signals remain pending if generated. Because it is not normally  
 120990 useful for an application to simultaneously ignore and block the same signal, it was unnecessary  
 120991 for POSIX.1 to specify behavior that would invalidate any of the historical implementations.

120992 There is one case in some historical implementations where an unblocked, pending signal does  
 120993 not remain pending until it is delivered. In the System V implementation of *signal()*, pending  
 120994 signals are discarded when the action is set to `SIG_DFL` or a signal-catching routine (as well as  
 120995 to `SIG_IGN`). Except in the case of setting `SIGCHLD` to `SIG_DFL`, implementations that do this

120996 do not conform completely to POSIX.1. Some earlier proposals for POSIX.1 explicitly stated this,  
 120997 but these statements were redundant due to the requirement that functions defined by POSIX.1  
 120998 not change attributes of processes defined by POSIX.1 except as explicitly stated.

120999 POSIX.1 specifically states that the order in which multiple, simultaneously pending signals are  
 121000 delivered is unspecified. This order has not been explicitly specified in historical  
 121001 implementations, but has remained quite consistent and been known to those familiar with the  
 121002 implementations. Thus, there have been cases where applications (usually system utilities) have  
 121003 been written with explicit or implicit dependencies on this order. Implementors and others  
 121004 porting existing applications may need to be aware of such dependencies.

121005 When there are multiple pending signals that are not blocked, implementations should arrange  
 121006 for the delivery of all signals at once, if possible. Some implementations stack calls to all pending  
 121007 signal-catching routines, making it appear that each signal-catcher was interrupted by the next  
 121008 signal. In this case, the implementation should ensure that this stacking of signals does not  
 121009 violate the semantics of the signal masks established by *sigaction()*. Other implementations  
 121010 process at most one signal when the operating system is entered, with remaining signals saved  
 121011 for later delivery. Although this practice is widespread, this behavior is neither standardized  
 121012 nor endorsed. In either case, implementations should attempt to deliver signals associated with  
 121013 the current state of the process (for example, SIGFPE) before other signals, if possible.

121014 In 4.2 BSD and 4.3 BSD, it is not permissible to ignore or explicitly block SIGCONT, because if  
 121015 blocking or ignoring this signal prevented it from continuing a stopped process, such a process  
 121016 could never be continued (only killed by SIGKILL). However, 4.2 BSD and 4.3 BSD do block  
 121017 SIGCONT during execution of its signal-catching function when it is caught, creating exactly  
 121018 this problem. A proposal was considered to disallow catching SIGCONT in addition to ignoring  
 121019 and blocking it, but this limitation led to objections. The consensus was to require that  
 121020 SIGCONT always continue a stopped process when generated. This removed the need to  
 121021 disallow ignoring or explicit blocking of the signal; note that SIG\_IGN and SIG\_DFL are  
 121022 equivalent for SIGCONT.

#### 121023 B.2.4.2 Realtime Signal Generation and Delivery

121024 The realtime signals functionality is required in this version of the standard for the following  
 121025 reasons:

- 121026 • The **sigevent** structure is used by other POSIX.1 functions that result in asynchronous  
 121027 event notifications to specify the notification mechanism to use and other information  
 121028 needed by the notification mechanism. POSIX.1-2008 defines only three symbolic values  
 121029 for the notification mechanism:

- 121030 — SIGEV\_NONE is used to indicate that no notification is required when the event  
 121031 occurs. This is useful for applications that use asynchronous I/O with polling for  
 121032 completion.

- 121033 — SIGEV\_SIGNAL indicates that a signal is generated when the event occurs.

- 121034 — SIGEV\_THREAD provides for “callback functions” for asynchronous notifications  
 121035 done by a function call within the context of a new thread. This provides a multi-  
 121036 threaded process with a more natural means of notification than signals.

121037 The primary difficulty with previous notification approaches has been to specify the  
 121038 environment of the notification routine.

- 121039 — One approach is to limit the notification routine to call only functions permitted in a  
 121040 signal handler. While the list of permissible functions is clearly stated, this is overly  
 121041 restrictive.

121042 — A second approach is to define a new list of functions or classes of functions that are  
121043 explicitly permitted or not permitted. This would give a programmer more lists to  
121044 deal with, which would be awkward.

121045 — The third approach is to define completely the environment for execution of the  
121046 notification function. A clear definition of an execution environment for notification  
121047 is provided by executing the notification function in the environment of a newly  
121048 created thread.

121049 Implementations may support additional notification mechanisms by defining new values  
121050 for *sigev\_notify*.

121051 For a notification type of SIGEV\_SIGNAL, the other members of the **sigevent** structure  
121052 defined by POSIX.1-2008 specify the realtime signal—that is, the signal number and  
121053 application-defined value that differentiates between occurrences of signals with the same  
121054 number—that will be generated when the event occurs. The structure is defined in  
121055 **<signal.h>**, even though the structure is not directly used by any of the signal functions,  
121056 because it is part of the signals interface used by the POSIX.1b “client functions”. When the  
121057 client functions include **<signal.h>** to define the signal names, the **sigevent** structure will  
121058 also be defined.

121059 An application-defined value passed to the signal handler is used to differentiate between  
121060 different “events” instead of requiring that the application use different signal numbers for  
121061 several reasons:

121062 — Realtime applications potentially handle a very large number of different events.  
121063 Requiring that implementations support a correspondingly large number of distinct  
121064 signal numbers will adversely impact the performance of signal delivery because the  
121065 signal masks to be manipulated on entry and exit to the handlers will become large.

121066 — Event notifications are prioritized by signal number (the rationale for this is  
121067 explained in the following paragraphs) and the use of different signal numbers to  
121068 differentiate between the different event notifications overloads the signal number  
121069 more than has already been done. It also requires that the application developer  
121070 make arbitrary assignments of priority to events that are logically of equal priority.

121071 A union is defined for the application-defined value so that either an integer constant or a  
121072 pointer can be portably passed to the signal-catching function. On some architectures a  
121073 pointer cannot be cast to an **int** and *vice versa*.

121074 Use of a structure here with an explicit notification type discriminant rather than explicit  
121075 parameters to realtime functions, or embedded in other realtime structures, provides for  
121076 future extensions to POSIX.1-2008. Additional, perhaps more efficient, notification  
121077 mechanisms can be supported for existing realtime function interfaces, such as timers and  
121078 asynchronous I/O, by extending the **sigevent** structure appropriately. The existing  
121079 realtime function interfaces will not have to be modified to use any such new notification  
121080 mechanism. The revised text concerning the SIGEV\_SIGNAL value makes consistent the  
121081 semantics of the members of the **sigevent** structure, particularly in the definitions of  
121082 *lio\_listio()* and *aio\_fsync()*. For uniformity, other revisions cause this specification to be  
121083 referred to rather than inaccurately duplicated in the descriptions of functions and  
121084 structures using the **sigevent** structure. The revised wording does not relax the  
121085 requirement that the signal number be in the range SIGRTMIN to SIGRTMAX to guarantee  
121086 queuing and passing of the application value, since that requirement is still implied by the  
121087 signal names.



- POSIX.1-2008 is intentionally vague on whether “non-realtime” signal-generating mechanisms can result in a **siginfo\_t** being supplied to the handler on delivery. In one existing implementation, a **siginfo\_t** is posted on signal generation, even though the implementation does not support queuing of multiple occurrences of a signal. It is not the intent of POSIX.1-2008 to preclude this, independent of the mandate to define signals that do support queuing. Any interpretation that appears to preclude this is a mistake in the reading or writing of the standard.

- Signals handled by realtime signal handlers might be generated by functions or conditions that do not allow the specification of an application-defined value and do not queue. POSIX.1-2008 specifies the *si\_code* member of the **siginfo\_t** structure used in existing practice and defines additional codes so that applications can detect whether an application-defined value is present or not. The code `SI_USER` for *kill()*-generated signals is adopted from existing practice.

- The *sigaction()* *sa\_flags* value `SA_SIGINFO` tells the implementation that the signal-catching function expects two additional arguments. When the flag is not set, a single argument, the signal number, is passed as specified by POSIX.1-2008. Although POSIX.1-2008 does not explicitly allow the *info* argument to the handler function to be `NULL`, this is existing practice. This provides for compatibility with programs whose signal-catching functions are not prepared to accept the additional arguments. POSIX.1-2008 is explicitly unspecified as to whether signals actually queue when `SA_SIGINFO` is not set for a signal, as there appear to be no benefits to applications in specifying one behavior or another. One existing implementation queues a **siginfo\_t** on each signal generation, unless the signal is already pending, in which case the implementation discards the new **siginfo\_t**; that is, the queue length is never greater than one. This implementation only examines `SA_SIGINFO` on signal delivery, discarding the queued **siginfo\_t** if its delivery was not requested.

The third argument to the signal-catching function, *context*, is left undefined by POSIX.1-2008, but is specified in the interface because it matches existing practice for the `SA_SIGINFO` flag. It was considered undesirable to require a separate implementation for `SA_SIGINFO` for POSIX conformance on implementations that already support the two additional parameters.

- The requirement to deliver lower numbered signals in the range `SIGRTMIN` to `SIGRTMAX` first, when multiple unblocked signals are pending, results from several considerations:

- A method is required to prioritize event notifications. The signal number was chosen instead of, for instance, associating a separate priority with each request, because an implementation has to check pending signals at various points and select one for delivery when more than one is pending. Specifying a selection order is the minimal additional semantic that will achieve prioritized delivery. If a separate priority were to be associated with queued signals, it would be necessary for an implementation to search all non-empty, non-blocked signal queues and select from among them the pending signal with the highest priority. This would significantly increase the cost of and decrease the determinism of signal delivery.

- Given the specified selection of the lowest numeric unblocked pending signal, preemptive priority signal delivery can be achieved using signal numbers and signal masks by ensuring that the *sa\_mask* for each signal number blocks all signals with a higher numeric value.

For realtime applications that want to use only the newly defined realtime signal numbers without interference from the standard signals, this can be achieved by blocking all of the standard signals in the thread signal mask and in the *sa\_mask*

121137 installed by the signal action for the realtime signal handlers.

121138 POSIX.1-2008 explicitly leaves unspecified the ordering of signals outside of the range of  
121139 realtime signals and the ordering of signals within this range with respect to those outside  
121140 the range. It was believed that this would unduly constrain implementations or standards  
121141 in the future definition of new signals.

121142 *B.2.4.3 Signal Actions*

121143 Early proposals mentioned SIGCONT as a second exception to the rule that signals are not  
121144 delivered to stopped processes until continued. Because POSIX.1-2008 now specifies that  
121145 SIGCONT causes the stopped process to continue when it is generated, delivery of SIGCONT is  
121146 not prevented because a process is stopped, even without an explicit exception to this rule.

121147 Ignoring a signal by setting the action to SIG\_IGN (or SIG\_DFL for signals whose default action  
121148 is to ignore) is not the same as installing a signal-catching function that simply returns. Invoking  
121149 such a function will interrupt certain system functions that block processes (for example, *wait()*,  
121150 *sigsuspend()*, *pause()*, *read()*, *write()*) while ignoring a signal has no such effect on the process.

121151 Historical implementations discard pending signals when the action is set to SIG\_IGN.  
121152 However, they do not always do the same when the action is set to SIG\_DFL and the default  
121153 action is to ignore the signal. POSIX.1-2008 requires this for the sake of consistency and also for  
121154 completeness, since the only signal this applies to is SIGCHLD, and POSIX.1-2008 disallows  
121155 setting its action to SIG\_IGN.

121156 Some implementations (System V, for example) assign different semantics for SIGCHLD  
121157 depending on whether the action is set to SIG\_IGN or SIG\_DFL. Since POSIX.1 requires that the  
121158 default action for SIGCHLD be to ignore the signal, applications should always set the action to  
121159 SIG\_DFL in order to avoid SIGCHLD.

121160 Whether or not an implementation allows SIG\_IGN as a SIGCHLD disposition to be inherited  
121161 across a call to one of the *exec* family of functions or *posix\_spawn()* is explicitly left as  
121162 unspecified. This change was made as a result of IEEE PASC Interpretation 1003.1 #132, and  
121163 permits the implementation to decide between the following alternatives:

- 121164 • Unconditionally leave SIGCHLD set to SIG\_IGN, in which case the implementation would  
121165 not allow applications that assume inheritance of SIG\_DFL to conform to POSIX.1-2008  
121166 without change. The implementation would, however, retain an ability to control  
121167 applications that create child processes but never call on the *wait* family of functions,  
121168 potentially filling up the process table.
- 121169 • Unconditionally reset SIGCHLD to SIG\_DFL, in which case the implementation would  
121170 allow applications that assume inheritance of SIG\_DFL to conform. The implementation  
121171 would, however, lose an ability to control applications that spawn child processes but  
121172 never reap them.
- 121173 • Provide some mechanism, not specified in POSIX.1-2008, to control inherited SIGCHLD  
121174 dispositions.

121175 Some implementations (System V, for example) will deliver a SIGCHLD signal immediately when  
121176 a process establishes a signal-catching function for SIGCHLD when that process has a child that  
121177 has already terminated. Other implementations, such as 4.3 BSD, do not generate a new  
121178 SIGCHLD signal in this way. In general, a process should not attempt to alter the signal action  
121179 for the SIGCHLD signal while it has any outstanding children. However, it is not always  
121180 possible for a process to avoid this; for example, shells sometimes start up processes in pipelines  
121181 with other processes from the pipeline as children. Processes that cannot ensure that they have  
121182 no children when altering the signal action for SIGCHLD thus need to be prepared for, but not

121183 depend on, generation of an immediate SIGCHLD signal.

121184 The default action of the stop signals (SIGSTOP, SIGTSTP, SIGTTIN, SIGTTOU) is to stop a  
121185 process that is executing. If a stop signal is delivered to a process that is already stopped, it has  
121186 no effect. In fact, if a stop signal is generated for a stopped process whose signal mask blocks the  
121187 signal, the signal will never be delivered to the process since the process must receive a  
121188 SIGCONT, which discards all pending stop signals, in order to continue executing.

121189 The SIGCONT signal continues a stopped process even if SIGCONT is blocked (or ignored).  
121190 However, if a signal-catching routine has been established for SIGCONT, it will not be entered  
121191 until SIGCONT is unblocked.

121192 If a process in an orphaned process group stops, it is no longer under the control of a job control  
121193 shell and hence would not normally ever be continued. Because of this, orphaned processes that  
121194 receive terminal-related stop signals (SIGTSTP, SIGTTIN, SIGTTOU, but not SIGSTOP) must not  
121195 be allowed to stop. The goal is to prevent stopped processes from languishing forever. (As  
121196 SIGSTOP is sent only via *kill()*, it is assumed that the process or user sending a SIGSTOP can  
121197 send a SIGCONT when desired.) Instead, the system must discard the stop signal. As an  
121198 extension, it may also deliver another signal in its place. 4.3 BSD sends a SIGKILL, which is  
121199 overly effective because SIGKILL is not catchable. Another possible choice is SIGHUP. 4.3 BSD  
121200 also does this for orphaned processes (processes whose parent has terminated) rather than for  
121201 members of orphaned process groups; this is less desirable because job control shells manage  
121202 process groups. POSIX.1 also prevents SIGTTIN and SIGTTOU signals from being generated for  
121203 processes in orphaned process groups as a direct result of activity on a terminal, preventing  
121204 infinite loops when *read()* and *write()* calls generate signals that are discarded; see [Section](#)  
121205 [A.11.1.4](#) (on page 3539). A similar restriction on the generation of SIGTSTP was considered, but  
121206 that would be unnecessary and more difficult to implement due to its asynchronous nature.

121207 Although POSIX.1 requires that signal-catching functions be called with only one argument,  
121208 there is nothing to prevent conforming implementations from extending POSIX.1 to pass  
121209 additional arguments, as long as Strictly Conforming POSIX.1 Applications continue to compile  
121210 and execute correctly. Most historical implementations do, in fact, pass additional, signal-  
121211 specific arguments to certain signal-catching routines.

121212 There was a proposal to change the declared type of the signal handler to:

121213 `void func (int sig, ...);`

121214 The usage of ellipses ("...") is ISO C standard syntax to indicate a variable number of  
121215 arguments. Its use was intended to allow the implementation to pass additional information to  
121216 the signal handler in a standard manner.

121217 Unfortunately, this construct would require all signal handlers to be defined with this syntax  
121218 because the ISO C standard allows implementations to use a different parameter passing  
121219 mechanism for variable parameter lists than for non-variable parameter lists. Thus, all existing  
121220 signal handlers in all existing applications would have to be changed to use the variable syntax  
121221 in order to be standard and portable. This is in conflict with the goal of Minimal Changes to  
121222 Existing Application Code.

121223 When terminating a process from a signal-catching function, processes should be aware of any  
121224 interpretation that their parent may make of the status returned by *wait()*, *waitid()*, or *waitpid()*.  
121225 In particular, a signal-catching function should not call *exit(0)* or *\_exit(0)* unless it wants to  
121226 indicate successful termination. A non-zero argument to *exit()* or *\_exit()* can be used to indicate  
121227 unsuccessful termination. Alternatively, the process can use *kill()* to send itself a fatal signal  
121228 (first ensuring that the signal is set to the default action and not blocked). See also the  
121229 RATIONALE section of the *\_exit()* function.

121230 The behavior of *unsafe* functions, as defined by this section, is undefined when they are called |



from (or after a *longjmp()* or *siglongjmp()* out of) signal-catching functions in certain circumstances. The behavior of async-signal-safe functions, as defined by this section, is as specified by POSIX.1, regardless of invocation from a signal-catching function. This is the only intended meaning of the statement that async-signal-safe functions may be used in signal-catching functions without restriction. Applications must still consider all effects of such functions on such things as data structures, files, and process state. In particular, application developers need to consider the restrictions on interactions when interrupting *sleep()* (see *sleep()*) and interactions among multiple handles for a file description. The fact that any specific function is listed as async-signal-safe does not necessarily mean that invocation of that function from a signal-catching function is recommended.

In order to prevent errors arising from interrupting non-async-signal-safe function calls, applications should protect calls to these functions either by blocking the appropriate signals or through the use of some programmatic semaphore. POSIX.1 does not address the more general problem of synchronizing access to shared data structures. Note in particular that even the “safe” functions may modify the global variable *errno*; the signal-catching function may want to save and restore its value. The same principles apply to the async-signal-safety of application routines and asynchronous data access.

Note that although *longjmp()* and *siglongjmp()* are in the list of async-signal-safe functions, there are restrictions on subsequent behavior after the function is called from a signal-catching function. This is because the code executing after *longjmp()* or *siglongjmp()* can call any unsafe functions with the same danger as calling those unsafe functions directly from the signal handler. Applications that use *longjmp()* or *siglongjmp()* out of signal handlers require rigorous protection in order to be portable. Many of the other functions that are excluded from the list are traditionally implemented using either the C language *malloc()* or *free()* functions or the ISO C standard I/O library, both of which traditionally use data structures in a non-async-signal-safe manner. Because any combination of different functions using a common data structure can cause async-signal-safety problems, POSIX.1 does not define the behavior when any unsafe function is called in (or after a *longjmp()* or *siglongjmp()* out of) a signal handler that interrupts any unsafe function or the non-async-signal-safe processing equivalent to *exit()* that is performed after return from the initial call to *main()*.

The only realtime extension to signal actions is the addition of the additional parameters to the signal-catching function. This extension has been explained and motivated in the previous section. In making this extension, though, developers of POSIX.1b ran into issues relating to function prototypes. In response to input from the POSIX.1 standard developers, members were added to the **sigaction** structure to specify function prototypes for the newer signal-catching function specified by POSIX.1b. These members follow changes that are being made to POSIX.1. Note that POSIX.1-2008 explicitly states that these fields may overlap so that a union can be defined. This enabled existing implementations of POSIX.1 to maintain binary-compatibility when these extensions were added.

The **siginfo\_t** structure was adopted for passing the application-defined value to match existing practice, but the existing practice has no provision for an application-defined value, so this was added. Note that POSIX normally reserves the “\_t” type designation for opaque types. The **siginfo\_t** structure breaks with this convention to follow existing practice and thus promote portability.

POSIX.1-2008 specifies several values for the *si\_code* member of the **siginfo\_t** structure. Some were introduced in POSIX.1b; others were XSI functionality in the Single UNIX Specification, Version 2 and Version 3, that has now become Base functionality. Historically, an *si\_code* value of less than or equal to zero indicated that the signal was generated by a process via the *kill()* function, and values of *si\_code* that provided additional information for implementation-generated signals, such as SIGFPE or SIGSEGV, were all positive. This functionality is partially

specified for XSI systems in that if *si\_code* is less than or equal to zero, the signal was generated by a process. However, since POSIX.1b did not specify that SI\_USER (or SI\_QUEUE) had a value less than or equal to zero, it is not true that when the signal is generated by a process, the value of *si\_code* will always be less than or equal to zero. XSI applications should check whether *si\_code* is SI\_USER or SI\_QUEUE in addition to checking whether it is less than or equal to zero. Applications on systems that do not support the XSI option should just check for SI\_USER and SI\_QUEUE.

If an implementation chooses to define additional values for *si\_code*, these values have to be different from the values of the non-signal-specific symbols specified by POSIX.1-2008. This will allow conforming applications to differentiate between signals generated by standard events and those generated by other implementation events in a manner compatible with existing practice.

The unique values of *si\_code* for the POSIX.1b asynchronous events have implications for implementations of, for example, asynchronous I/O or message passing in user space library code. Such an implementation will be required to provide a hidden interface to the signal generation mechanism that allows the library to specify the standard values of *si\_code*.

POSIX.1-2008 also specifies additional members of **siginfo\_t**, beyond those that were in POSIX.1b. Like the *si\_code* values mentioned above, these were XSI functionality in the Single UNIX Specification, Version 2 and Version 3, that has now become Base functionality. They provide additional information when *si\_code* has one of the values that moved from XSI to Base.

Although it is not explicitly visible to applications, there are additional semantics for signal actions implied by queued signals and their interaction with other POSIX.1b realtime functions. Specifically:

- It is not necessary to queue signals whose action is SIG\_IGN.
- For implementations that support POSIX.1b timers, some interaction with the timer functions at signal delivery is implied to manage the timer overrun count.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/5 is applied, reordering the RTS shaded text under the third and fourth paragraphs of the SIG\_DFL description. This corrects an earlier editorial error in this section.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/6 is applied, adding the *abort()* function to the list of async-signal-safe functions.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/4 is applied, adding the *socketmark()* function to the list of functions that shall be either reentrant or non-interruptible by signals and shall be async-signal-safe.

#### 121315 B.2.4.4 Signal Effects on Other Functions

The most common behavior of an interrupted function after a signal-catching function returns is for the interrupted function to give an [EINTR] error unless the SA\_RESTART flag is in effect for the signal. However, there are a number of specific exceptions, including *sleep()* and certain situations with *read()* and *write()*.

The historical implementations of many functions defined by POSIX.1-2008 are not interruptible, but delay delivery of signals generated during their execution until after they complete. This is never a problem for functions that are guaranteed to complete in a short (imperceptible to a human) period of time. It is normally those functions that can suspend a process indefinitely or for long periods of time (for example, *wait()*, *pause()*, *sigsuspend()*, *sleep()*, or *read()/write()* on a slow device like a terminal) that are interruptible. This permits applications to respond to interactive signals or to set timeouts on calls to most such functions with *alarm()*. Therefore,

121327 implementations should generally make such functions (including ones defined as extensions)  
121328 interruptible.

121329 Functions not mentioned explicitly as interruptible may be so on some implementations,  
121330 possibly as an extension where the function gives an [EINTR] error. There are several functions  
121331 (for example, *getpid()*, *getuid()*) that are specified as never returning an error, which can thus  
121332 never be extended in this way.

121333 If a signal-catching function returns while the SA\_RESTART flag is in effect, an interrupted  
121334 function is restarted at the point it was interrupted. Conforming applications cannot make  
121335 assumptions about the internal behavior of interrupted functions, even if the functions are  
121336 async-signal-safe. For example, suppose the *read()* function is interrupted with SA\_RESTART in  
121337 effect, the signal-catching function closes the file descriptor being read from and returns, and the  
121338 *read()* function is then restarted; in this case the application cannot assume that the *read()*  
121339 function will give an [EBADF] error, since *read()* might have checked the file descriptor for  
121340 validity before being interrupted.

## 121341 B.2.5 Standard I/O Streams

121342 Although the ISO C standard guarantees that, at program start-up, *stdin* is open for reading and  
121343 *stdout* and *stderr* are open for writing, this guarantee is contingent (as are all guarantees made by  
121344 the ISO C and POSIX standards) on the program being executed in a conforming environment.  
121345 Programs executed with file descriptor 0 not open for reading or with file descriptor 1 or 2 not  
121346 open for writing are executed in a non-conforming environment. Application writers are warned  
121347 (in *exec*, *posix\_spawn()*, and Section C.2.7, on page 3723) not to execute a standard utility or a  
121348 conforming application with file descriptor 0 not open for reading or with file descriptor 1 or 2  
121349 not open for writing.

### 121350 B.2.5.1 Interaction of File Descriptors and Standard I/O Streams

121351 There is no additional rationale provided for this section.

### 121352 B.2.5.2 Stream Orientation and Encoding Rules

121353 There is no additional rationale provided for this section.

## 121354 B.2.6 STREAMS

121355 STREAMS are included into POSIX.1-2008 as part of the alignment with the Single UNIX  
121356 Specification, but marked as an option in recognition that not all systems may wish to  
121357 implement the facility. The option within POSIX.1-2008 is denoted by the XSR margin marker.  
121358 The standard developers made this option independent of the XSI option. In this version of the  
121359 standard this option is marked obsolescent.

121360 STREAMS are a method of implementing network services and other character-based  
121361 input/output mechanisms, with the STREAM being a full-duplex connection between a process  
121362 and a device. STREAMS provides direct access to protocol modules, and optional protocol  
121363 modules can be interposed between the process-end of the STREAM and the device-driver at the  
121364 device-end of the STREAM. Pipes can be implemented using the STREAMS mechanism, so they  
121365 can provide process-to-process as well as process-to-device communications.

121366 This section introduces STREAMS I/O, the message types used to control them, an overview of  
121367 the priority mechanism, and the interfaces used to access them.

121368 B.2.6.1 Accessing STREAMS

121369 There is no additional rationale provided for this section.

121370 **B.2.7 XSI Interprocess Communication**

121371 There are two forms of IPC supported as options in POSIX.1-2008. The traditional System V IPC  
121372 routines derived from the SVID—that is, the *msg\**(), *sem\**(), and *shm\**() interfaces—are  
121373 mandatory on XSI-conformant systems. Thus, all XSI-conformant systems provide the same  
121374 mechanisms for manipulating messages, shared memory, and semaphores.

121375 In addition, the POSIX Realtime Extension provides an alternate set of routines for those systems  
121376 supporting the appropriate options.

121377 The application developer is presented with a choice: the System V interfaces or the POSIX  
121378 interfaces (loosely derived from the Berkeley interfaces). The XSI profile prefers the System V  
121379 interfaces, but the POSIX interfaces may be more suitable for realtime or other performance-  
121380 sensitive applications.

121381 B.2.7.1 IPC General Information

121382 General information that is shared by all three mechanisms is described in this section. The  
121383 common permissions mechanism is briefly introduced, describing the mode bits, and how they  
121384 are used to determine whether or not a process has access to read or write/alter the appropriate  
121385 instance of one of the IPC mechanisms. All other relevant information is contained in the  
121386 reference pages themselves.

121387 The semaphore type of IPC allows processes to communicate through the exchange of  
121388 semaphore values. A semaphore is a positive integer. Since many applications require the use of  
121389 more than one semaphore, XSI-conformant systems have the ability to create sets or arrays of  
121390 semaphores.

121391 Calls to support semaphores include:

121392 *semctl*(), *semget*(), *semop*()

121393 Semaphore sets are created by using the *semget*() function.

121394 The message type of IPC allows processes to communicate through the exchange of data stored  
121395 in buffers. This data is transmitted between processes in discrete portions known as messages.

121396 Calls to support message queues include:

121397 *msgctl*(), *msgget*(), *msgrcv*(), *msgsnd*()

121398 The shared memory type of IPC allows two or more processes to share memory and  
121399 consequently the data contained therein. This is done by allowing processes to set up access to a  
121400 common memory address space. This sharing of memory provides a fast means of exchange of  
121401 data between processes.

121402 Calls to support shared memory include:

121403 *shmctl*(), *shmdt*(), *shmget*()

121404 The *ftok*() interface is also provided.

**B.2.8 Realtime****Advisory Information**

POSIX.1b contains an Informative Annex with proposed interfaces for “realtime files”. These interfaces could determine groups of the exact parameters required to do “direct I/O” or “extents”. These interfaces were objected to by a significant portion of the balloting group as too complex. A conforming application had little chance of correctly navigating the large parameter space to match its desires to the system. In addition, they only applied to a new type of file (realtime files) and they told the implementation exactly what to do as opposed to advising the implementation on application behavior and letting it optimize for the system the (portable) application was running on. For example, it was not clear how a system that had a disk array should set its parameters.

There seemed to be several overall goals:

- Optimizing sequential access
- Optimizing caching behavior
- Optimizing I/O data transfer
- Preallocation

The advisory interfaces, *posix\_fadvise()* and *posix\_madvise()*, satisfy the first two goals. The `POSIX_FADV_SEQUENTIAL` and `POSIX_MADV_SEQUENTIAL` advice tells the implementation to expect serial access. Typically the system will prefetch the next several serial accesses in order to overlap I/O. It may also free previously accessed serial data if memory is tight. If the application is not doing serial access it can use `POSIX_FADV_WILLNEED` and `POSIX_MADV_WILLNEED` to accomplish I/O overlap, as required. When the application advises `POSIX_FADV_RANDOM` or `POSIX_MADV_RANDOM` behavior, the implementation usually tries to fetch a minimum amount of data with each request and it does not expect much locality. `POSIX_FADV_DONTNEED` and `POSIX_MADV_DONTNEED` allow the system to free up caching resources as the data will not be required in the near future.

`POSIX_FADV_NOREUSE` tells the system that caching the specified data is not optimal. For file I/O, the transfer should go directly to the user buffer instead of being cached internally by the implementation. To portably perform direct disk I/O on all systems, the application must perform its I/O transfers according to the following rules:

1. The user buffer should be aligned according to the `{POSIX_REC_XFER_ALIGN} pathconf()` variable.
2. The number of bytes transferred in an I/O operation should be a multiple of the `{POSIX_ALLOC_SIZE_MIN} pathconf()` variable.
3. The offset into the file at the start of an I/O operation should be a multiple of the `{POSIX_ALLOC_SIZE_MIN} pathconf()` variable.
4. The application should ensure that all threads which open a given file specify `POSIX_FADV_NOREUSE` to be sure that there is no unexpected interaction between threads using buffered I/O and threads using direct I/O to the same file.

In some cases, a user buffer must be properly aligned in order to be transferred directly to/from the device. The `{POSIX_REC_XFER_ALIGN} pathconf()` variable tells the application the proper alignment.

The preallocation goal is met by the space control function, *posix\_fallocate()*. The application can use *posix\_fallocate()* to guarantee no `[ENOSPC]` errors and to improve performance by prepaying any overhead required for block allocation.



Implementations may use information conveyed by a previous *posix\_fadvise()* call to influence the manner in which allocation is performed. For example, if an application did the following calls:

```
fd = open("file");
posix_fadvise(fd, offset, len, POSIX_FADV_SEQUENTIAL);
posix_fallocate(fd, len, size);
```

an implementation might allocate the file contiguously on disk.

Finally, the *pathconf()* variables {POSIX\_REC\_MIN\_XFER\_SIZE}, {POSIX\_REC\_MAX\_XFER\_SIZE}, and {POSIX\_REC\_INCR\_XFER\_SIZE} tell the application a range of transfer sizes that are recommended for best I/O performance.

Where bounded response time is required, the vendor can supply the appropriate settings of the advisories to achieve a guaranteed performance level.

The interfaces meet the goals while allowing applications using regular files to take advantage of performance optimizations. The interfaces tell the implementation expected application behavior which the implementation can use to optimize performance on a particular system with a particular dynamic load.

The *posix\_memalign()* function was added to allow for the allocation of specifically aligned buffers; for example, for {POSIX\_REC\_XFER\_ALIGN}.

The working group also considered the alternative of adding a function which would return an aligned pointer to memory within a user-supplied buffer. This was not considered to be the best method, because it potentially wastes large amounts of memory when buffers need to be aligned on large alignment boundaries.

## Message Passing

This section provides the rationale for the definition of the message passing interface in POSIX.1-2008. This is presented in terms of the objectives, models, and requirements imposed upon this interface.

- Objectives

Many applications, including both realtime and database applications, require a means of passing arbitrary amounts of data between cooperating processes comprising the overall application on one or more processors. Many conventional interfaces for interprocess communication are insufficient for realtime applications in that efficient and deterministic data passing methods cannot be implemented. This has prompted the definition of message passing interfaces providing these facilities:

- Open a message queue.
- Send a message to a message queue.
- Receive a message from a queue, either synchronously or asynchronously.
- Alter message queue attributes for flow and resource control.

It is assumed that an application may consist of multiple cooperating processes and that these processes may wish to communicate and coordinate their activities. The message passing facility described in POSIX.1-2008 allows processes to communicate through system-wide queues. These message queues are accessed through names that may be pathnames. A message queue can be opened for use by multiple sending and/or multiple receiving processes.

- 121493 • Background on Embedded Applications
- 121494 Interprocess communication utilizing message passing is a key facility for the construction  
121495 of deterministic, high-performance realtime applications. The facility is present in all  
121496 realtime systems and is the framework upon which the application is constructed. The  
121497 performance of the facility is usually a direct indication of the performance of the resulting  
121498 application.
- 121499 Realtime applications, especially for embedded systems, are typically designed around the  
121500 performance constraints imposed by the message passing mechanisms. Applications for  
121501 embedded systems are typically very tightly constrained. Application developers expect to  
121502 design and control the entire system. In order to minimize system costs, the writer will  
121503 attempt to use all resources to their utmost and minimize the requirement to add  
121504 additional memory or processors.
- 121505 The embedded applications usually share address spaces and only a simple message  
121506 passing mechanism is required. The application can readily access common data incurring  
121507 only mutual-exclusion overheads. The models desired are the simplest possible with the  
121508 application building higher-level facilities only when needed.
- 121509 • Requirements
- 121510 The following requirements determined the features of the message passing facilities  
121511 defined in POSIX.1-2008:
- 121512 — Naming of Message Queues
- 121513 The mechanism for gaining access to a message queue is a pathname evaluated in a  
121514 context that is allowed to be a file system name space, or it can be independent of  
121515 any file system. This is a specific attempt to allow implementations based on either  
121516 method in order to address both embedded systems and to also allow  
121517 implementation in larger systems.
- 121518 The interface of *mq\_open()* is defined to allow but not require the access control and  
121519 name conflicts resulting from utilizing a file system for name resolution. All required  
121520 behavior is specified for the access control case. Yet a conforming implementation,  
121521 such as an embedded system kernel, may define that there are no distinctions  
121522 between users and may define that all processes have all access privileges.
- 121523 — Embedded System Naming
- 121524 Embedded systems need to be able to utilize independent name spaces for accessing  
121525 the various system objects. They typically do not have a file system, precluding its  
121526 utilization as a common name resolution mechanism. The modularity of an  
121527 embedded system limits the connections between separate mechanisms that can be  
121528 allowed.
- 121529 Embedded systems typically do not have any access protection. Since the system  
121530 does not support the mixing of applications from different areas, and usually does  
121531 not even have the concept of an authorization entity, access control is not useful.
- 121532 — Large System Naming
- 121533 On systems with more functionality, the name resolution must support the ability to  
121534 use the file system as the name resolution mechanism/object storage medium and to  
121535 have control over access to the objects. Utilizing the pathname space can result in  
121536 further errors when the names conflict with other objects.

121537 — Fixed Size of Messages

121538 The interfaces impose a fixed upper bound on the size of messages that can be sent to  
121539 a specific message queue. The size is set on an individual queue basis and cannot be  
121540 changed dynamically.

121541 The purpose of the fixed size is to increase the ability of the system to optimize the  
121542 implementation of *mq\_send()* and *mq\_receive()*. With fixed sizes of messages and  
121543 fixed numbers of messages, specific message blocks can be pre-allocated. This  
121544 eliminates a significant amount of checking for errors and boundary conditions.  
121545 Additionally, an implementation can optimize data copying to maximize  
121546 performance. Finally, with a restricted range of message sizes, an implementation is  
121547 better able to provide deterministic operations.

121548 — Prioritization of Messages

121549 Message prioritization allows the application to determine the order in which  
121550 messages are received. Prioritization of messages is a key facility that is provided by  
121551 most realtime kernels and is heavily utilized by the applications. The major purpose  
121552 of having priorities in message queues is to avoid priority inversions in the message  
121553 system, where a high-priority message is delayed behind one or more lower-priority  
121554 messages. This allows the applications to be designed so that they do not need to be  
121555 interrupted in order to change the flow of control when exceptional conditions occur.  
121556 The prioritization does add additional overhead to the message operations in those  
121557 cases it is actually used but a clever implementation can optimize for the FIFO case to  
121558 make that more efficient.

121559 — Asynchronous Notification

121560 The interface supports the ability to have a task asynchronously notified of the  
121561 availability of a message on the queue. The purpose of this facility is to allow the task  
121562 to perform other functions and yet still be notified that a message has become  
121563 available on the queue.

121564 To understand the requirement for this function, it is useful to understand two  
121565 models of application design: a single task performing multiple functions and  
121566 multiple tasks performing a single function. Each of these models has advantages.

121567 Asynchronous notification is required to build the model of a single task performing  
121568 multiple operations. This model typically results from either the expectation that  
121569 interruption is less expensive than utilizing a separate task or from the growth of the  
121570 application to include additional functions.

121571 **Semaphores**

121572 Semaphores are a high-performance process synchronization mechanism. Semaphores are  
121573 named by null-terminated strings of characters.

121574 A semaphore is created using the *sem\_init()* function or the *sem\_open()* function with the  
121575 *O\_CREAT* flag set in *oflag*.

121576 To use a semaphore, a process has to first initialize the semaphore or inherit an open descriptor  
121577 for the semaphore via *fork()*.

121578 A semaphore preserves its state when the last reference is closed. For example, if a semaphore  
121579 has a value of 13 when the last reference is closed, it will have a value of 13 when it is next  
121580 opened.

121581 When a semaphore is created, an initial state for the semaphore has to be provided. This value is



a non-negative integer. Negative values are not possible since they indicate the presence of blocked processes. The persistence of any of these objects across a system crash or a system reboot is undefined. Conforming applications must not depend on any sort of persistence across a system reboot or a system crash.

- Models and Requirements

A realtime system requires synchronization and communication between the processes comprising the overall application. An efficient and reliable synchronization mechanism has to be provided in a realtime system that will allow more than one schedulable process mutually-exclusive access to the same resource. This synchronization mechanism has to allow for the optimal implementation of synchronization or systems implementors will define other, more cost-effective methods.

At issue are the methods whereby multiple processes (tasks) can be designed and implemented to work together in order to perform a single function. This requires interprocess communication and synchronization. A semaphore mechanism is the lowest level of synchronization that can be provided by an operating system.

A semaphore is defined as an object that has an integral value and a set of blocked processes associated with it. If the value is positive or zero, then the set of blocked processes is empty; otherwise, the size of the set is equal to the absolute value of the semaphore value. The value of the semaphore can be incremented or decremented by any process with access to the semaphore and must be done as an indivisible operation. When a semaphore value is less than or equal to zero, any process that attempts to lock it again will block or be informed that it is not possible to perform the operation.

A semaphore may be used to guard access to any resource accessible by more than one schedulable task in the system. It is a global entity and not associated with any particular process. As such, a method of obtaining access to the semaphore has to be provided by the operating system. A process that wants access to a critical resource (section) has to wait on the semaphore that guards that resource. When the semaphore is locked on behalf of a process, it knows that it can utilize the resource without interference by any other cooperating process in the system. When the process finishes its operation on the resource, leaving it in a well-defined state, it posts the semaphore, indicating that some other process may now obtain the resource associated with that semaphore.

In this section, mutexes and condition variables are specified as the synchronization mechanisms between threads.

These primitives are typically used for synchronizing threads that share memory in a single process. However, this section provides an option allowing the use of these synchronization interfaces and objects between processes that share memory, regardless of the method for sharing memory.

Much experience with semaphores shows that there are two distinct uses of synchronization: locking, which is typically of short duration; and waiting, which is typically of long or unbounded duration. These distinct usages map directly onto mutexes and condition variables, respectively.

Semaphores are provided in POSIX.1-2008 primarily to provide a means of synchronization for processes; these processes may or may not share memory. Mutexes and condition variables are specified as synchronization mechanisms between threads; these threads always share (some) memory. Both are synchronization paradigms that have been in widespread use for a number of years. Each set of primitives is particularly well matched to certain problems.

With respect to binary semaphores, experience has shown that condition variables and

mutexes are easier to use for many synchronization problems than binary semaphores. The primary reason for this is the explicit appearance of a Boolean predicate that specifies when the condition wait is satisfied. This Boolean predicate terminates a loop, including the call to `pthread_cond_wait()`. As a result, extra wakeups are benign since the predicate governs whether the thread will actually proceed past the condition wait. With stateful primitives, such as binary semaphores, the wakeup in itself typically means that the wait is satisfied. The burden of ensuring correctness for such waits is thus placed on *all* signalers of the semaphore rather than on an *explicitly coded* Boolean predicate located at the condition wait. Experience has shown that the latter creates a major improvement in safety and ease-of-use.

Counting semaphores are well matched to dealing with producer/consumer problems, including those that might exist between threads of different processes, or between a signal handler and a thread. In the former case, there may be little or no memory shared by the processes; in the latter case, one is not communicating between co-equal threads, but between a thread and an interrupt-like entity. It is for these reasons that POSIX.1-2008 allows semaphores to be used by threads.

Mutexes and condition variables have been effectively used with and without priority inheritance, priority ceiling, and other attributes to synchronize threads that share memory. The efficiency of their implementation is comparable to or better than that of other synchronization primitives that are sometimes harder to use (for example, binary semaphores). Furthermore, there is at least one known implementation of Ada tasking that uses these primitives. Mutexes and condition variables together constitute an appropriate, sufficient, and complete set of inter-thread synchronization primitives.

Efficient multi-threaded applications require high-performance synchronization primitives. Considerations of efficiency and generality require a small set of primitives upon which more sophisticated synchronization functions can be built.

• Standardization Issues

It is possible to implement very high-performance semaphores using test-and-set instructions on shared memory locations. The library routines that implement such a high-performance interface have to properly ensure that a `sem_wait()` or `sem_trywait()` operation that cannot be performed will issue a blocking semaphore system call or properly report the condition to the application. The same interface to the application program would be provided by a high-performance implementation.

B.2.8.1 Realtime Signals

**Realtime Signals Extension**

This portion of the rationale presents models, requirements, and standardization issues relevant to the Realtime Signals Extension. This extension provides the capability required to support reliable, deterministic, asynchronous notification of events. While a new mechanism, unencumbered by the historical usage and semantics of POSIX.1 signals, might allow for a more efficient implementation, the application requirements for event notification can be met with a small number of extensions to signals. Therefore, a minimal set of extensions to signals to support the application requirements is specified.

The realtime signal extensions specified in this section are used by other realtime functions requiring asynchronous notification:

• Models

The model supported is one of multiple cooperating processes, each of which handles

multiple asynchronous external events. Events represent occurrences that are generated as the result of some activity in the system. Examples of occurrences that can constitute an event include:

- Completion of an asynchronous I/O request
- Expiration of a POSIX.1b timer
- Arrival of an interprocess message
- Generation of a user-defined event

Processing of these events may occur synchronously via polling for event notifications or asynchronously via a software interrupt mechanism. Existing practice for this model is well established for traditional proprietary realtime operating systems, realtime executives, and realtime extended POSIX-like systems.

A contrasting model is that of “cooperating sequential processes” where each process handles a single priority of events via polling. Each process blocks while waiting for events, and each process depends on the preemptive, priority-based process scheduling mechanism to arbitrate between events of different priority that need to be processed concurrently. Existing practice for this model is also well established for small realtime executives that typically execute in an unprotected physical address space, but it is just emerging in the context of a fuller function operating system with multiple virtual address spaces.

It could be argued that the cooperating sequential process model, and the facilities supported by the POSIX Threads Extension obviate a software interrupt model. But, even with the cooperating sequential process model, the need has been recognized for a software interrupt model to handle exceptional conditions and process aborting, so the mechanism must be supported in any case. Furthermore, it is not the purview of POSIX.1-2008 to attempt to convince realtime practitioners that their current application models based on software interrupts are “broken” and should be replaced by the cooperating sequential process model. Rather, it is the charter of POSIX.1-2008 to provide standard extensions to mechanisms that support existing realtime practice.

#### • Requirements

This section discusses the following realtime application requirements for asynchronous event notification:

- Reliable delivery of asynchronous event notification

The events notification mechanism guarantees delivery of an event notification. Asynchronous operations (such as asynchronous I/O and timers) that complete significantly after they are invoked have to guarantee that delivery of the event notification can occur at the time of completion.

- Prioritized handling of asynchronous event notifications

The events notification mechanism supports the assigning of a user function as an event notification handler. Furthermore, the mechanism supports the preemption of an event handler function by a higher priority event notification and supports the selection of the highest priority pending event notification when multiple notifications (of different priority) are pending simultaneously.

The model here is based on hardware interrupts. Asynchronous event handling allows the application to ensure that time-critical events are immediately processed when delivered, without the indeterminism of being at a random location within a polling loop. Use of handler priority allows the specification of how handlers are

- 121722 interrupted by other higher priority handlers.
- 121723 — Differentiation between multiple occurrences of event notifications of the same type
- 121724 The events notification mechanism passes an application-defined value to the event  
121725 handler function. This value can be used for a variety of purposes, such as enabling  
121726 the application to identify which of several possible events of the same type (for  
121727 example, timer expirations) has occurred.
- 121728 — Polled reception of asynchronous event notifications
- 121729 The events notification mechanism supports blocking and non-blocking polls for  
121730 asynchronous event notification.
- 121731 The polled mode of operation is often preferred over the interrupt mode by those  
121732 practitioners accustomed to this model. Providing support for this model facilitates  
121733 the porting of applications based on this model to POSIX.1b conforming systems.
- 121734 — Deterministic response to asynchronous event notifications
- 121735 The events notification mechanism does not preclude implementations that provide  
121736 deterministic event dispatch latency and minimizes the number of system calls  
121737 needed to use the event facilities during realtime processing.
- 121738 • Rationale for Extension
- 121739 POSIX.1 signals have many of the characteristics necessary to support the asynchronous  
121740 handling of event notifications, and the Realtime Signals Extension addresses the  
121741 following deficiencies in the POSIX.1 signal mechanism:
- 121742 — Signals do not support reliable delivery of event notification. Subsequent  
121743 occurrences of a pending signal are not guaranteed to be delivered.
- 121744 — Signals do not support prioritized delivery of event notifications. The order of signal  
121745 delivery when multiple unblocked signals are pending is undefined.
- 121746 — Signals do not support the differentiation between multiple signals of the same type.

121747 *B.2.8.2 Asynchronous I/O*

121748 Many applications need to interact with the I/O subsystem in an asynchronous manner. The  
121749 asynchronous I/O mechanism provides the ability to overlap application processing and I/O  
121750 operations initiated by the application. The asynchronous I/O mechanism allows a single  
121751 process to perform I/O simultaneously to a single file multiple times or to multiple files  
121752 multiple times.

121753 **Overview**

121754 Asynchronous I/O operations proceed in logical parallel with the processing done by the  
121755 application after the asynchronous I/O has been initiated. Other than this difference,  
121756 asynchronous I/O behaves similarly to normal I/O using *read()*, *write()*, *lseek()*, and *fsync()*.  
121757 The effect of issuing an asynchronous I/O request is as if a separate thread of execution were to  
121758 perform atomically the implied *lseek()* operation, if any, and then the requested I/O operation  
121759 (either *read()*, *write()*, or *fsync()*). There is no seek implied with a call to *aio\_fsync()*. Concurrent  
121760 asynchronous operations and synchronous operations applied to the same file update the file as  
121761 if the I/O operations had proceeded serially.

121762 When asynchronous I/O completes, a signal can be delivered to the application to indicate the  
121763 completion of the I/O. This signal can be used to indicate that buffers and control blocks used  
121764 for asynchronous I/O can be reused. Signal delivery is not required for an asynchronous

operation and may be turned off on a per-operation basis by the application. Signals may also be synchronously polled using *aio\_suspend()*, *sigtimedwait()*, or *sigwaitinfo()*.

Normal I/O has a return value and an error status associated with it. Asynchronous I/O returns a value and an error status when the operation is first submitted, but that only relates to whether the operation was successfully queued up for servicing. The I/O operation itself also has a return status and an error value. To allow the application to retrieve the return status and the error value, functions are provided that, given the address of an asynchronous I/O control block, yield the return and error status associated with the operation. Until an asynchronous I/O operation is done, its error status is [EINPROGRESS]. Thus, an application can poll for completion of an asynchronous I/O operation by waiting for the error status to become equal to a value other than [EINPROGRESS]. The return status of an asynchronous I/O operation is undefined so long as the error status is equal to [EINPROGRESS].

Storage for asynchronous operation return and error status may be limited. Submission of asynchronous I/O operations may fail if this storage is exceeded. When an application retrieves the return status of a given asynchronous operation, therefore, any system-maintained storage used for this status and the error status may be reclaimed for use by other asynchronous operations.

Asynchronous I/O can be performed on file descriptors that have been enabled for POSIX.1b synchronized I/O. In this case, the I/O operation still occurs asynchronously, as defined herein; however, the asynchronous operation I/O in this case is not completed until the I/O has reached either the state of synchronized I/O data integrity completion or synchronized I/O file integrity completion, depending on the sort of synchronized I/O that is enabled on the file descriptor.

## Models

Three models illustrate the use of asynchronous I/O: a journalization model, a data acquisition model, and a model of the use of asynchronous I/O in supercomputing applications.

- Journalization Model

Many realtime applications perform low-priority journalizing functions. Journalizing requires that logging records be queued for output without blocking the initiating process.

- Data Acquisition Model

A data acquisition process may also serve as a model. The process has two or more channels delivering intermittent data that must be read within a certain time. The process issues one asynchronous read on each channel. When one of the channels needs data collection, the process reads the data and posts it through an asynchronous write to secondary memory for future processing.

- Supercomputing Model

The supercomputing community has used asynchronous I/O much like that specified in POSIX.1 for many years. This community requires the ability to perform multiple I/O operations to multiple devices with a minimal number of entries to “the system”; each entry to “the system” provokes a major delay in operations when compared to the normal progress made by the application. This existing practice motivated the use of combined *lseek()* and *read()* or *write()* calls, as well as the *lio\_listio()* call. Another common practice is to disable signal notification for I/O completion, and simply poll for I/O completion at some interval by which the I/O should be completed. Likewise, interfaces like *aio\_cancel()* have been in successful commercial use for many years. Note also that an underlying implementation of asynchronous I/O will require the ability, at least internally, to cancel outstanding asynchronous I/O, at least when the process exits. (Consider an asynchronous read from a terminal, when the process intends to exit immediately.)



## Requirements

Asynchronous input and output for realtime implementations have these requirements:

- The ability to queue multiple asynchronous read and write operations to a single open instance. Both sequential and random access should be supported.
- The ability to queue asynchronous read and write operations to multiple open instances.
- The ability to obtain completion status information by polling and/or asynchronous event notification.
- Asynchronous event notification on asynchronous I/O completion is optional.
- It has to be possible for the application to associate the event with the *aioctx* for the operation that generated the event.
- The ability to cancel queued requests.
- The ability to wait upon asynchronous I/O completion in conjunction with other types of events.
- The ability to accept an *aio\_read()* and an *aio\_cancel()* for a device that accepts a *read()*, and the ability to accept an *aio\_write()* and an *aio\_cancel()* for a device that accepts a *write()*. This does not imply that the operation is asynchronous.

## Standardization Issues

The following issues are addressed by the standardization of asynchronous I/O:

- Rationale for New Interface

Non-blocking I/O does not satisfy the needs of either realtime or high-performance computing models; these models require that a process overlap program execution and I/O processing. Realtime applications will often make use of direct I/O to or from the address space of the process, or require synchronized (unbuffered) I/O; they also require the ability to overlap this I/O with other computation. In addition, asynchronous I/O allows an application to keep a device busy at all times, possibly achieving greater throughput. Supercomputing and database architectures will often have specialized hardware that can provide true asynchrony underlying the logical asynchrony provided by this interface. In addition, asynchronous I/O should be supported by all types of files and devices in the same manner.

- Effect of Buffering

If asynchronous I/O is performed on a file that is buffered prior to being actually written to the device, it is possible that asynchronous I/O will offer no performance advantage over normal I/O; the cycles *stolen* to perform the asynchronous I/O will be taken away from the running process and the I/O will occur at interrupt time. This potential lack of gain in performance in no way obviates the need for asynchronous I/O by realtime applications, which very often will use specialized hardware support, multiple processors, and/or unbuffered, synchronized I/O.

### B.2.8.3 Memory Management

All memory management and shared memory definitions are located in the `<sys/mman.h>` header. This is for alignment with historical practice.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/7 is applied, correcting the shading and margin markers in the introduction to Section 2.8.3.1.

## Memory Locking Functions

This portion of the rationale presents models, requirements, and standardization issues relevant to process memory locking.

- Models

Realtime systems that conform to POSIX.1-2008 are expected (and desired) to be supported on systems with demand-paged virtual memory management, non-paged swapping memory management, and physical memory systems with no memory management hardware. The general case, however, is the demand-paged, virtual memory system with each POSIX process running in a virtual address space. Note that this includes architectures where each process resides in its own virtual address space and architectures where the address space of each process is only a portion of a larger global virtual address space.

The concept of memory locking is introduced to eliminate the indeterminacy introduced by paging and swapping, and to support an upper bound on the time required to access the memory mapped into the address space of a process. Ideally, this upper bound will be the same as the time required for the processor to access “main memory”, including any address translation and cache miss overheads. But some implementations—primarily on mainframes—will not actually force locked pages to be loaded and held resident in main memory. Rather, they will handle locked pages so that accesses to these pages will meet the performance metrics for locked process memory in the implementation. Also, although it is not, for example, the intention that this interface, as specified, be used to lock process memory into “cache”, it is conceivable that an implementation could support a large static RAM memory and define this as “main memory” and use a large[r] dynamic RAM as “backing store”. These interfaces could then be interpreted as supporting the locking of process memory into the static RAM. Support for multiple levels of backing store would require extensions to these interfaces.

Implementations may also use memory locking to guarantee a fixed translation between virtual and physical addresses where such is beneficial to improving determinacy for direct-to/from-process input/output. POSIX.1-2008 does not guarantee to the application that the virtual-to-physical address translations, if such exist, are fixed, because such behavior would not be implementable on all architectures on which implementations of POSIX.1-2008 are expected. But POSIX.1-2008 does mandate that an implementation define, for the benefit of potential users, whether or not locking guarantees fixed translations.

Memory locking is defined with respect to the address space of a process. Only the pages mapped into the address space of a process may be locked by the process, and when the pages are no longer mapped into the address space—for whatever reason—the locks established with respect to that address space are removed. Shared memory areas warrant special mention, as they may be mapped into more than one address space or mapped more than once into the address space of a process; locks may be established on pages within these areas with respect to several of these mappings. In such a case, the lock state of the underlying physical pages is the logical OR of the lock state with respect to each of the mappings. Only when all such locks have been removed are the shared pages considered unlocked.

In recognition of the page granularity of Memory Management Units (MMU), and in order to support locking of ranges of address space, memory locking is defined in terms of “page” granularity. That is, for the interfaces that support an address and size specification for the region to be locked, the address must be on a page boundary, and all pages mapped by the specified range are locked, if valid. This means that the length is implicitly rounded

up to a multiple of the page size. The page size is implementation-defined and is available to applications as a compile-time symbolic constant or at runtime via *sysconf()*.

A “real memory” POSIX.1b implementation that has no MMU could elect not to support these interfaces, returning [ENOSYS]. But an application could easily interpret this as meaning that the implementation would unconditionally page or swap the application when such is not the case. It is the intention of POSIX.1-2008 that such a system could define these interfaces as “NO-OPs”, returning success without actually performing any function except for mandated argument checking.

#### • Requirements

For realtime applications, memory locking is generally considered to be required as part of application initialization. This locking is performed after an application has been loaded (that is, *exec'd*) and the program remains locked for its entire lifetime. But to support applications that undergo major mode changes where, in one mode, locking is required, but in another it is not, the specified interfaces allow repeated locking and unlocking of memory within the lifetime of a process.

When a realtime application locks its address space, it should not be necessary for the application to then “touch” all of the pages in the address space to guarantee that they are resident or else suffer potential paging delays the first time the page is referenced. Thus, POSIX.1-2008 requires that the pages locked by the specified interfaces be resident when the locking functions return successfully.

Many architectures support system-managed stacks that grow automatically when the current extent of the stack is exceeded. A realtime application has a requirement to be able to “preallocate” sufficient stack space and lock it down so that it will not suffer page faults to grow the stack during critical realtime operation. There was no consensus on a portable way to specify how much stack space is needed, so POSIX.1-2008 supports no specific interface for preallocating stack space. But an application can portably lock down a specific amount of stack space by specifying MCL\_FUTURE in a call to *mlockall()* and then calling a dummy function that declares an automatic array of the desired size.

Memory locking for realtime applications is also generally considered to be an “all or nothing” proposition. That is, the entire process, or none, is locked down. But, for applications that have well-defined sections that need to be locked and others that do not, POSIX.1-2008 supports an optional set of interfaces to lock or unlock a range of process addresses. Reasons for locking down a specific range include:

- An asynchronous event handler function that must respond to external events in a deterministic manner such that page faults cannot be tolerated
- An input/output “buffer” area that is the target for direct-to-process I/O, and the overhead of implicit locking and unlocking for each I/O call cannot be tolerated

Finally, locking is generally viewed as an “application-wide” function. That is, the application is globally aware of which regions are locked and which are not over time. This is in contrast to a function that is used temporarily within a “third party” library routine whose function is unknown to the application, and therefore must have no “side-effects”. The specified interfaces, therefore, do not support “lock stacking” or “lock nesting” within a process. But, for pages that are shared between processes or mapped more than once into a process address space, “lock stacking” is essentially mandated by the requirement that unlocking of pages that are mapped by more than one process or more than once by the same process does not affect locks established on the other mappings.

There was some support for “lock stacking” so that locking could be transparently used in functions or opaque modules. But the consensus was not to burden all implementations



121951 with lock stacking (and reference counting), and an implementation option was proposed.  
 121952 There were strong objections to the option because applications would have to support  
 121953 both options in order to remain portable. The consensus was to eliminate lock stacking  
 121954 altogether, primarily through overwhelming support for the System V “m[un]lock[all]”  
 121955 interface on which POSIX.1-2008 is now based.

121956 Locks are not inherited across *fork()*s because some implementations implement *fork()* by  
 121957 creating new address spaces for the child. In such an implementation, requiring locks to be  
 121958 inherited would lead to new situations in which a fork would fail due to the inability of  
 121959 the system to lock sufficient memory to lock both the parent and the child. The consensus  
 121960 was that there was no benefit to such inheritance. Note that this does not mean that locks  
 121961 are removed when, for instance, a thread is created in the same address space.

121962 Similarly, locks are not inherited across *exec* because some implementations implement *exec*  
 121963 by unmapping all of the pages in the address space (which, by definition, removes the  
 121964 locks on these pages), and maps in pages of the *exec*'d image. In such an implementation,  
 121965 requiring locks to be inherited would lead to new situations in which *exec* would fail.  
 121966 Reporting this failure would be very cumbersome to detect in time to report to the calling  
 121967 process, and no appropriate mechanism exists for informing the *exec*'d process of its status.

121968 It was determined that, if the newly loaded application required locking, it was the  
 121969 responsibility of that application to establish the locks. This is also in keeping with the  
 121970 general view that it is the responsibility of the application to be aware of all locks that are  
 121971 established.

121972 There was one request to allow (not mandate) locks to be inherited across *fork()*, and a  
 121973 request for a flag, *MCL\_INHERIT*, that would specify inheritance of memory locks across  
 121974 *execs*. Given the difficulties raised by this and the general lack of support for the feature in  
 121975 POSIX.1-2008, it was not added. POSIX.1-2008 does not preclude an implementation from  
 121976 providing this feature for administrative purposes, such as a “run” command that will  
 121977 lock down and execute a specified application. Additionally, the rationale for the objection  
 121978 equated *fork()* with creating a thread in the address space. POSIX.1-2008 does not mandate  
 121979 releasing locks when creating additional threads in an existing process.

#### 121980 • Standardization Issues

121981 One goal of POSIX.1-2008 is to define a set of primitives that provide the necessary  
 121982 functionality for realtime applications, with consideration for the needs of other  
 121983 application domains where such were identified, which is based to the extent possible on  
 121984 existing industry practice.

121985 The Memory Locking option is required by many realtime applications to tune  
 121986 performance. Such a facility is accomplished by placing constraints on the virtual memory  
 121987 system to limit paging of time of the process or of critical sections of the process. This  
 121988 facility should not be used by most non-realtime applications.

121989 Optional features provided in POSIX.1-2008 allow applications to lock selected address  
 121990 ranges with the caveat that the process is responsible for being aware of the page  
 121991 granularity of locking and the unnested nature of the locks.

## Mapped Files Functions

The memory mapped files functionality provides a mechanism that allows a process to access files by directly incorporating file data into its address space. Once a file is “mapped” into a process address space, the data can be manipulated by instructions as memory. The use of mapped files can significantly reduce I/O data movement since file data does not have to be copied into process data buffers as in *read()* and *write()*. If more than one process maps a file, its contents are shared among them. This provides a low overhead mechanism by which processes can synchronize and communicate.

- Historical Perspective

Realtime applications have historically been implemented using a collection of cooperating processes or tasks. In early systems, these processes ran on bare hardware (that is, without an operating system) with no memory relocation or protection. The application paradigms that arose from this environment involve the sharing of data between the processes.

When realtime systems were implemented on top of vendor-supplied operating systems, the paradigm or performance benefits of direct access to data by multiple processes was still deemed necessary. As a result, operating systems that claim to support realtime applications must support the shared memory paradigm.

Additionally, a number of realtime systems provide the ability to map specific sections of the physical address space into the address space of a process. This ability is required if an application is to obtain direct access to memory locations that have specific properties (for example, refresh buffers or display devices, dual ported memory locations, DMA target locations). The use of this ability is common enough to warrant some degree of standardization of its interface. This ability overlaps the general paradigm of shared memory in that, in both instances, common global objects are made addressable by individual processes or tasks.

Finally, a number of systems also provide the ability to map process addresses to files. This provides both a general means of sharing persistent objects, and using files in a manner that optimizes memory and swapping space usage.

Simple shared memory is clearly a special case of the more general file mapping capability. In addition, there is relatively widespread agreement and implementation of the file mapping interface. In these systems, many different types of objects can be mapped (for example, files, memory, devices, and so on) using the same mapping interfaces. This approach both minimizes interface proliferation and maximizes the generality of programs using the mapping interfaces.

- Memory Mapped Files Usage

A memory object can be concurrently mapped into the address space of one or more processes. The *mmap()* and *munmap()* functions allow a process to manipulate their address space by mapping portions of memory objects into it and removing them from it. When multiple processes map the same memory object, they can share access to the underlying data. Implementations may restrict the size and alignment of mappings to be on *page*-size boundaries. The page size, in bytes, is the value of the system-configurable variable {PAGESIZE}, typically accessed by calling *sysconf()* with a *name* argument of *\_SC\_PAGESIZE*. If an implementation has no restrictions on size or alignment, it may specify a 1-byte page size.

To map memory, a process first opens a memory object. The *ftruncate()* function can be used to contract or extend the size of the memory object even when the object is currently mapped. If the memory object is extended, the contents of the extended areas are zeros.

122039 After opening a memory object, the application maps the object into its address space  
122040 using the *mmap()* function call. Once a mapping has been established, it remains mapped  
122041 until unmapped with *munmap()*, even if the memory object is closed. The *mprotect()*  
122042 function can be used to change the memory protections initially established by *mmap()*.

122043 A *close()* of the file descriptor, while invalidating the file descriptor itself, does not unmap  
122044 any mappings established for the memory object. The address space, including all mapped  
122045 regions, is inherited on *fork()*. The entire address space is unmapped on process  
122046 termination or by successful calls to any of the *exec* family of functions.

122047 The *msync()* function is used to force mapped file data to permanent storage.

122048 • Effects on Other Functions

122049 With memory mapped files, the operation of the *open()*, *creat()*, and *unlink()* functions are  
122050 a natural result of using the file system name space to map the global names for memory  
122051 objects.

122052 The *ftruncate()* function can be used to set the length of a sharable memory object.

122053 The meaning of *stat()* fields other than the size and protection information is undefined on  
122054 implementations where memory objects are not implemented using regular files. When  
122055 regular files are used, the times reflect when the implementation updated the file image of  
122056 the data, not when a process updated the data in memory.

122057 The operations of *fdopen()*, *write()*, *read()*, and *lseek()* were made unspecified for objects  
122058 opened with *shm\_open()*, so that implementations that did not implement memory objects  
122059 as regular files would not have to support the operation of these functions on shared  
122060 memory objects.

122061 The behavior of memory objects with respect to *close()*, *dup()*, *dup2()*, *open()*, *close()*,  
122062 *fork()*, *\_exit()*, and the *exec* family of functions is the same as the behavior of the existing  
122063 practice of the *mmap()* function.

122064 A memory object can still be referenced after a close. That is, any mappings made to the  
122065 file are still in effect, and reads and writes that are made to those mappings are still valid  
122066 and are shared with other processes that have the same mapping. Likewise, the memory  
122067 object can still be used if any references remain after its name(s) have been deleted. Any  
122068 references that remain after a close must not appear to the application as file descriptors.

122069 This is existing practice for *mmap()* and *close()*. In addition, there are already mappings  
122070 present (text, data, stack) that do not have open file descriptors. The text mapping in  
122071 particular is considered a reference to the file containing the text. The desire was to treat all  
122072 mappings by the process uniformly. Also, many modern implementations use *mmap()* to  
122073 implement shared libraries, and it would not be desirable to keep file descriptors for each  
122074 of the many libraries an application can use. It was felt there were many other existing  
122075 programs that used this behavior to free a file descriptor, and thus POSIX.1-2008 could not  
122076 forbid it and still claim to be using existing practice.

122077 For implementations that implement memory objects using memory only, memory objects  
122078 will retain the memory allocated to the file after the last close and will use that same  
122079 memory on the next open. Note that closing the memory object is not the same as deleting  
122080 the name, since the memory object is still defined in the memory object name space.

122081 The locks of *fcntl()* do not block any read or write operation, including read or write access  
122082 to shared memory or mapped files. In addition, implementations that only support shared  
122083 memory objects should not be required to implement record locks. The reference to *fcntl()*  
122084 is added to make this point explicitly. The other *fcntl()* commands are useful with shared  
122085 memory objects.

122086 The size of pages that mapping hardware may be able to support may be a configurable  
122087 value, or it may change based on hardware implementations. The addition of the  
122088 `_SC_PAGESIZE` parameter to the `sysconf()` function is provided for determining the  
122089 mapping page size at runtime.

## 122090 Shared Memory Functions

122091 Implementations may support the Shared Memory Objects option independently of memory  
122092 mapped files. Shared memory objects are named regions of storage that may be independent of  
122093 the file system and can be mapped into the address space of one or more processes to allow  
122094 them to share the associated memory.

### 122095 • Requirements

122096 Shared memory is used to share data among several processes, each potentially running at  
122097 different priority levels, responding to different inputs, or performing separate tasks.  
122098 Shared memory is not just simply providing common access to data, it is providing the  
122099 fastest possible communication between the processes. With one memory write operation,  
122100 a process can pass information to as many processes as have the memory region mapped.

122101 As a result, shared memory provides a mechanism that can be used for all other  
122102 interprocess communication facilities. It may also be used by an application for  
122103 implementing more sophisticated mechanisms than semaphores and message queues.

122104 The need for a shared memory interface is obvious for virtual memory systems, where the  
122105 operating system is directly preventing processes from accessing each other's data.  
122106 However, in unprotected systems, such as those found in some embedded controllers, a  
122107 shared memory interface is needed to provide a portable mechanism to allocate a region of  
122108 memory to be shared and then to communicate the address of that region to other  
122109 processes.

122110 This, then, provides the minimum functionality that a shared memory interface must have  
122111 in order to support realtime applications: to allocate and name an object to be mapped into  
122112 memory for potential sharing (`open()` or `shm_open()`), and to make the memory object  
122113 available within the address space of a process (`mmap()`). To complete the interface, a  
122114 mechanism to release the claim of a process on a shared memory object (`munmap()`) is also  
122115 needed, as well as a mechanism for deleting the name of a sharable object that was  
122116 previously created (`unlink()` or `shm_unlink()`).

122117 After a mapping has been established, an implementation should not have to provide  
122118 services to maintain that mapping. All memory writes into that area will appear  
122119 immediately in the memory mapping of that region by any other processes.

122120 Thus, requirements include:

- 122121 — Support creation of sharable memory objects and the mapping of these objects into  
122122 the address space of a process.
- 122123 — Sharable memory objects should be accessed by global names accessible from all  
122124 processes.
- 122125 — Support the mapping of specific sections of physical address space (such as a  
122126 memory mapped device) into the address space of a process. This should not be  
122127 done by the process specifying the actual address, but again by an implementation-  
122128 defined global name (such as a special device name) dedicated to this purpose.
- 122129 — Support the mapping of discrete portions of these memory objects.

- 122130 — Support for minimum hardware configurations that contain no physical media on
- 122131 which to store shared memory contents permanently.
- 122132 — The ability to preallocate the entire shared memory region so that minimum
- 122133 hardware configurations without virtual memory support can guarantee contiguous
- 122134 space.
- 122135 — The maximizing of performance by not requiring functionality that would require
- 122136 implementation interaction above creating the shared memory area and returning
- 122137 the mapping.

122138 Note that the above requirements do not preclude:

- 122139 — The sharable memory object from being implemented using actual files on an actual
- 122140 file system.
- 122141 — The global name that is accessible from all processes being restricted to a file system
- 122142 area that is dedicated to handling shared memory.
- 122143 — An implementation not providing implementation-defined global names for the
- 122144 purpose of physical address mapping.

122145 • Shared Memory Objects Usage

122146 If the Shared Memory Objects option is supported, a shared memory object may be

122147 created, or opened if it already exists, with the *shm\_open()* function. If the shared memory

122148 object is created, it has a length of zero. The *ftruncate()* function can be used to set the size

122149 of the shared memory object after creation. The *shm\_unlink()* function removes the name

122150 for a shared memory object created by *shm\_open()*.

122151 • Shared Memory Overview

122152 The shared memory facility defined by POSIX.1-2008 usually results in memory locations

122153 being added to the address space of the process. The implementation returns the address

122154 of the new space to the application by means of a pointer. This works well in languages

122155 like C. However, in languages without pointer types it will not work. In the bindings for

122156 such a language, either a special COMMON section will need to be defined (which is

122157 unlikely), or the binding will have to allow existing structures to be mapped. The

122158 implementation will likely have to place restrictions on the size and alignment of such

122159 structures or will have to map a suitable region of the address space of the process into the

122160 memory object, and thus into other processes. These are issues for that particular language

122161 binding. For POSIX.1-2008, however, the practice will not be forbidden, merely undefined.

122162 Two potentially different name spaces are used for naming objects that may be mapped

122163 into process address spaces. When using memory mapped files, files may be accessed via

122164 *open()*. When the Shared Memory Objects option is supported, sharable memory objects

122165 that might not be files may be accessed via the *shm\_open()* function. These operations are

122166 not mutually-exclusive.

122167 Some implementations supporting the Shared Memory Objects option may choose to

122168 implement the shared memory object name space as part of the file system name space.

122169 There are several reasons for this:

- 122170 — It allows applications to prevent name conflicts by use of the directory structure.
- 122171 — It uses an existing mechanism for accessing global objects and prevents the creation
- 122172 of a new mechanism for naming global objects.

122173 In such implementations, memory objects can be implemented using regular files, if that is

122174 what the implementation chooses. The *shm\_open()* function can be implemented as an



122175 *open()* call in a fixed directory with the `O_CLOEXEC` flag set. The *shm\_unlink()* function  
122176 can be implemented as an *unlink()* call.

122177 On the other hand, it is also expected that small embedded systems that support the  
122178 Shared Memory Objects option may wish to implement shared memory without having  
122179 any file systems present. In this case, the implementations may choose to use a simple  
122180 string valued name space for shared memory regions. The *shm\_open()* function permits  
122181 either type of implementation.

122182 Some implementations have hardware that supports protection of mapped data from  
122183 certain classes of access and some do not. Systems that supply this functionality support  
122184 the memory protection functionality.

122185 Some implementations restrict size, alignment, and protections to be on *page*-size  
122186 boundaries. If an implementation has no restrictions on size or alignment, it may specify a  
122187 1-byte page size. Applications on implementations that do support larger pages must be  
122188 cognizant of the page size since this is the alignment and protection boundary.

122189 Simple embedded implementations may have a 1-byte page size and only support the  
122190 Shared Memory Objects option. This provides simple shared memory between processes  
122191 without requiring mapping hardware.

122192 POSIX.1-2008 specifically allows a memory object to remain referenced after a close  
122193 because that is existing practice for the *mmap()* function.

## 122194 **Typed Memory Functions**

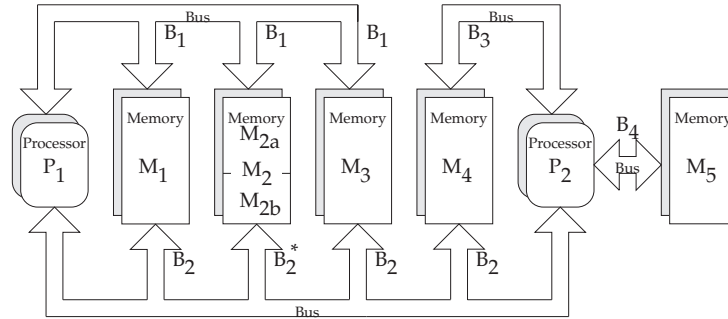
122195 Implementations may support the Typed Memory Objects option without supporting either the  
122196 Shared Memory option or memory mapped files. Types memory objects are pools of specialized  
122197 storage, different from the main memory resource normally used by a processor to hold code  
122198 and data, that can be mapped into the address space of one or more processes.

### 122199 • Model

122200 Realtime systems conforming to one of the POSIX.13 realtime profiles are expected (and  
122201 desired) to be supported on systems with more than one type or pool of memory (for  
122202 example, SRAM, DRAM, ROM, EPROM, EEPROM), where each type or pool of memory  
122203 may be accessible by one or more processors via one or more buses (ports). Memory  
122204 mapped files, shared memory objects, and the language-specific storage allocation  
122205 operators (*malloc()* for the ISO C standard, *new* for ISO Ada) fail to provide application  
122206 program interfaces versatile enough to allow applications to control their utilization of  
122207 such diverse memory resources. The typed memory interfaces *posix\_typed\_mem\_open()*,  
122208 *posix\_mem\_offset()*, *posix\_typed\_mem\_get\_info()*, *mmap()*, and *munmap()* defined herein  
122209 support the model of typed memory described below.

122210 For purposes of this model, a system comprises several processors (for example,  $P_1$  and  
122211  $P_2$ ), several physical memory pools (for example,  $M_1$ ,  $M_2$ ,  $M_{2a}$ ,  $M_{2b}$ ,  $M_3$ ,  $M_4$ , and  $M_5$ ), and  
122212 several buses or “ports” (for example,  $B_1$ ,  $B_2$ ,  $B_3$ , and  $B_4$ ) interconnecting the various  
122213 processors and memory pools in some system-specific way. Notice that some memory  
122214 pools may be contained in others (for example,  $M_{2a}$  and  $M_{2b}$  are contained in  $M_2$ ).

122215 **Figure B-1** (on page 3595) shows an example of such a model. In a system like this, an  
122216 application should be able to perform the following operations:



\* All addresses in pool M<sub>2</sub> (comprising pools M<sub>2a</sub> and M<sub>2b</sub>) accessible via port B<sub>1</sub>.  
 Addresses in pool M<sub>2b</sub> are also accessible via port B<sub>2</sub>.  
 Addresses in pool M<sub>2a</sub> are *not* accessible via port B<sub>2</sub>.

**Figure B-1** Example of a System with Typed Memory

#### — Typed Memory Allocation

An application should be able to allocate memory dynamically from the desired pool using the desired bus, and map it into the address space of a process. For example, processor P<sub>1</sub> can allocate some portion of memory pool M<sub>1</sub> through port B<sub>1</sub>, treating all unmapped subareas of M<sub>1</sub> as a heap-storage resource from which memory may be allocated. This portion of memory is mapped into address space of the process, and subsequently deallocated when unmapped from all processes.

#### — Using the Same Storage Region from Different Buses

An application process with a mapped region of storage that is accessed from one bus should be able to map that same storage area at another address (subject to page size restrictions detailed in *mmap()*), to allow it to be accessed from another bus. For example, processor P<sub>1</sub> may wish to access the same region of memory pool M<sub>2b</sub> both through ports B<sub>1</sub> and B<sub>2</sub>.

#### — Sharing Typed Memory Regions

Several application processes running on the same or different processors may wish to share a particular region of a typed memory pool. Each process or processor may wish to access this region through different buses. For example, processor P<sub>1</sub> may want to share a region of memory pool M<sub>4</sub> with processor P<sub>2</sub>, and they may be required to use buses B<sub>2</sub> and B<sub>3</sub>, respectively, to minimize bus contention. A problem arises here when a process allocates and maps a portion of fragmented memory and then wants to share this region of memory with another process, either in the same processor or different processors. The solution adopted is to allow the first process to find out the memory map (offsets and lengths) of all the different fragments of memory that were mapped into its address space, by repeatedly calling *posix\_mem\_offset()*. Then, this process can pass the offsets and lengths obtained to the second process, which can then map the same memory fragments into its address space.

#### — Contiguous Allocation

The problem of finding the memory map of the different fragments of the memory pool that were mapped into logically contiguous addresses of a given process can be

solved by requesting contiguous allocation. For example, a process in  $P_1$  can allocate 10 Kbytes of physically contiguous memory from  $M_3-B_1$ , and obtain the offset (within pool  $M_3$ ) of this block of memory. Then, it can pass this offset (and the length) to a process in  $P_2$  using some interprocess communication mechanism. The second process can map the same block of memory by using the offset transferred and specifying  $M_3-B_2$ .

#### 122254 — Unallocated Mapping

Any subarea of a memory pool that is mapped to a process, either as the result of an allocation request or an explicit mapping, is normally unavailable for allocation. Special processes such as debuggers, however, may need to map large areas of a typed memory pool, yet leave those areas available for allocation.

Typed memory allocation and mapping has to coexist with storage allocation operators like *malloc()*, but systems are free to choose how to implement this coexistence. For example, it may be system configuration-dependent if all available system memory is made part of one of the typed memory pools or if some part will be restricted to conventional allocation operators. Equally system configuration-dependent may be the availability of operators like *malloc()* to allocate storage from certain typed memory pools. It is not excluded to configure a system such that a given named pool,  $P_1$ , is in turn split into non-overlapping named subpools. For example,  $M_1-B_1$ ,  $M_2-B_1$ , and  $M_3-B_1$  could also be accessed as one common pool  $M_{123}-B_1$ . A call to *malloc()* on  $P_1$  could work on such a larger pool while full optimization of memory usage by  $P_1$  would require typed memory allocation at the subpool level.

#### 122270 • Existing Practice

OS-9 provides for the naming (numbering) and prioritization of memory types by a system administrator. It then provides APIs to request memory allocation of typed (colored) memory by number, and to generate a bus address from a mapped memory address (translate). When requesting colored memory, the user can specify type 0 to signify allocation from the first available type in priority order.

HP-RT presents interfaces to map different kinds of storage regions that are visible through a VME bus, although it does not provide allocation operations. It also provides functions to perform address translation between VME addresses and virtual addresses. It represents a VME-bus unique solution to the general problem.

The PSOS approach is similar (that is, based on a pre-established mapping of bus address ranges to specific memories) with a concept of segments and regions (regions dynamically allocated from a heap which is a special segment). Therefore, PSOS does not fully address the general allocation problem either. PSOS does not have a “process”-based model, but more of a “thread”-only-based model of multi-tasking. So mapping to a process address space is not an issue.

QNX uses the System V approach of opening specially named devices (shared memory segments) and using *mmap()* to then gain access from the process. They do not address allocation directly, but once typed shared memory can be mapped, an “allocation manager” process could be written to handle requests for allocation.

The System V approach also included allocation, implemented by opening yet other special “devices” which allocate, rather than appearing as a whole memory object.

The Orkid realtime kernel interface definition has operations to manage memory “regions” and “pools”, which are areas of memory that may reflect the differing physical nature of the memory. Operations to allocate memory from these regions and pools are also provided.



- Requirements

Existing practice in SVID-derived UNIX systems relies on functionality similar to *mmap()* and its related interfaces to achieve mapping and allocation of typed memory. However, the issue of sharing typed memory (allocated or mapped) and the complication of multiple ports are not addressed in any consistent way by existing UNIX system practice. Part of this functionality is existing practice in specialized realtime operating systems. In order to solidify the capabilities implied by the model above, the following requirements are imposed on the interface:

- Identification of Typed Memory Pools and Ports

All processes (running in all processors) in the system are able to identify a particular (system configured) typed memory pool accessed through a particular (system configured) port by a name. That name is a member of a name space common to all these processes, but need not be the same name space as that containing ordinary pathnames. The association between memory pools/ports and corresponding names is typically established when the system is configured. The “open” operation for typed memory objects should be distinct from the *open()* function, for consistency with other similar services, but implementable on top of *open()*. This implies that the handle for a typed memory object will be a file descriptor.

- Allocation and Mapping of Typed Memory

Once a typed memory object has been identified by a process, it is possible to both map user-selected subareas of that object into process address space and to map system-selected (that is, dynamically allocated) subareas of that object, with user-specified length, into process address space. It is also possible to determine the maximum length of memory allocation that may be requested from a given typed memory object.

- Sharing Typed Memory

Two or more processes are able to share portions of typed memory, either user-selected or dynamically allocated. This requirement applies also to dynamically allocated regions of memory that are composed of several non-contiguous pieces.

- Contiguous Allocation

For dynamic allocation, it is the user’s option whether the system is required to allocate a contiguous subarea within the typed memory object, or whether it is permitted to allocate discontinuous fragments which appear contiguous in the process mapping. Contiguous allocation simplifies the process of sharing allocated typed memory, while discontinuous allocation allows for potentially better recovery of deallocated typed memory.

- Accessing Typed Memory Through Different Ports

Once a subarea of a typed memory object has been mapped, it is possible to determine the location and length corresponding to a user-selected portion of that object within the memory pool. This location and length can then be used to remap that portion of memory for access from another port. If the referenced portion of typed memory was allocated discontinuously, the length thus determined may be shorter than anticipated, and the user code must adapt to the value returned.

- Deallocation

When a previously mapped subarea of typed memory is no longer mapped by any process in the system—as a result of a call or calls to *munmap()*—that subarea

becomes potentially reusable for dynamic allocation; actual reuse of the subarea is a function of the dynamic typed memory allocation policy.

— Unallocated Mapping

It must be possible to map user-selected subareas of a typed memory object without marking that subarea as unavailable for allocation. This option is not the default behavior, and requires appropriate privileges.

• Scenario

The following scenario will serve to clarify the use of the typed memory interfaces.

Process A running on  $P_1$  (see Figure B-1, on page 3595) wants to allocate some memory from memory pool  $M_2$ , and it wants to share this portion of memory with process B running on  $P_2$ . Since  $P_2$  only has access to the lower part of  $M_2$ , both processes will use the memory pool named  $M_{2b}$  which is the part of  $M_2$  that is accessible both from  $P_1$  and  $P_2$ . The operations that both processes need to perform are shown below:

— Allocating Typed Memory

Process A calls *posix\_typed\_mem\_open()* with the name **/typed.m2b-b1** and a *tflag* of **POSIX\_TYPED\_MEM\_ALLOCATE** to get a file descriptor usable for allocating from pool  $M_{2b}$  accessed through port  $B_1$ . It then calls *mmap()* with this file descriptor requesting a length of 4096 bytes. The system allocates two discontinuous blocks of sizes 1024 and 3072 bytes within  $M_{2b}$ . The *mmap()* function returns a pointer to a 4096-byte array in process A's logical address space, mapping the allocated blocks contiguously. Process A can then utilize the array, and store data in it.

— Determining the Location of the Allocated Blocks

Process A can determine the lengths and offsets (relative to  $M_{2b}$ ) of the two blocks allocated, by using the following procedure: First, process A calls *posix\_mem\_offset()* with the address of the first element of the array and length 4096. Upon return, the offset and length (1024 bytes) of the first block are returned. A second call to *posix\_mem\_offset()* is then made using the address of the first element of the array plus 1024 (the length of the first block), and a new length of 4096–1024. If there were more fragments allocated, this procedure could have been continued within a loop until the offsets and lengths of all the blocks were obtained. Notice that this relatively complex procedure can be avoided if contiguous allocation is requested (by opening the typed memory object with the *tflag* **POSIX\_TYPED\_MEM\_ALLOCATE\_CONTIG**).

— Sharing Data Across Processes

Process A passes the two offset values and lengths obtained from the *posix\_mem\_offset()* calls to process B running on  $P_2$ , via some form of interprocess communication. Process B can gain access to process A's data by calling *posix\_typed\_mem\_open()* with the name **/typed.m2b-b2** and a *tflag* of zero, then using two *mmap()* calls on the resulting file descriptor to map the two subareas of that typed memory object to its own address space.

• Rationale for no *mem\_alloc()* and *mem\_free()*

The standard developers had originally proposed a pair of new flags to *mmap()* which, when applied to a typed memory object descriptor, would cause *mmap()* to allocate dynamically from an unallocated and unmapped area of the typed memory object. Deallocation was similarly accomplished through the use of *munmap()*. This was rejected by the ballot group because it excessively complicated the (already rather complex)

*mmap()* interface and introduced semantics useful only for typed memory, to a function which must also map shared memory and files. They felt that a memory allocator should be built on top of *mmap()* instead of being incorporated within the same interface, much as the ISO C standard libraries build *malloc()* on top of the virtual memory mapping functions *brk()* and *sbrk()*. This would eliminate the complicated semantics involved with unmapping only part of an allocated block of typed memory.

To attempt to achieve ballot group consensus, typed memory allocation and deallocation was first migrated from *mmap()* and *munmap()* to a pair of complementary functions modeled on the ISO C standard *malloc()* and *free()*. The *mem\_alloc()* function specified explicitly the typed memory object (typed memory pool/access port) from which allocation takes place, unlike *malloc()* where the memory pool and port are unspecified. The *mem\_free()* function handled deallocation. These new semantics still met all of the requirements detailed above without modifying the behavior of *mmap()* except to allow it to map specified areas of typed memory objects. An implementation would have been free to implement *mem\_alloc()* and *mem\_free()* over *mmap()*, through *mmap()*, or independently but cooperating with *mmap()*.

The ballot group was queried to see if this was an acceptable alternative, and while there was some agreement that it achieved the goal of removing the complicated semantics of allocation from the *mmap()* interface, several balloters realized that it just created two additional functions that behaved, in great part, like *mmap()*. These balloters proposed an alternative which has been implemented here in place of a separate *mem\_alloc()* and *mem\_free()*. This alternative is based on four specific suggestions:

1. The *posix\_typed\_mem\_open()* function should provide a flag which specifies “allocate on *mmap()*” (otherwise, *mmap()* just maps the underlying object). This allows things roughly similar to */dev/zero* versus */dev/swap*. Two such flags have been implemented, one of which forces contiguous allocation.
2. The *posix\_mem\_offset()* function is acceptable because it can be applied usefully to mapped objects in general. It should return the file descriptor of the underlying object.
3. The *mem\_get\_info()* function in an earlier draft should be renamed *posix\_typed\_mem\_get\_info()* because it is not generally applicable to memory objects. It should probably return the file descriptor’s allocation attribute. The renaming of the function has been implemented, but having it return a piece of information which is readily known by an application without this function has been rejected. Its whole purpose is to query the typed memory object for attributes that are not user-specified, but determined by the implementation.
4. There should be no separate *mem\_alloc()* or *mem\_free()* functions. Instead, using *mmap()* on a typed memory object opened with an “allocate on *mmap()*” flag should be used to force allocation. These are precisely the semantics defined in the current draft.

- Rationale for no Typed Memory Access Management

The working group had originally defined an additional interface (and an additional kind of object: typed memory master) to establish and dissolve mappings to typed memory on behalf of devices or processors which were independent of the operating system and had no inherent capability to directly establish mappings on their own. This was to have provided functionality similar to device driver interfaces such as *physio()* and their underlying bus-specific interfaces (for example, *mballoc()*) which serve to set up and break down DMA pathways, and derive mapped addresses for use by hardware devices and processor cards.

122437 The ballot group felt that this was beyond the scope of POSIX.1 and its amendments.  
122438 Furthermore, the removal of interrupt handling interfaces from a preceding amendment  
122439 (the IEEE Std 1003.1d-1999) during its balloting process renders these typed memory  
122440 access management interfaces an incomplete solution to portable device management from  
122441 a user process; it would be possible to initiate a device transfer to/from typed memory, but  
122442 impossible to handle the transfer-complete interrupt in a portable way.

122443 To achieve ballot group consensus, all references to typed memory access management  
122444 capabilities were removed. The concept of portable interfaces from a device driver to both  
122445 operating system and hardware is being addressed by the Uniform Driver Interface (UDI)  
122446 industry forum, with formal standardization deferred until proof of concept and industry-  
122447 wide acceptance and implementation.

#### 122448 B.2.8.4 Process Scheduling

122449 IEEE PASC Interpretation 1003.1 #96 has been applied, adding the *pthread\_setschedprio()*  
122450 function. This was added since previously there was no way for a thread to lower its own  
122451 priority without going to the tail of the threads list for its new priority. This capability is  
122452 necessary to bound the duration of priority inversion encountered by a thread.

122453 The following portion of the rationale presents models, requirements, and standardization  
122454 issues relevant to process scheduling; see also [Section B.2.9.4](#) (on page 3640).

122455 In an operating system supporting multiple concurrent processes, the system determines the  
122456 order in which processes execute to meet implementation-defined goals. For time-sharing  
122457 systems, the goal is to enhance system throughput and promote fairness; the application is  
122458 provided with little or no control over this sequencing function. While this is acceptable and  
122459 desirable behavior in a time-sharing system, it is inappropriate in a realtime system; realtime  
122460 applications must specifically control the execution sequence of their concurrent processes in  
122461 order to meet externally defined response requirements.

122462 In POSIX.1-2008, the control over process sequencing is provided using a concept of scheduling  
122463 policies. These policies, described in detail in this section, define the behavior of the system  
122464 whenever processor resources are to be allocated to competing processes. Only the behavior of  
122465 the policy is defined; conforming implementations are free to use any mechanism desired to  
122466 achieve the described behavior.

##### 122467 • Models

122468 In an operating system supporting multiple concurrent processes, the system determines  
122469 the order in which processes execute and might force long-running processes to yield to  
122470 other processes at certain intervals. Typically, the scheduling code is executed whenever an  
122471 event occurs that might alter the process to be executed next.

122472 The simplest scheduling strategy is a “first-in, first-out” (FIFO) dispatcher. Whenever a  
122473 process becomes runnable, it is placed on the end of a ready list. The process at the front of  
122474 the ready list is executed until it exits or becomes blocked, at which point it is removed  
122475 from the list. This scheduling technique is also known as “run-to-completion” or “run-to-  
122476 block”.

122477 A natural extension to this scheduling technique is the assignment of a “non-migrating  
122478 priority” to each process. This policy differs from strict FIFO scheduling in only one  
122479 respect: whenever a process becomes runnable, it is placed at the end of the list of  
122480 processes runnable at that priority level. When selecting a process to run, the system  
122481 always selects the first process from the highest priority queue with a runnable process.  
122482 Thus, when a process becomes unblocked, it will preempt a running process of lower  
122483 priority without otherwise altering the ready list. Further, if a process elects to alter its

122484 priority, it is removed from the ready list and reinserted, using its new priority, according  
122485 to the policy above.

122486 While the above policy might be considered unfriendly in a time-sharing environment in  
122487 which multiple users require more balanced resource allocation, it could be ideal in a  
122488 realtime environment for several reasons. The most important of these is that it is  
122489 deterministic: the highest-priority process is always run and, among processes of equal  
122490 priority, the process that has been runnable for the longest time is executed first. Because of  
122491 this determinism, cooperating processes can implement more complex scheduling simply  
122492 by altering their priority. For instance, if processes at a single priority were to reschedule  
122493 themselves at fixed time intervals, a time-slice policy would result.

122494 In a dedicated operating system in which all processes are well-behaved realtime  
122495 applications, non-migrating priority scheduling is sufficient. However, many existing  
122496 implementations provide for more complex scheduling policies.

122497 POSIX.1-2008 specifies a linear scheduling model. In this model, every process in the  
122498 system has a priority. The system scheduler always dispatches a process that has the  
122499 highest (generally the most time-critical) priority among all runnable processes in the  
122500 system. As long as there is only one such process, the dispatching policy is trivial. When  
122501 multiple processes of equal priority are eligible to run, they are ordered according to a  
122502 strict run-to-completion (FIFO) policy.

122503 The priority is represented as a positive integer and is inherited from the parent process.  
122504 For processes running under a fixed priority scheduling policy, the priority is never altered  
122505 except by an explicit function call.

122506 It was determined arbitrarily that larger integers correspond to “higher priorities”.

122507 Certain implementations might impose restrictions on the priority ranges to which  
122508 processes can be assigned. There also can be restrictions on the set of policies to which  
122509 processes can be set.

122510 • Requirements

122511 Realtime processes require that scheduling be fast and deterministic, and that it guarantees  
122512 to preempt lower priority processes.

122513 Thus, given the linear scheduling model, realtime processes require that they be run at a  
122514 priority that is higher than other processes. Within this framework, realtime processes are  
122515 free to yield execution resources to each other in a completely portable and  
122516 implementation-defined manner.

122517 As there is a generally perceived requirement for processes at the same priority level to  
122518 share processor resources more equitably, provisions are made by providing a scheduling  
122519 policy (that is, SCHED\_RR) intended to provide a timeslice-like facility.

122520 **Note:** The following topics assume that low numeric priority implies low scheduling criticality  
122521 and *vice versa*.

122522 • Rationale for New Interface

122523 Realtime applications need to be able to determine when processes will run in relation to  
122524 each other. It must be possible to guarantee that a critical process will run whenever it is  
122525 runnable; that is, whenever it wants to for as long as it needs. SCHED\_FIFO satisfies this  
122526 requirement. Additionally, SCHED\_RR was defined to meet a realtime requirement for a  
122527 well-defined time-sharing policy for processes at the same priority.

122528 It would be possible to use the BSD *setpriority()* and *getpriority()* functions by redefining  
122529 the meaning of the “nice” parameter according to the scheduling policy currently in use by



122530 the process. The System V *nice()* interface was felt to be undesirable for realtime because it  
122531 specifies an adjustment to the “nice” value, rather than setting it to an explicit value.  
122532 Realtime applications will usually want to set priority to an explicit value. Also, System V  
122533 *nice()* does not allow for changing the priority of another process.

122534 With the POSIX.1b interfaces, the traditional “nice” value does not affect the SCHED\_FIFO  
122535 or SCHED\_RR scheduling policies. If a “nice” value is supported, it is implementation-  
122536 defined whether it affects the SCHED\_OTHER policy.

122537 An important aspect of POSIX.1-2008 is the explicit description of the queuing and  
122538 preemption rules. It is critical, to achieve deterministic scheduling, that such rules be  
122539 stated clearly in POSIX.1-2008.

122540 POSIX.1-2008 does not address the interaction between priority and swapping. The issues  
122541 involved with swapping and virtual memory paging are extremely implementation-  
122542 defined and would be nearly impossible to standardize at this point. The proposed  
122543 scheduling paradigm, however, fully describes the scheduling behavior of runnable  
122544 processes, of which one criterion is that the working set be resident in memory. Assuming  
122545 the existence of a portable interface for locking portions of a process in memory, paging  
122546 behavior need not affect the scheduling of realtime processes.

122547 POSIX.1-2008 also does not address the priorities of “system” processes. In general, these  
122548 processes should always execute in low-priority ranges to avoid conflict with other  
122549 realtime processes. Implementations should document the priority ranges in which system  
122550 processes run.

122551 The default scheduling policy is not defined. The effect of I/O interrupts and other system  
122552 processing activities is not defined. The temporary lending of priority from one process to  
122553 another (such as for the purposes of affecting freeing resources) by the system is not  
122554 addressed. Preemption of resources is not addressed. Restrictions on the ability of a  
122555 process to affect other processes beyond a certain level (influence levels) is not addressed.

122556 The rationale used to justify the simple time-quantum scheduler is that it is common  
122557 practice to depend upon this type of scheduling to ensure “fair” distribution of processor  
122558 resources among portions of the application that must interoperate in a serial fashion. Note  
122559 that POSIX.1-2008 is silent with respect to the setting of this time quantum, or whether it is  
122560 a system-wide value or a per-process value, although it appears that the prevailing  
122561 realtime practice is for it to be a system-wide value.

122562 In a system with  $N$  processes at a given priority, all processor-bound, in which the time  
122563 quantum is equal for all processes at a specific priority level, the following assumptions  
122564 are made of such a scheduling policy:

- 122565 1. A time quantum  $Q$  exists and the current process will own control of the processor  
122566 for at least a duration of  $Q$  and will have the processor for a duration of  $Q$ .
- 122567 2. The  $N$ th process at that priority will control a processor within a duration of  $(N-1)$   
122568  $\times Q$ .

122569 These assumptions are necessary to provide equal access to the processor and bounded  
122570 response from the application.

122571 The assumptions hold for the described scheduling policy only if no system overhead,  
122572 such as interrupt servicing, is present. If the interrupt servicing load is non-zero, then one  
122573 of the two assumptions becomes fallacious, based upon how  $Q$  is measured by the system.

122574 If  $Q$  is measured by clock time, then the assumption that the process obtains a duration  $Q$   
122575 processor time is false if interrupt overhead exists. Indeed, a scenario can be constructed  
122576 with  $N$  processes in which a single process undergoes complete processor starvation if a

122577 peripheral device, such as an analog-to-digital converter, generates significant interrupt  
122578 activity periodically with a period of  $N \times Q$ .

122579 If  $Q$  is measured as actual processor time, then the assumption that the  $N$ th process runs in  
122580 within the duration  $(N-1) \times Q$  is false.

122581 It should be noted that SCHED\_FIFO suffers from interrupt-based delay as well. However,  
122582 for SCHED\_FIFO, the implied response of the system is “as soon as possible”, so that the  
122583 interrupt load for this case is a vendor selection and not a compliance issue.

122584 With this in mind, it is necessary either to complete the definition by including bounds on  
122585 the interrupt load, or to modify the assumptions that can be made about the scheduling  
122586 policy.

122587 Since the motivation of inclusion of the policy is common usage, and since current  
122588 applications do not enjoy the luxury of bounded interrupt load, item (2) above is sufficient  
122589 to express existing application needs and is less restrictive in the standard definition. No  
122590 difference in interface is necessary.

122591 In an implementation in which the time quantum is equal for all processes at a specific  
122592 priority, our assumptions can then be restated as:

- 122593 — A time quantum  $Q$  exists, and a processor-bound process will be rescheduled after a  
122594 duration of, at most,  $Q$ . Time quantum  $Q$  may be defined in either wall clock time or  
122595 execution time.
- 122596 — In general, the  $N$ th process of a priority level should wait no longer than  $(N-1) \times Q$   
122597 time to execute, assuming no processes exist at higher priority levels.
- 122598 — No process should wait indefinitely.

122599 For implementations supporting per-process time quanta, these assumptions can be  
122600 readily extended.

## 122601 **Sporadic Server Scheduling Policy**

122602 The sporadic server is a mechanism defined for scheduling aperiodic activities in time-critical  
122603 realtime systems. This mechanism reserves a certain bounded amount of execution capacity for  
122604 processing aperiodic events at a high priority level. Any aperiodic events that cannot be  
122605 processed within the bounded amount of execution capacity are executed in the background at a  
122606 low priority level. Thus, a certain amount of execution capacity can be guaranteed to be  
122607 available for processing periodic tasks, even under burst conditions in the arrival of aperiodic  
122608 processing requests (that is, a large number of requests in a short time interval). The sporadic  
122609 server also simplifies the schedulability analysis of the realtime system, because it allows  
122610 aperiodic processes or threads to be treated as if they were periodic. The sporadic server was  
122611 first described by Sprunt, et al.

122612 The key concept of the sporadic server is to provide and limit a certain amount of computation  
122613 capacity for processing aperiodic events at their assigned normal priority, during a time interval  
122614 called the “replenishment period”. Once the entity controlled by the sporadic server mechanism  
122615 is initialized with its period and execution-time budget attributes, it preserves its execution  
122616 capacity until an aperiodic request arrives. The request will be serviced (if there are no higher  
122617 priority activities pending) as long as there is execution capacity left. If the request is completed,  
122618 the actual execution time used to service it is subtracted from the capacity, and a replenishment  
122619 of this amount of execution time is scheduled to happen one replenishment period after the  
122620 arrival of the aperiodic request. If the request is not completed, because there is no execution  
122621 capacity left, then the aperiodic process or thread is assigned a lower background priority. For  
122622 each portion of consumed execution capacity the execution time used is replenished after one

replenishment period. At the time of replenishment, if the sporadic server was executing at a background priority level, its priority is elevated to the normal level. Other similar replenishment policies have been defined, but the one presented here represents a compromise between efficiency and implementation complexity.

The interface that appears in this section defines a new scheduling policy for threads and processes that behaves according to the rules of the sporadic server mechanism. Scheduling attributes are defined and functions are provided to allow the user to set and get the parameters that control the scheduling behavior of this mechanism, namely the normal and low priority, the replenishment period, the maximum number of pending replenishment operations, and the initial execution-time budget.

- Scheduling Aperiodic Activities

Virtually all realtime applications are required to process aperiodic activities. In many cases, there are tight timing constraints that the response to the aperiodic events must meet. Usual timing requirements imposed on the response to these events are:

- The effects of an aperiodic activity on the response time of lower priority activities must be controllable and predictable.
- The system must provide the fastest possible response time to aperiodic events.
- It must be possible to take advantage of all the available processing bandwidth not needed by time-critical activities to enhance average-case response times to aperiodic events.

Traditional methods for scheduling aperiodic activities are background processing, polling tasks, and direct event execution:

- Background processing consists of assigning a very low priority to the processing of aperiodic events. It utilizes all the available bandwidth in the system that has not been consumed by higher priority threads. However, it is very difficult, or impossible, to meet requirements on average-case response time, because the aperiodic entity has to wait for the execution of all other entities which have higher priority.
- Polling consists of creating a periodic process or thread for servicing aperiodic requests. At regular intervals, the polling entity is started and its services accumulated pending aperiodic requests. If no aperiodic requests are pending, the polling entity suspends itself until its next period. Polling allows the aperiodic requests to be processed at a higher priority level. However, worst and average-case response times of polling entities are a direct function of the polling period, and there is execution overhead for each polling period, even if no event has arrived. If the deadline of the aperiodic activity is short compared to the inter-arrival time, the polling frequency must be increased to guarantee meeting the deadline. For this case, the increase in frequency can dramatically reduce the efficiency of the system and, therefore, its capacity to meet all deadlines. Yet, polling represents a good way to handle a large class of practical problems because it preserves system predictability, and because the amortized overhead drops as load increases.
- Direct event execution consists of executing the aperiodic events at a high fixed-priority level. Typically, the aperiodic event is processed by an interrupt service routine as soon as it arrives. This technique provides predictable response times for aperiodic events, but makes the response times of all lower priority activities completely unpredictable under burst arrival conditions. Therefore, if the density of aperiodic event arrivals is unbounded, it may be a dangerous technique for time-critical systems. Yet, for those cases in which the physics of the system imposes a



bound on the event arrival rate, it is probably the most efficient technique.

- The sporadic server scheduling algorithm combines the predictability of the polling approach with the short response times of the direct event execution. Thus, it allows systems to meet an important class of application requirements that cannot be met by using the traditional approaches. Multiple sporadic servers with different attributes can be applied to the scheduling of multiple classes of aperiodic events, each with different kinds of timing requirements, such as individual deadlines, average response times, and so on. It also has many other interesting applications for realtime, such as scheduling producer/consumer tasks in time-critical systems, limiting the effects of faults on the estimation of task execution-time requirements, and so on.

- Existing Practice

The sporadic server has been used in different kinds of applications, including military avionics, robot control systems, industrial automation systems, and so on. There are examples of many systems that cannot be successfully scheduled using the classic approaches, such as direct event execution, or polling, and are schedulable using a sporadic server scheduler. The sporadic server algorithm itself can successfully schedule all systems scheduled with direct event execution or polling.

The sporadic server scheduling policy has been implemented as a commercial product in the run-time system of the Verdex Ada compiler. There are also many applications that have used a much less efficient application-level sporadic server. These realtime applications would benefit from a sporadic server scheduler implemented at the scheduler level.

- Library-Level *versus* Kernel-Level Implementation

The sporadic server interface described in this section requires the sporadic server policy to be implemented at the same level as the scheduler. This means that the process sporadic server must be implemented at the kernel level and the thread sporadic server policy implemented at the same level as the thread scheduler; that is, kernel or library level.

In an earlier interface for the sporadic server, this mechanism was implementable at a different level than the scheduler. This feature allowed the implementor to choose between an efficient scheduler-level implementation, or a simpler user or library-level implementation. However, the working group considered that this interface made the use of sporadic servers more complex, and that library-level implementations would lack some of the important functionality of the sporadic server, namely the limitation of the actual execution time of aperiodic activities. The working group also felt that the interface described in this chapter does not preclude library-level implementations of threads intended to provide efficient low-overhead scheduling for those threads that are not scheduled under the sporadic server policy.

- Range of Scheduling Priorities

Each of the scheduling policies supported in POSIX.1-2008 has an associated range of priorities. The priority ranges for each policy might or might not overlap with the priority ranges of other policies. For time-critical realtime applications it is usual for periodic and aperiodic activities to be scheduled together in the same processor. Periodic activities will usually be scheduled using the SCHED\_FIFO scheduling policy, while aperiodic activities may be scheduled using SCHED\_SPORADIC. Since the application developer will require complete control over the relative priorities of these activities in order to meet his timing requirements, it would be desirable for the priority ranges of SCHED\_FIFO and SCHED\_SPORADIC to overlap completely. Therefore, although POSIX.1-2008 does not

122719 require any particular relationship between the different priority ranges, it is  
122720 recommended that these two ranges should coincide.

122721 • Dynamically Setting the Sporadic Server Policy

122722 Several members of the working group requested that implementations should not be  
122723 required to support dynamically setting the sporadic server scheduling policy for a thread.  
122724 The reason is that this policy may have a high overhead for library-level implementations  
122725 of threads, and if threads are allowed to dynamically set this policy, this overhead can be  
122726 experienced even if the thread does not use that policy. By disallowing the dynamic setting  
122727 of the sporadic server scheduling policy, these implementations can accomplish efficient  
122728 scheduling for threads using other policies. If a strictly conforming application needs to  
122729 use the sporadic server policy, and is therefore willing to pay the overhead, it must set this  
122730 policy at the time of thread creation.

122731 • Limitation of the Number of Pending Replenishments

122732 The number of simultaneously pending replenishment operations must be limited for each  
122733 sporadic server for two reasons: an unlimited number of replenishment operations would  
122734 need an unlimited number of system resources to store all the pending replenishment  
122735 operations; on the other hand, in some implementations each replenishment operation will  
122736 represent a source of priority inversion (just for the duration of the replenishment  
122737 operation) and thus, the maximum amount of replenishments must be bounded to  
122738 guarantee bounded response times. The way in which the number of replenishments is  
122739 bounded is by lowering the priority of the sporadic server to *sched\_ss\_low\_priority* when  
122740 the number of pending replenishments has reached its limit. In this way, no new  
122741 replenishments are scheduled until the number of pending replenishments decreases.

122742 In the sporadic server scheduling policy defined in POSIX.1-2008, the application can  
122743 specify the maximum number of pending replenishment operations for a single sporadic  
122744 server, by setting the value of the *sched\_ss\_max\_repl* scheduling parameter. This value must  
122745 be between one and {SS\_REPL\_MAX}, which is a maximum limit imposed by the  
122746 implementation. The limit {SS\_REPL\_MAX} must be greater than or equal to  
122747 {\_POSIX\_SS\_REPL\_MAX}, which is defined to be four in POSIX.1-2008. The minimum  
122748 limit of four was chosen so that an application can at least guarantee that four different  
122749 aperiodic events can be processed during each interval of length equal to the  
122750 replenishment period.

122751 B.2.8.5 Clocks and Timers

122752 • Clocks

122753 POSIX.1-2008 and the ISO C standard both define functions for obtaining system time.  
122754 Implicit behind these functions is a mechanism for measuring passage of time. This  
122755 specification makes this mechanism explicit and calls it a clock. The CLOCK\_REALTIME  
122756 clock required by POSIX.1-2008 is a higher resolution version of the clock that maintains  
122757 POSIX.1 system time. This is a “system-wide” clock, in that it is visible to all processes  
122758 and, were it possible for multiple processes to all read the clock at the same time, they  
122759 would see the same value.

122760 An extensible interface was defined, with the ability for implementations to define  
122761 additional clocks. This was done because of the observation that many realtime platforms  
122762 support multiple clocks, and it was desired to fit this model within the standard interface.  
122763 But implementation-defined clocks need not represent actual hardware devices, nor are  
122764 they necessarily system-wide.

- Timers

Two timer types are required for a system to support realtime applications:

1. One-shot

A one-shot timer is a timer that is armed with an initial expiration time, either relative to the current time or at an absolute time (based on some timing base, such as time in seconds and nanoseconds since the Epoch). The timer expires once and then is disarmed. With the specified facilities, this is accomplished by setting the *it\_value* member of the *value* argument to the desired expiration time and the *it\_interval* member to zero.

2. Periodic

A periodic timer is a timer that is armed with an initial expiration time, again either relative or absolute, and a repetition interval. When the initial expiration occurs, the timer is reloaded with the repetition interval and continues counting. With the specified facilities, this is accomplished by setting the *it\_value* member of the *value* argument to the desired initial expiration time and the *it\_interval* member to the desired repetition interval.

For both of these types of timers, the time of the initial timer expiration can be specified in two ways:

1. Relative (to the current time)
2. Absolute

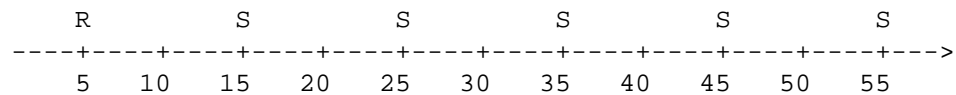
- Examples of Using Realtime Timers

In the diagrams below, *S* indicates a program schedule, *R* shows a schedule method request, and *E* suggests an internal operating system event.

— Periodic Timer: Data Logging

During an experiment, it might be necessary to log realtime data periodically to an internal buffer or to a mass storage device. With a periodic scheduling method, a logging module can be started automatically at fixed time intervals to log the data.

Program schedule is requested every 10 seconds.



[Time (in Seconds)]

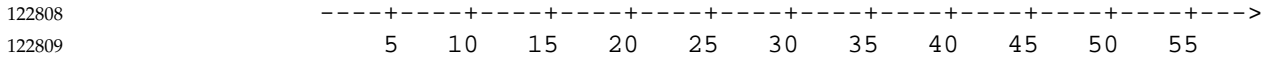
To achieve this type of scheduling using the specified facilities, one would allocate a per-process timer based on clock ID `CLOCK_REALTIME`. Then the timer would be armed via a call to `timer_settime()` with the `TIMER_ABSTIME` flag reset, and with an initial expiration value and a repetition interval of 10 seconds.

— One-shot Timer (Relative Time): Device Initialization

In an emission test environment, large sample bags are used to capture the exhaust from a vehicle. The exhaust is purged from these bags before each and every test. With a one-shot timer, a module could initiate the purge function and then suspend itself for a predetermined period of time while the sample bags are prepared.

Program schedule requested 20 seconds after call is issued.





122810 [Time (in Seconds)]

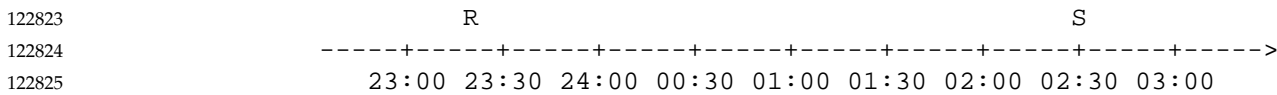
122811 To achieve this type of scheduling using the specified facilities, one would allocate a  
122812 per-process timer based on clock ID `CLOCK_REALTIME`. Then the timer would be  
122813 armed via a call to `timer_settime()` with the `TIMER_ABSTIME` flag reset, and with an  
122814 initial expiration value of 20 seconds and a repetition interval of zero.

122815 Note that if the program wishes merely to suspend itself for the specified interval, it  
122816 could more easily use `nanosleep()`.

122817 — One-shot Timer (Absolute Time): Data Transmission

122818 The results from an experiment are often moved to a different system within a  
122819 network for post-processing or archiving. With an absolute one-shot timer, a module  
122820 that moves data from a test-cell computer to a host computer can be automatically  
122821 scheduled on a daily basis.

122822 Program schedule requested for 2:30 a.m.



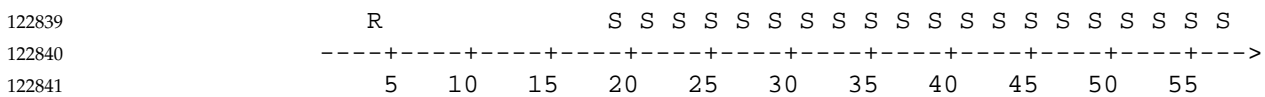
122826 [Time of Day]

122827 To achieve this type of scheduling using the specified facilities, a per-process timer  
122828 would be allocated based on clock ID `CLOCK_REALTIME`. Then the timer would be  
122829 armed via a call to `timer_settime()` with the `TIMER_ABSTIME` flag set, and an initial  
122830 expiration value equal to 2:30 a.m. of the next day.

122831 — Periodic Timer (Relative Time): Signal Stabilization

122832 Some measurement devices, such as emission analyzers, do not respond  
122833 instantaneously to an introduced sample. With a periodic timer with a relative initial  
122834 expiration time, a module that introduces a sample and records the average response  
122835 could suspend itself for a predetermined period of time while the signal is stabilized  
122836 and then sample at a fixed rate.

122837 Program schedule requested 15 seconds after call is issued and every 2 seconds  
122838 thereafter.



122842 [Time (in Seconds)]

122843 To achieve this type of scheduling using the specified facilities, one would allocate a  
122844 per-process timer based on clock ID `CLOCK_REALTIME`. Then the timer would be  
122845 armed via a call to `timer_settime()` with `TIMER_ABSTIME` flag reset, and with an  
122846 initial expiration value of 15 seconds and a repetition interval of 2 seconds.

122847 — Periodic Timer (Absolute Time): Work Shift-related Processing

122848 Resource utilization data is useful when time to perform experiments is being  
122849 scheduled at a facility. With a periodic timer with an absolute initial expiration time,  
122850 a module can be scheduled at the beginning of a work shift to gather resource  
122851 utilization data throughout the shift. This data can be used to allocate resources

122852 effectively to minimize bottlenecks and delays and maximize facility throughput.

122853 Program schedule requested for 2:00 a.m. and every 15 minutes thereafter.

122854 R S S S S S S

122855 -----+-----+-----+-----+-----+-----+-----+-----+----->

122856 23:00 23:30 24:00 00:30 01:00 01:30 02:00 02:30 03:00

122857 [Time of Day]

122858 To achieve this type of scheduling using the specified facilities, one would allocate a

122859 per-process timer based on clock ID `CLOCK_REALTIME`. Then the timer would be

122860 armed via a call to `timer_settime()` with `TIMER_ABSTIME` flag set, and with an initial

122861 expiration value equal to 2:00 a.m. and a repetition interval equal to 15 minutes.

122862 • Relationship of Timers to Clocks

122863 The relationship between clocks and timers armed with an absolute time is

122864 straightforward: a timer expiration signal is requested when the associated clock reaches

122865 or exceeds the specified time. The relationship between clocks and timers armed with a

122866 relative time (an interval) is less obvious, but not unintuitive. In this case, a timer

122867 expiration signal is requested when the specified interval, *as measured by the associated clock*,

122868 has passed. For the required `CLOCK_REALTIME` clock, this allows timer expiration

122869 signals to be requested at specified “wall clock” times (absolute), or when a specified

122870 interval of “realtime” has passed (relative). For an implementation-defined clock—say, a

122871 process virtual time clock—timer expirations could be requested when the process has

122872 used a specified total amount of virtual time (absolute), or when it has used a specified

122873 *additional* amount of virtual time (relative).

122874 The interfaces also allow flexibility in the implementation of the functions. For example, an

122875 implementation could convert all absolute times to intervals by subtracting the clock value

122876 at the time of the call from the requested expiration time and “counting down” at the

122877 supported resolution. Or it could convert all relative times to absolute expiration time by

122878 adding in the clock value at the time of the call and comparing the clock value to the

122879 expiration time at the supported resolution. Or it might even choose to maintain absolute

122880 times as absolute and compare them to the clock value at the supported resolution for

122881 absolute timers, and maintain relative times as intervals and count them down at the

122882 resolution supported for relative timers. The choice will be driven by efficiency

122883 considerations and the underlying hardware or software clock implementation.

122884 • Data Definitions for Clocks and Timers

122885 POSIX.1-2008 uses a time representation capable of supporting nanosecond resolution

122886 timers for the following reasons:

- 122887 — To enable POSIX.1-2008 to represent those computer systems already using
- 122888 nanosecond or submicrosecond resolution clocks.
- 122889 — To accommodate those per-process timers that might need nanoseconds to specify an
- 122890 absolute value of system-wide clocks, even though the resolution of the per-process
- 122891 timer may only be milliseconds, or *vice versa*.
- 122892 — Because the number of nanoseconds in a second can be represented in 32 bits.

122893 Time values are represented in the **timespec** structure. The `tv_sec` member is of type **time\_t**

122894 so that this member is compatible with time values used by POSIX.1 functions and the

122895 ISO C standard. The `tv_nsec` member is a **signed long** in order to simplify and clarify code

122896 that decrements or finds differences of time values. Note that because 1 billion (number of

122897 nanoseconds per second) is less than half of the value representable by a signed 32-bit



122898 value, it is always possible to add two valid fractional seconds represented as integral  
122899 nanoseconds without overflowing the signed 32-bit value.

122900 A maximum allowable resolution for the CLOCK\_REALTIME clock of 20 ms (1/50  
122901 seconds) was chosen to allow line frequency clocks in European countries to be  
122902 conforming. 60 Hz clocks in the US will also be conforming, as will finer granularity  
122903 clocks, although a Strictly Conforming Application cannot assume a granularity of less  
122904 than 20 ms (1/50 seconds).

122905 The minimum allowable maximum time allowed for the CLOCK\_REALTIME clock and  
122906 the function *nanosleep()*, and timers created with *clock\_id=CLOCK\_REALTIME*, is  
122907 determined by the fact that the *tv\_sec* member is of type **time\_t**.

122908 POSIX.1-2008 specifies that timer expirations must not be delivered early, and *nanosleep()*  
122909 must not return early due to quantization error. POSIX.1-2008 discusses the various  
122910 implementations of *alarm()* in the rationale and states that implementations that do not  
122911 allow alarm signals to occur early are the most appropriate, but refrained from mandating  
122912 this behavior. Because of the importance of predictability to realtime applications,  
122913 POSIX.1-2008 takes a stronger stance.

122914 The standard developers considered using a time representation that differs from  
122915 POSIX.1b in the second 32 bit of the 64-bit value. Whereas POSIX.1b defines this field as a  
122916 fractional second in nanoseconds, the other methodology defines this as a binary fraction  
122917 of one second, with the radix point assumed before the most significant bit.

122918 POSIX.1b is a software, source-level standard and most of the benefits of the alternate  
122919 representation are enjoyed by hardware implementations of clocks and algorithms. It was  
122920 felt that mandating this format for POSIX.1b clocks and timers would unnecessarily  
122921 burden the application developer with writing, possibly non-portable, multiple precision  
122922 arithmetic packages to perform conversion between binary fractions and integral units  
122923 such as nanoseconds, milliseconds, and so on.

#### 122924 **Rationale for the Monotonic Clock**

122925 For those applications that use time services to achieve realtime behavior, changing the value of  
122926 the clock on which these services rely may cause erroneous timing behavior. For these  
122927 applications, it is necessary to have a monotonic clock which cannot run backwards, and which  
122928 has a maximum clock jump that is required to be documented by the implementation.  
122929 Additionally, it is desirable (but not required by POSIX.1-2008) that the monotonic clock  
122930 increases its value uniformly. This clock should not be affected by changes to the system time;  
122931 for example, to synchronize the clock with an external source or to account for leap seconds.  
122932 Such changes would cause errors in the measurement of time intervals for those time services  
122933 that use the absolute value of the clock.

122934 One could argue that by defining the behavior of time services when the value of a clock is  
122935 changed, deterministic realtime behavior can be achieved. For example, one could specify that  
122936 relative time services should be unaffected by changes in the value of a clock. However, there are  
122937 time services that are based upon an absolute time, but that are essentially intended as relative  
122938 time services. For example, *pthread\_cond\_timedwait()* uses an absolute time to allow it to wake  
122939 up after the required interval despite spurious wakeups. Although sometimes the  
122940 *pthread\_cond\_timedwait()* timeouts are absolute in nature, there are many occasions in which they  
122941 are relative, and their absolute value is determined from the current time plus a relative time  
122942 interval. In this latter case, if the clock changes while the thread is waiting, the wait interval will  
122943 not be the expected length. If a *pthread\_cond\_timedwait()* function were created that would take a  
122944 relative time, it would not solve the problem because to retain the intended “deadline” a thread  
122945 would need to compensate for latency due to the spurious wakeup, and preemption between

wakeup and the next wait.

The solution is to create a new monotonic clock, whose value does not change except for the regular ticking of the clock, and use this clock for implementing the various relative timeouts that appear in the different POSIX interfaces, as well as allow `pthread_cond_timedwait()` to choose this new clock for its timeout. A new `clock_nanosleep()` function is created to allow an application to take advantage of this newly defined clock. Notice that the monotonic clock may be implemented using the same hardware clock as the system clock.

Relative timeouts for `sigtimedwait()` and `aio_suspend()` have been redefined to use the monotonic clock, if present. The `alarm()` function has not been redefined, because the same effect but with better resolution can be achieved by creating a timer (for which the appropriate clock may be chosen).

The `pthread_cond_timedwait()` function has been treated in a different way, compared to other functions with absolute timeouts, because it is used to wait for an event, and thus it may have a deadline, while the other timeouts are generally used as an error recovery mechanism, and for them the use of the monotonic clock is not so important. Since the desired timeout for the `pthread_cond_timedwait()` function may either be a relative interval or an absolute time of day deadline, a new initialization attribute has been created for condition variables to specify the clock that is used for measuring the timeout in a call to `pthread_cond_timedwait()`. In this way, if a relative timeout is desired, the monotonic clock will be used; if an absolute deadline is required instead, the `CLOCK_REALTIME` or another appropriate clock may be used. This capability has not been added to other functions with absolute timeouts because for those functions the expected use of the timeout is mostly to prevent errors, and not so often to meet precise deadlines. As a consequence, the complexity of adding this capability is not justified by its perceived application usage.

The `nanosleep()` function has not been modified with the introduction of the monotonic clock. Instead, a new `clock_nanosleep()` function has been created, in which the desired clock may be specified in the function call.

- History of Resolution Issues

Due to the shift from relative to absolute timeouts in IEEE Std 1003.1d-1999, the amendments to the `sem_timedwait()`, `pthread_mutex_timedlock()`, `mq_timedreceive()`, and `mq_timedsend()` functions of that standard have been removed. Those amendments specified that `CLOCK_MONOTONIC` would be used for the (relative) timeouts if the Monotonic Clock option was supported.

Having these functions continue to be tied solely to `CLOCK_MONOTONIC` would not work. Since the absolute value of a time value obtained from `CLOCK_MONOTONIC` is unspecified, under the absolute timeouts interface, applications would behave differently depending on whether the Monotonic Clock option was supported or not (because the absolute value of the clock would have different meanings in either case).

Two options were considered:

1. Leave the current behavior unchanged, which specifies the `CLOCK_REALTIME` clock for these (absolute) timeouts, to allow portability of applications between implementations supporting or not the Monotonic Clock option.
2. Modify these functions in the way that `pthread_cond_timedwait()` was modified to allow a choice of clock, so that an application could use `CLOCK_REALTIME` when it is trying to achieve an absolute timeout and `CLOCK_MONOTONIC` when it is trying to achieve a relative timeout.

It was decided that the features of `CLOCK_MONOTONIC` are not as critical to these

122993 functions as they are to `pthread_cond_timedwait()`. The `pthread_cond_timedwait()` function is  
122994 given a relative timeout; the timeout may represent a deadline for an event. When these  
122995 functions are given relative timeouts, the timeouts are typically for error recovery  
122996 purposes and need not be so precise.

122997 Therefore, it was decided that these functions should be tied to `CLOCK_REALTIME` and  
122998 not complicated by being given a choice of clock.

## 122999 **Execution Time Monitoring**

### 123000 • Introduction

123001 The main goals of the execution time monitoring facilities defined in this chapter are to  
123002 measure the execution time of processes and threads and to allow an application to  
123003 establish CPU time limits for these entities.

123004 The analysis phase of time-critical realtime systems often relies on the measurement of  
123005 execution times of individual threads or processes to determine whether the timing  
123006 requirements will be met. Also, performance analysis techniques for soft deadline realtime  
123007 systems rely heavily on the determination of these execution times. The execution time  
123008 monitoring functions provide application developers with the ability to measure these  
123009 execution times online and open the possibility of dynamic execution-time analysis and  
123010 system reconfiguration, if required.

123011 The second goal of allowing an application to establish execution time limits for individual  
123012 processes or threads and detecting when they overrun allows program robustness to be  
123013 increased by enabling online checking of the execution times.

123014 If errors are detected—possibly because of erroneous program constructs, the existence of  
123015 errors in the analysis phase, or a burst of event arrivals—online detection and recovery is  
123016 possible in a portable way. This feature can be extremely important for many time-critical  
123017 applications. Other applications require trapping CPU-time errors as a normal way to exit  
123018 an algorithm; for instance, some realtime artificial intelligence applications trigger a  
123019 number of independent inference processes of varying accuracy and speed, limit how long  
123020 they can run, and pick the best answer available when time runs out. In many periodic  
123021 systems, overrun processes are simply restarted in the next resource period, after necessary  
123022 end-of-period actions have been taken. This allows algorithms that are inherently data-  
123023 dependent to be made predictable.

123024 The interface that appears in this chapter defines a new type of clock, the CPU-time clock,  
123025 which measures execution time. Each process or thread can invoke the clock and timer  
123026 functions defined in POSIX.1 to use them. Functions are also provided to access the CPU-  
123027 time clock of other processes or threads to enable remote monitoring of these clocks.  
123028 Monitoring of threads of other processes is not supported, since these threads are not  
123029 visible from outside of their own process with the interfaces defined in POSIX.1.

### 123030 • Execution Time Monitoring Interface

123031 The clock and timer interface defined in POSIX.1 historically only defined one clock, which  
123032 measures wall-clock time. The requirements for measuring execution time of processes and  
123033 threads, and setting limits to their execution time by detecting when they overrun, can be  
123034 accomplished with that interface if a new kind of clock is defined. These new clocks  
123035 measure execution time, and one is associated with each process and with each thread. The  
123036 clock functions currently defined in POSIX.1 can be used to read and set these CPU-time  
123037 clocks, and timers can be created using these clocks as their timing base. These timers can  
123038 then be used to send a signal when some specified execution time has been exceeded. The  
123039 CPU-time clocks of each process or thread can be accessed by using the symbols



123040 CLOCK\_PROCESS\_CPUTIME\_ID or CLOCK\_THREAD\_CPUTIME\_ID.

123041 The clock and timer interface defined in POSIX.1 and extended with the new kind of CPU-  
123042 time clock would only allow processes or threads to access their own CPU-time clocks.  
123043 However, many realtime systems require the possibility of monitoring the execution time  
123044 of processes or threads from independent monitoring entities. In order to allow  
123045 applications to construct independent monitoring entities that do not require cooperation  
123046 from or modification of the monitored entities, two functions have been added:  
123047 *clock\_getcpuclockid()*, for accessing CPU-time clocks of other processes, and  
123048 *pthread\_getcpuclockid()*, for accessing CPU-time clocks of other threads. These functions  
123049 return the clock identifier associated with the process or thread specified in the call. These  
123050 clock IDs can then be used in the rest of the clock function calls.

123051 The clocks accessed through these functions could also be used as a timing base for the  
123052 creation of timers, thereby allowing independent monitoring entities to limit the CPU time  
123053 consumed by other entities. However, this possibility would imply additional complexity  
123054 and overhead because of the need to maintain a timer queue for each process or thread, to  
123055 store the different expiration times associated with timers created by different processes or  
123056 threads. The working group decided this additional overhead was not justified by  
123057 application requirements. Therefore, creation of timers attached to the CPU-time clocks of  
123058 other processes or threads has been specified as implementation-defined.

123059 • Overhead Considerations

123060 The measurement of execution time may introduce additional overhead in the thread  
123061 scheduling, because of the need to keep track of the time consumed by each of these  
123062 entities. In library-level implementations of threads, the efficiency of scheduling could be  
123063 somehow compromised because of the need to make a kernel call, at each context switch,  
123064 to read the process CPU-time clock. Consequently, a thread creation attribute called *cpu-*  
123065 *clock-requirement* was defined, to allow threads to disconnect their respective CPU-time  
123066 clocks. However, the Ballot Group considered that this attribute itself introduced some  
123067 overhead, and that in current implementations it was not worth the effort. Therefore, the  
123068 attribute was deleted, and thus thread CPU-time clocks are required for all threads if the  
123069 Thread CPU-Time Clocks option is supported.

123070 • Accuracy of CPU-Time Clocks

123071 The mechanism used to measure the execution time of processes and threads is specified in  
123072 POSIX.1-2008 as implementation-defined. The reason for this is that both the underlying  
123073 hardware and the implementation architecture have a very strong influence on the  
123074 accuracy achievable for measuring CPU time. For some implementations, the specification  
123075 of strict accuracy requirements would represent very large overheads, or even the  
123076 impossibility of being implemented.

123077 Since the mechanism for measuring execution time is implementation-defined, realtime  
123078 applications will be able to take advantage of accurate implementations using a portable  
123079 interface. Of course, strictly conforming applications cannot rely on any particular degree  
123080 of accuracy, in the same way as they cannot rely on a very accurate measurement of wall  
123081 clock time. There will always exist applications whose accuracy or efficiency requirements  
123082 on the implementation are more rigid than the values defined in POSIX.1-2008 or any  
123083 other standard.

123084 In any case, there is a minimum set of characteristics that realtime applications would  
123085 expect from most implementations. One such characteristic is that the sum of all the  
123086 execution times of all the threads in a process equals the process execution time, when no  
123087 CPU-time clocks are disabled. This need not always be the case because implementations  
123088 may differ in how they account for time during context switches. Another characteristic is

that the sum of the execution times of all processes in a system equals the number of processors, multiplied by the elapsed time, assuming that no processor is idle during that elapsed time. However, in some implementations it might not be possible to relate CPU time to elapsed time. For example, in a heterogeneous multi-processor system in which each processor runs at a different speed, an implementation may choose to define each “second” of CPU time to be a certain number of “cycles” that a CPU has executed.

- Existing Practice

Measuring and limiting the execution time of each concurrent activity are common features of most industrial implementations of realtime systems. Almost all critical realtime systems are currently built upon a cyclic executive. With this approach, a regular timer interrupt kicks off the next sequence of computations. It also checks that the current sequence has completed. If it has not, then some error recovery action can be undertaken (or at least an overrun is avoided). Current software engineering principles and the increasing complexity of software are driving application developers to implement these systems on multi-threaded or multi-process operating systems. Therefore, if a POSIX operating system is to be used for this type of application, then it must offer the same level of protection.

Execution time clocks are also common in most UNIX implementations, although these clocks usually have requirements different from those of realtime applications. The POSIX.1 *times()* function supports the measurement of the execution time of the calling process, and its terminated child processes. This execution time is measured in clock ticks and is supplied as two different values with the user and system execution times, respectively. BSD supports the function *getrusage()*, which allows the calling process to get information about the resources used by itself and/or all of its terminated child processes. The resource usage includes user and system CPU time. Some UNIX systems have options to specify high resolution (up to one microsecond) CPU-time clocks using the *times()* or the *getrusage()* functions.

The *times()* and *getrusage()* interfaces do not meet important realtime requirements, such as the possibility of monitoring execution time from a different process or thread, or the possibility of detecting an execution time overrun. The latter requirement is supported in some UNIX implementations that are able to send a signal when the execution time of a process has exceeded some specified value. For example, BSD defines the functions *getitimer()* and *setitimer()*, which can operate either on a realtime clock (wall-clock), or on virtual-time or profile-time clocks which measure CPU time in two different ways. These functions do not support access to the execution time of other processes.

The IBM MVS operating system supports per-process and per-thread execution time clocks. It also supports limiting the execution time of a given process.

Given all this existing practice, the working group considered that the POSIX.1 clocks and timers interface was appropriate to meet most of the requirements that realtime applications have for execution time clocks. Functions were added to get the CPU time clock IDs, and to allow/disallow the thread CPU-time clocks (in order to preserve the efficiency of some implementations of threads).

- Clock Constants

The definition of the manifest constants `CLOCK_PROCESS_CPUTIME_ID` and `CLOCK_THREAD_CPUTIME_ID` allows processes or threads, respectively, to access their own execution-time clocks. However, given a process or thread, access to its own execution-time clock is also possible if the clock ID of this clock is obtained through a call to *clock\_getcpuclockid()* or *pthread\_getcpuclockid()*. Therefore, these constants are not necessary and could be deleted to make the interface simpler. Their existence saves one

system call in the first access to the CPU-time clock of each process or thread. The working group considered this issue and decided to leave the constants in POSIX.1-2008 because they are closer to the POSIX.1b use of clock identifiers.

- Library Implementations of Threads

In library implementations of threads, kernel entities and library threads can coexist. In this case, if the CPU-time clocks are supported, most of the clock and timer functions will need to have two implementations: one in the thread library, and one in the system calls library. The main difference between these two implementations is that the thread library implementation will have to deal with clocks and timers that reside in the thread space, while the kernel implementation will operate on timers and clocks that reside in kernel space. In the library implementation, if the clock ID refers to a clock that resides in the kernel, a kernel call will have to be made. The correct version of the function can be chosen by specifying the appropriate order for the libraries during the link process.

- History of Resolution Issues: Deletion of the *enable* Attribute

In early proposals, consideration was given to inclusion of an attribute called *enable* for CPU-time clocks. This would allow implementations to avoid the overhead of measuring execution time for those processes or threads for which this measurement was not required. However, this is unnecessary since processes are already required to measure execution time by the POSIX.1 *times()* function. Consequently, the *enable* attribute is not present.

## Rationale Relating to Timeouts

- Requirements for Timeouts

Realtime systems which must operate reliably over extended periods without human intervention are characteristic in embedded applications such as avionics, machine control, and space exploration, as well as more mundane applications such as cable TV, security systems, and plant automation. A multi-tasking paradigm, in which many independent and/or cooperating software functions relinquish the processor(s) while waiting for a specific stimulus, resource, condition, or operation completion, is very useful in producing well engineered programs for such systems. For such systems to be robust and fault-tolerant, expected occurrences that are unduly delayed or that never occur must be detected so that appropriate recovery actions may be taken. This is difficult if there is no way for a task to regain control of a processor once it has relinquished control (blocked) awaiting an occurrence which, perhaps because of corrupted code, hardware malfunction, or latent software bugs, will not happen when expected. Therefore, the common practice in realtime operating systems is to provide a capability to time out such blocking services. Although there are several methods to achieve this already defined by POSIX, none are as reliable or efficient as initiating a timeout simultaneously with initiating a blocking service. This is especially critical in hard-realtime embedded systems because the processors typically have little time reserve, and allowed fault recovery times are measured in milliseconds rather than seconds.

The working group largely agreed that such timeouts were necessary and ought to become part of POSIX.1-2008, particularly vendors of realtime operating systems whose customers had already expressed a strong need for timeouts. There was some resistance to inclusion of timeouts in POSIX.1-2008 because the desired effect, fault tolerance, could, in theory, be achieved using existing facilities and alternative software designs, but there was no compelling evidence that realtime system designers would embrace such designs at the sacrifice of performance and/or simplicity.

- Which Services should be Timed Out?

Originally, the working group considered the prospect of providing timeouts on all blocking services, including those currently existing in POSIX.1, POSIX.1b, and POSIX.1c, and future interfaces to be defined by other working groups, as sort of a general policy. This was rather quickly rejected because of the scope of such a change, and the fact that many of those services would not normally be used in a realtime context. More traditional timesharing solutions to timeout would suffice for most of the POSIX.1 interfaces, while others had asynchronous alternatives which, while more complex to utilize, would be adequate for some realtime and all non-realtime applications.

The list of potential candidates for timeouts was narrowed to the following for further consideration:

- POSIX.1b
  - `sem_wait()`
  - `mq_receive()`
  - `mq_send()`
  - `lio_listio()`
  - `aio_suspend()`
  - `sigwait()` (timeout already implemented by `sigtimedwait()`)
- POSIX.1c
  - `pthread_mutex_lock()`
  - `pthread_join()`
  - `pthread_cond_wait()`  
(timeout already implemented by `pthread_cond_timedwait()`)
- POSIX.1
  - `read()`
  - `write()`

After further review by the working group, the `lio_listio()`, `read()`, and `write()` functions (all forms of blocking synchronous I/O) were eliminated from the list because of the following:

- Asynchronous alternatives exist
- Timeouts can be implemented, albeit non-portably, in device drivers
- A strong desire not to introduce modifications to POSIX.1 interfaces

The working group ultimately rejected `pthread_join()` since both that interface and a timed variant of that interface are non-minimal and may be implemented as a function. See below for a library implementation of `pthread_join()`.

Thus, there was a consensus among the working group members to add timeouts to 4 of the remaining 5 functions (the timeout for `aio_suspend()` was ultimately added directly to POSIX.1b, while the others were added by POSIX.1d). However, `pthread_mutex_lock()` remained contentious.

Many feel that `pthread_mutex_lock()` falls into the same class as the other functions; that is, it is desirable to time out a mutex lock because a mutex may fail to be unlocked due to

errant or corrupted code in a critical section (looping or branching outside of the unlock code), and therefore is equally in need of a reliable, simple, and efficient timeout. In fact, since mutexes are intended to guard small critical sections, most `pthread_mutex_lock()` calls would be expected to obtain the lock without blocking nor utilizing any kernel service, even in implementations of threads with global contention scope; the timeout alternative need only be considered after it is determined that the thread must block.

Those opposed to timing out mutexes feel that the very simplicity of the mutex is compromised by adding a timeout semantic, and that to do so is senseless. They claim that if a timed mutex is really deemed useful by a particular application, then it can be constructed from the facilities already in POSIX.1b and POSIX.1c. The following two C-language library implementations of mutex locking with timeout represent the solutions offered (in both implementations, the timeout parameter is specified as absolute time, not relative time as in the proposed POSIX.1c interfaces).

- Spinlock Implementation

```
#include <pthread.h>
#include <time.h>
#include <errno.h>

int pthread_mutex_timedlock(pthread_mutex_t *mutex,
                           const struct timespec *timeout)
{
    struct timespec timenow;

    while (pthread_mutex_trylock(mutex) == EBUSY)
    {
        clock_gettime(CLOCK_REALTIME, &timenow);
        if (timespec_cmp(&timenow, timeout) >= 0)
        {
            return ETIMEDOUT;
        }
        pthread_yield();
    }
    return 0;
}
```

The Spinlock implementation is generally unsuitable for any application using priority-based thread scheduling policies such as SCHED\_FIFO or SCHED\_RR, since the mutex could currently be held by a thread of lower priority within the same allocation domain, but since the waiting thread never blocks, only threads of equal or higher priority will ever run, and the mutex cannot be unlocked. Setting priority inheritance or priority ceiling protocol on the mutex does not solve this problem, since the priority of a mutex owning thread is only boosted if higher priority threads are blocked waiting for the mutex; clearly not the case for this spinlock.

- Condition Wait Implementation

```
#include <pthread.h>
#include <time.h>
#include <errno.h>

struct timed_mutex
{
    int locked;
    pthread_mutex_t mutex;
}
```

```

123274         pthread_cond_t cond;
123275     };
123276     typedef struct timed_mutex timed_mutex_t;
123277     int timed_mutex_lock(timed_mutex_t *tm,
123278         const struct timespec *timeout)
123279     {
123280         int timedout=FALSE;
123281         int error_status;
123282
123283         pthread_mutex_lock(&tm->mutex);
123284
123285         while (tm->locked && !timedout)
123286         {
123287             if ((error_status=pthread_cond_timedwait(&tm->cond,
123288                 &tm->mutex,
123289                 timeout))!=0)
123290             {
123291                 if (error_status==ETIMEDOUT) timedout = TRUE;
123292             }
123293         }
123294
123295         if(timedout)
123296         {
123297             pthread_mutex_unlock(&tm->mutex);
123298             return ETIMEDOUT;
123299         }
123300         else
123301         {
123302             tm->locked = TRUE;
123303             pthread_mutex_unlock(&tm->mutex);
123304             return 0;
123305         }
123306     }
123307
123308     void timed_mutex_unlock(timed_mutex_t *tm)
123309     {
123310         pthread_mutex_lock(&tm->mutex); / for case assignment not atomic /
123311         tm->locked = FALSE;
123312         pthread_mutex_unlock(&tm->mutex);
123313         pthread_cond_signal(&tm->cond);
123314     }

```

The Condition Wait implementation effectively substitutes the *pthread\_cond\_timedwait()* function (which is currently timed out) for the desired *pthread\_mutex\_timedlock()*. Since waits on condition variables currently do not include protocols which avoid priority inversion, this method is generally unsuitable for realtime applications because it does not provide the same priority inversion protection as the untimed *pthread\_mutex\_lock()*. Also, for any given implementations of the current mutex and condition variable primitives, this library implementation has a performance cost at least 2.5 times that of the untimed *pthread\_mutex\_lock()* even in the case where the timed mutex is readily locked without blocking (the interfaces required for this case are shown in bold). Even in uniprocessors or where assignment is atomic, at least an additional *pthread\_cond\_signal()* is required. *pthread\_mutex\_timedlock()* could be implemented at effectively no performance penalty in this case because the timeout parameters need only be considered after it is determined that the mutex cannot be locked immediately.



123324 Thus it has not yet been shown that the full semantics of mutex locking with timeout can  
 123325 be efficiently and reliably achieved using existing interfaces. Even if the existence of an  
 123326 acceptable library implementation were proven, it is difficult to justify why the interface  
 123327 itself should not be made portable, especially considering approval for the other four  
 123328 timeouts.

- 123329 • Rationale for Library Implementation of *pthread\_timedjoin()*

123330 Library implementation of *pthread\_timedjoin()*:

```

123331 /*
123332  * Construct a thread variety entirely from existing functions
123333  * with which a join can be done, allowing the join to time out.
123334  */
123335 #include <pthread.h>
123336 #include <time.h>

123337 struct timed_thread {
123338     pthread_t t;
123339     pthread_mutex_t m;
123340     int exiting;
123341     pthread_cond_t exit_c;
123342     void *(*start_routine)(void *arg);
123343     void *arg;
123344     void *status;
123345 };

123346 typedef struct timed_thread *timed_thread_t;
123347 static pthread_key_t timed_thread_key;
123348 static pthread_once_t timed_thread_once = PTHREAD_ONCE_INIT;

123349 static void timed_thread_init()
123350 {
123351     pthread_key_create(&timed_thread_key, NULL);
123352 }

123353 static void *timed_thread_start_routine(void *args)
123354 /*
123355  * Routine to establish thread-specific data value and run the actual
123356  * thread start routine which was supplied to timed_thread_create().
123357  */
123358 {
123359     timed_thread_t tt = (timed_thread_t) args;
123360     pthread_once(&timed_thread_once, timed_thread_init);
123361     pthread_setspecific(timed_thread_key, (void *)tt);
123362     timed_thread_exit((tt->start_routine)(tt->arg));
123363 }

123364 int timed_thread_create(timed_thread_t ttp, const pthread_attr_t *attr,
123365     void *(*start_routine)(void *), void *arg)
123366 /*
123367  * Allocate a thread which can be used with timed_thread_join().
123368  */
123369 {
123370     timed_thread_t tt;
  
```

```

123371         int result;
123372         tt = (timed_thread_t) malloc(sizeof(struct timed_thread));
123373         pthread_mutex_init(&tt->m, NULL);
123374         tt->exiting = FALSE;
123375         pthread_cond_init(&tt->exit_c, NULL);
123376         tt->start_routine = start_routine;
123377         tt->arg = arg;
123378         tt->status = NULL;
123379
123380         if ((result = pthread_create(&tt->t, attr,
123381             timed_thread_start_routine, (void *)tt)) != 0) {
123382             free(tt);
123383             return result;
123384         }
123385         pthread_detach(tt->t);
123386         ttp = tt;
123387         return 0;
123388     }
123389
123390     int timed_thread_join(timed_thread_t tt,
123391         struct timespec *timeout,
123392         void **status)
123393     {
123394         int result;
123395
123396         pthread_mutex_lock(&tt->m);
123397         result = 0;
123398         /*
123399          * Wait until the thread announces that it is exiting,
123400          * or until timeout.
123401          */
123402         while (result == 0 && ! tt->exiting) {
123403             result = pthread_cond_timedwait(&tt->exit_c, &tt->m, timeout);
123404         }
123405         pthread_mutex_unlock(&tt->m);
123406         if (result == 0 && tt->exiting) {
123407             *status = tt->status;
123408             free((void *)tt);
123409             return result;
123410         }
123411         return result;
123412     }
123413
123414     void timed_thread_exit(void *status)
123415     {
123416         timed_thread_t tt;
123417         void *specific;
123418
123419         if ((specific=pthread_getspecific(timed_thread_key)) == NULL){
123420             /*
123421              * Handle cases which won't happen with correct usage.
123422              */
123423             pthread_exit( NULL);
123424         }
123425     }

```



```

123420         tt = (timed_thread_t) specific;
123421         pthread_mutex_lock(&tt->m);
123422         /*
123423          * Tell a joiner that we're exiting.
123424          */
123425         tt->status = status;
123426         tt->exiting = TRUE;
123427         pthread_cond_signal(&tt->exit_c);
123428         pthread_mutex_unlock(&tt->m);
123429         /*
123430          * Call pthread exit() to call destructors and really
123431          * exit the thread.
123432          */
123433         pthread_exit(NULL);
123434     }

```

The *pthread\_join()* C-language example shown above demonstrates that it is possible, using existing pthread facilities, to construct a variety of thread which allows for joining such a thread, but which allows the join operation to time out. It does this by using a *pthread\_cond\_timedwait()* to wait for the thread to exit. A **timed\_thread\_t** descriptor structure is used to pass parameters from the creating thread to the created thread, and from the exiting thread to the joining thread. This implementation is roughly equivalent to what a normal *pthread\_join()* implementation would do, with the single change being that *pthread\_cond\_timedwait()* is used in place of a simple *pthread\_cond\_wait()*.

Since it is possible to implement such a facility entirely from existing pthread interfaces, and with roughly equal efficiency and complexity to an implementation which would be provided directly by a pthreads implementation, it was the consensus of the working group members that any *pthread\_timedjoin()* facility would be unnecessary, and should not be provided.

- Form of the Timeout Interfaces

The working group considered a number of alternative ways to add timeouts to blocking services. At first, a system interface which would specify a one-shot or persistent timeout to be applied to subsequent blocking services invoked by the calling process or thread was considered because it allowed all blocking services to be timed out in a uniform manner with a single additional interface; this was rather quickly rejected because it could easily result in the wrong services being timed out.

It was suggested that a timeout value might be specified as an attribute of the object (semaphore, mutex, message queue, and so on), but there was no consensus on this, either on a case-by-case basis or for all timeouts.

Looking at the two existing timeouts for blocking services indicates that the working group members favor a separate interface for the timed version of a function. However, *pthread\_cond\_timedwait()* utilizes an absolute timeout value while *sigtimedwait()* uses a relative timeout value. The working group members agreed that relative timeout values are appropriate where the timeout mechanism's primary use was to deal with an unexpected or error situation, but they are inappropriate when the timeout must expire at a particular time, or before a specific deadline. For the timeouts being introduced in POSIX.1-2008, the working group considered allowing both relative and absolute timeouts as is done with POSIX.1b timers, but ultimately favored the simpler absolute timeout form.

An absolute time measure can be easily implemented on top of an interface that specifies relative time, by reading the clock, calculating the difference between the current time and

123469 the desired wakeup time, and issuing a relative timeout call. But there is a race condition  
123470 with this approach because the thread could be preempted after reading the clock, but  
123471 before making the timed-out call; in this case, the thread would be awakened later than it  
123472 should and, thus, if the wakeup time represented a deadline, it would miss it.

123473 There is also a race condition when trying to build a relative timeout on top of an interface  
123474 that specifies absolute timeouts. In this case, the clock would have to be read to calculate  
123475 the absolute wakeup time as the sum of the current time plus the relative timeout interval.  
123476 In this case, if the thread is preempted after reading the clock but before making the timed-  
123477 out call, the thread would be awakened earlier than desired.

123478 But the race condition with the absolute timeouts interface is not as bad as the one that  
123479 happens with the relative timeout interface, because there are simple workarounds. For the  
123480 absolute timeouts interface, if the timing requirement is a deadline, the deadline can still  
123481 be met because the thread woke up earlier than the deadline. If the timeout is just used as  
123482 an error recovery mechanism, the precision of timing is not really important. If the timing  
123483 requirement is that between actions A and B a minimum interval of time must elapse, the  
123484 absolute timeout interface can be safely used by reading the clock after action A has been  
123485 started. It could be argued that, since the call with the absolute timeout is atomic from the  
123486 application point of view, it is not possible to read the clock after action A, if this action is  
123487 part of the timed-out call. But looking at the nature of the calls for which timeouts are  
123488 specified (locking a mutex, waiting for a semaphore, waiting for a message, or waiting  
123489 until there is space in a message queue), the timeouts that an application would build on  
123490 these actions would not be triggered by these actions themselves, but by some other  
123491 external action. For example, if waiting for a message to arrive to a message queue, and  
123492 waiting for at least 20 milliseconds, this time interval would start to be counted from some  
123493 event that would trigger both the action that produces the message, as well as the action  
123494 that waits for the message to arrive, and not by the wait-for-message operation itself. In  
123495 this case, the workaround proposed above could be used.

123496 For these reasons, the absolute timeout is preferred over the relative timeout interface.

## 123497 **B.2.9 Threads**

123498 Threads will normally be more expensive than subroutines (or functions, routines, and so on) if  
123499 specialized hardware support is not provided. Nevertheless, threads should be sufficiently  
123500 efficient to encourage their use as a medium to fine-grained structuring mechanism for  
123501 parallelism in an application. Structuring an application using threads then allows it to take  
123502 immediate advantage of any underlying parallelism available in the host environment. This  
123503 means implementors are encouraged to optimize for fast execution at the possible expense of  
123504 efficient utilization of storage. For example, a common thread creation technique is to cache  
123505 appropriate thread data structures. That is, rather than releasing system resources, the  
123506 implementation retains these resources and reuses them when the program next asks to create a  
123507 new thread. If this reuse of thread resources is to be possible, there has to be very little unique  
123508 state associated with each thread, because any such state has to be reset when the thread is  
123509 reused.

## Thread Creation Attributes

Attributes objects are provided for threads, mutexes, and condition variables as a mechanism to support probable future standardization in these areas without requiring that the interface itself be changed.

Attributes objects provide clean isolation of the configurable aspects of threads. For example, “stack size” is an important attribute of a thread, but it cannot be expressed portably. When porting a threaded program, stack sizes often need to be adjusted. The use of attributes objects can help by allowing the changes to be isolated in a single place, rather than being spread across every instance of thread creation.

Attributes objects can be used to set up *classes* of threads with similar attributes; for example, “threads with large stacks and high priority” or “threads with minimal stacks”. These classes can be defined in a single place and then referenced wherever threads need to be created. Changes to “class” decisions become straightforward, and detailed analysis of each `pthread_create()` call is not required.

The attributes objects are defined as opaque types as an aid to extensibility. If these objects had been specified as structures, adding new attributes would force recompilation of all multi-threaded programs when the attributes objects are extended; this might not be possible if different program components were supplied by different vendors.

Additionally, opaque attributes objects present opportunities for improving performance. Argument validity can be checked once when attributes are set, rather than each time a thread is created. Implementations will often need to cache kernel objects that are expensive to create. Opaque attributes objects provide an efficient mechanism to detect when cached objects become invalid due to attribute changes.

Because assignment is not necessarily defined on a given opaque type, implementation-defined default values cannot be defined in a portable way. The solution to this problem is to allow attribute objects to be initialized dynamically by attributes object initialization functions, so that default values can be supplied automatically by the implementation.

The following proposal was provided as a suggested alternative to the supplied attributes:

1. Maintain the style of passing a parameter formed by the bitwise-inclusive OR of flags to the initialization routines (`pthread_create()`, `pthread_mutex_init()`, `pthread_cond_init()`). The parameter containing the flags should be an opaque type for extensibility. If no flags are set in the parameter, then the objects are created with default characteristics. An implementation may specify implementation-defined flag values and associated behavior.
2. If further specialization of mutexes and condition variables is necessary, implementations may specify additional procedures that operate on the `pthread_mutex_t` and `pthread_cond_t` objects (instead of on attributes objects).

The difficulties with this solution are:

1. A bitmask is not opaque if bits have to be set into bit-vector attributes objects using explicitly-coded bitwise-inclusive OR operations. If the set of options exceeds an `int`, application programmers need to know the location of each bit. If bits are set or read by encapsulation (that is, `get*()` or `set*()` functions), then the bitmask is merely an implementation of attributes objects as currently defined and should not be exposed to the programmer.
2. Many attributes are not Boolean or very small integral values. For example, scheduling policy may be placed in 3 bits or 4 bits, but priority requires 5 bits or more, thereby taking up at least 8 bits out of a possible 16 bits on machines with 16-bit integers. Because of this,

123557 the bitmask can only reasonably control whether particular attributes are set or not, and it  
123558 cannot serve as the repository of the value itself. The value needs to be specified as a  
123559 function parameter (which is non-extensible), or by setting a structure field (which is non-  
123560 opaque), or by *get\*()* and *set\*()* functions (making the bitmask a redundant addition to  
123561 the attributes objects).

123562 Stack size is defined as an optional attribute because the very notion of a stack is inherently  
123563 machine-dependent. Some implementations may not be able to change the size of the stack, for  
123564 example, and others may not need to because stack pages may be discontinuous and can be  
123565 allocated and released on demand.

123566 The attribute mechanism has been designed in large measure for extensibility. Future extensions  
123567 to the attribute mechanism or to any attributes object defined in POSIX.1-2008 have to be done  
123568 with care so as not to affect binary-compatibility.

123569 Attribute objects, even if allocated by means of dynamic allocation functions such as *malloc()*,  
123570 may have their size fixed at compile time. This means, for example, a *pthread\_create()* in an  
123571 implementation with extensions to the **pthread\_attr\_t** cannot look beyond the area that the  
123572 binary application assumes is valid. This suggests that implementations should maintain a size  
123573 field in the attributes object, as well as possibly version information, if extensions in different  
123574 directions (possibly by different vendors) are to be accommodated.

## 123575 Thread Implementation Models

123576 There are various thread implementation models. At one end of the spectrum is the “library-  
123577 thread model”. In such a model, the threads of a process are not visible to the operating system  
123578 kernel, and the threads are not kernel-scheduled entities. The process is the only kernel-  
123579 scheduled entity. The process is scheduled onto the processor by the kernel according to the  
123580 scheduling attributes of the process. The threads are scheduled onto the single kernel-scheduled  
123581 entity (the process) by the runtime library according to the scheduling attributes of the threads.  
123582 A problem with this model is that it constrains concurrency. Since there is only one kernel-  
123583 scheduled entity (namely, the process), only one thread per process can execute at a time. If the  
123584 thread that is executing blocks on I/O, then the whole process blocks.

123585 At the other end of the spectrum is the “kernel-thread model”. In this model, all threads are  
123586 visible to the operating system kernel. Thus, all threads are kernel-scheduled entities, and all  
123587 threads can concurrently execute. The threads are scheduled onto processors by the kernel  
123588 according to the scheduling attributes of the threads. The drawback to this model is that the  
123589 creation and management of the threads entails operating system calls, as opposed to subroutine  
123590 calls, which makes kernel threads heavier weight than library threads.

123591 Hybrids of these two models are common. A hybrid model offers the speed of library threads  
123592 and the concurrency of kernel threads. In hybrid models, a process has some (relatively small)  
123593 number of kernel scheduled entities associated with it. It also has a potentially much larger  
123594 number of library threads associated with it. Some library threads may be bound to kernel-  
123595 scheduled entities, while the other library threads are multiplexed onto the remaining kernel-  
123596 scheduled entities. There are two levels of thread scheduling:

- 123597 1. The runtime library manages the scheduling of (unbound) library threads onto kernel-  
123598 scheduled entities.
- 123599 2. The kernel manages the scheduling of kernel-scheduled entities onto processors.

123600 For this reason, a hybrid model is referred to as a two-level threads scheduling model. In this  
123601 model, the process can have multiple concurrently executing threads; specifically, it can have as  
123602 many concurrently executing threads as it has kernel-scheduled entities.

## Thread-Specific Data

Many applications require that a certain amount of context be maintained on a per-thread basis across procedure calls. A common example is a multi-threaded library routine that allocates resources from a common pool and maintains an active resource list for each thread. The thread-specific data interface provided to meet these needs may be viewed as a two-dimensional array of values with keys serving as the row index and thread IDs as the column index (although the implementation need not work this way).

- Models

Three possible thread-specific data models were considered:

1. No Explicit Support

A standard thread-specific data interface is not strictly necessary to support applications that require per-thread context. One could, for example, provide a hash function that converted a **pthread\_t** into an integer value that could then be used to index into a global array of per-thread data pointers. This hash function, in conjunction with *pthread\_self()*, would be all the interface required to support a mechanism of this sort. Unfortunately, this technique is cumbersome. It can lead to duplicated code as each set of cooperating modules implements their own per-thread data management schemes. This technique would also require that **pthread\_t** not be an opaque type.

2. Single (**void \***) Pointer

Another technique would be to provide a single word of per-thread storage and a pair of functions to fetch and store the value of this word. The word could then hold a pointer to a block of per-thread memory. The allocation, partitioning, and general use of this memory would be entirely up to the application. Although this method is not as problematic as technique 1, it suffers from interoperability problems. For example, all modules using the per-thread pointer would have to agree on a common usage protocol.

3. Key/Value Mechanism

This method associates an opaque key (for example, stored in a variable of type **pthread\_key\_t**) with each per-thread datum. These keys play the role of identifiers for per-thread data. This technique is the most generic and avoids the problems noted above, albeit at the cost of some complexity.

The primary advantage of the third model is its information hiding properties. Modules using this model are free to create and use their own key(s) independent of all other such usage, whereas the other models require that all modules that use thread-specific context explicitly cooperate with all other such modules. The data-independence provided by the third model is worth the additional interface. Therefore, the third model was chosen.

- Requirements

It is important that it be possible to implement the thread-specific data interface without the use of thread private memory. To do otherwise would increase the weight of each thread, thereby limiting the range of applications for which the threads interfaces provided by POSIX.1-2008 is appropriate.

The values that one binds to the key via *pthread\_setspecific()* may, in fact, be pointers to shared storage locations available to all threads. It is only the key/value bindings that are maintained on a per-thread basis, and these can be kept in any portion of the address space that is reserved for use by the calling thread (for example, on the stack). Thus, no per-



thread MMU state is required to implement the interface. On the other hand, there is nothing in the interface specification to preclude the use of a per-thread MMU state if it is available (for example, the key values returned by `pthread_key_create()` could be thread private memory addresses).

- Standardization Issues

Thread-specific data is a requirement for a usable thread interface. The binding described in this section provides a portable thread-specific data mechanism for languages that do not directly support a thread-specific storage class. A binding to POSIX.1-2008 for a language that does include such a storage class need not provide this specific interface.

If a language were to include the notion of thread-specific storage, it would be desirable (but *not* required) to provide an implementation of the pthreads thread-specific data interface based on the language feature. For example, assume that a compiler for a C-like language supports a *private* storage class that provides thread-specific storage. Something similar to the following macros might be used to effect a compatible implementation:

```
#define pthread_key_t          private void *
#define pthread_key_create(key) /* no-op */
#define pthread_setspecific(key,value) (key)=(value)
#define pthread_getspecific(key)      (key)
```

**Note:** For the sake of clarity, this example ignores destructor functions. A correct implementation would have to support them.

## Barriers

- Background

Barriers are typically used in parallel DO/FOR loops to ensure that all threads have reached a particular stage in a parallel computation before allowing any to proceed to the next stage. Highly efficient implementation is possible on machines which support a “Fetch and Add” operation as described in the referenced Almasi and Gottlieb (1989).

The use of return value `PTHREAD_BARRIER_SERIAL_THREAD` is shown in the following example:

```
if ( (status=pthread_barrier_wait(&barrier)) ==
    PTHREAD_BARRIER_SERIAL_THREAD) {
    ...serial section
}
else if (status != 0) {
    ...error processing
}
status=pthread_barrier_wait(&barrier);
...
```

This behavior allows a serial section of code to be executed by one thread as soon as all threads reach the first barrier. The second barrier prevents the other threads from proceeding until the serial section being executed by the one thread has completed.

Although barriers can be implemented with mutexes and condition variables, the referenced Almasi and Gottlieb (1989) provides ample illustration that such implementations are significantly less efficient than is possible. While the relative efficiency of barriers may well vary by implementation, it is important that they be recognized in the POSIX.1-2008 to facilitate applications portability while providing the necessary freedom to implementors.

- Lack of Timeout Feature

Alternate versions of most blocking routines have been provided to support watchdog timeouts. No alternate interface of this sort has been provided for barrier waits for the following reasons:

- Multiple threads may use different timeout values, some of which may be indefinite. It is not clear which threads should break through the barrier with a timeout error if and when these timeouts expire.
- The barrier may become unusable once a thread breaks out of a `pthread_barrier_wait()` with a timeout error. There is, in general, no way to guarantee the consistency of a barrier's internal data structures once a thread has timed out of a `pthread_barrier_wait()`. Even the inclusion of a special barrier reinitialization function would not help much since it is not clear how this function would affect the behavior of threads that reach the barrier between the original timeout and the call to the reinitialization function.

## Spin Locks

- Background

Spin locks represent an extremely low-level synchronization mechanism suitable primarily for use on shared memory multi-processors. It is typically an atomically modified Boolean value that is set to one when the lock is held and to zero when the lock is freed.

When a caller requests a spin lock that is already held, it typically spins in a loop testing whether the lock has become available. Such spinning wastes processor cycles so the lock should only be held for short durations and not across sleep/block operations. Callers should unlock spin locks before calling sleep operations.

Spin locks are available on a variety of systems. The functions included in POSIX.1-2008 are an attempt to standardize that existing practice.

- Lack of Timeout Feature

Alternate versions of most blocking routines have been provided to support watchdog timeouts. No alternate interface of this sort has been provided for spin locks for the following reasons:

- It is impossible to determine appropriate timeout intervals for spin locks in a portable manner. The amount of time one can expect to spend spin-waiting is inversely proportional to the degree of parallelism provided by the system.  
It can vary from a few cycles when each competing thread is running on its own processor, to an indefinite amount of time when all threads are multiplexed on a single processor (which is why spin locking is not advisable on uniprocessors).
- When used properly, the amount of time the calling thread spends waiting on a spin lock should be considerably less than the time required to set up a corresponding watchdog timer. Since the primary purpose of spin locks is to provide a low-overhead synchronization mechanism for multi-processors, the overhead of a timeout mechanism was deemed unacceptable.

It was also suggested that an additional `count` argument be provided (on the `pthread_spin_lock()` call) in lieu of a true timeout so that a spin lock call could fail gracefully if it was unable to apply the lock after `count` attempts. This idea was rejected because it is not existing practice. Furthermore, the same effect can be obtained with `pthread_spin_trylock()`, as illustrated below:

```

123740     int n = MAX_SPIN;
123741     while ( --n >= 0 )
123742     {
123743         if ( !pthread_spin_try_lock(...) )
123744             break;
123745     }
123746     if ( n >= 0 )
123747     {
123748         /* Successfully acquired the lock */
123749     }
123750     else
123751     {
123752         /* Unable to acquire the lock */
123753     }

```

- *process-shared* Attribute

The initialization functions associated with most POSIX synchronization objects (for example, mutexes, barriers, and read-write locks) take an attributes object with a *process-shared* attribute that specifies whether or not the object is to be shared across processes. In the draft corresponding to the first balloting round, two separate initialization functions are provided for spin locks, however: one for spin locks that were to be shared across processes (*spin\_init()*), and one for locks that were only used by multiple threads within a single process (*pthread\_spin\_init()*). This was done so as to keep the overhead associated with spin waiting to an absolute minimum. However, the balloting group requested that, since the overhead associated to a bit check was small, spin locks should be consistent with the rest of the synchronization primitives, and thus the *process-shared* attribute was introduced for spin locks.

- Spin Locks *versus* Mutexes

It has been suggested that mutexes are an adequate synchronization mechanism and spin locks are not necessary. Locking mechanisms typically must trade off the processor resources consumed while setting up to block the thread and the processor resources consumed by the thread while it is blocked. Spin locks require very little resources to set up the blocking of a thread. Existing practice is to simply loop, repeating the atomic locking operation until the lock is available. While the resources consumed to set up blocking of the thread are low, the thread continues to consume processor resources while it is waiting.

On the other hand, mutexes may be implemented such that the processor resources consumed to block the thread are large relative to a spin lock. After detecting that the mutex lock is not available, the thread must alter its scheduling state, add itself to a set of waiting threads, and, when the lock becomes available again, undo all of this before taking over ownership of the mutex. However, while a thread is blocked by a mutex, no processor resources are consumed.

Therefore, spin locks and mutexes may be implemented to have different characteristics. Spin locks may have lower overall overhead for very short-term blocking, and mutexes may have lower overall overhead when a thread will be blocked for longer periods of time. The presence of both interfaces allows implementations with these two different characteristics, both of which may be useful to a particular application.

It has also been suggested that applications can build their own spin locks from the *pthread\_mutex\_trylock()* function:



123788           while (pthread\_mutex\_trylock(&mutex));

123789           The apparent simplicity of this construct is somewhat deceiving, however. While the actual  
123790           wait is quite efficient, various guarantees on the integrity of mutex objects (for example,  
123791           priority inheritance rules) may add overhead to the successful path of the trylock  
123792           operation that is not required of spin locks. One could, of course, add an attribute to the  
123793           mutex to bypass such overhead, but the very act of finding and testing this attribute  
123794           represents more overhead than is found in the typical spin lock.

123795           The need to hold spin lock overhead to an absolute minimum also makes it impossible to  
123796           provide guarantees against starvation similar to those provided for mutexes or read-write  
123797           locks. The overhead required to implement such guarantees (for example, disabling  
123798           preemption before spinning) may well exceed the overhead of the spin wait itself by many  
123799           orders of magnitude. If a “safe” spin wait seems desirable, it can always be provided  
123800           (albeit at some performance cost) via appropriate mutex attributes.

## 123801           **Robust Mutexes**

123802           Robust mutexes are intended to protect applications that use mutexes to protect data shared  
123803           between different processes. If a process is terminated by a signal while a thread is holding a  
123804           mutex, there is no chance for the process to clean up after it. Waiters for the locked mutex might  
123805           wait indefinitely.

123806           With robust mutexes the problem can be solved: whenever a fatal signal terminates a process,  
123807           current or future waiters of the mutex are notified about this fact. The locking function provides  
123808           notification of this condition through the error condition [EOWNERDEAD]. A thread then has  
123809           the chance to clean up the state protected by the mutex and mark the state as consistent again by  
123810           a call to *pthread\_mutex\_consistent()*.

123811           Pre-existing implementations have used the semantics of robust mutexes for a variety of  
123812           situations, some of them not defined in the standard. Where a normally terminated process (i.e.,  
123813           when one thread calls *exit()*) causes notification of other waiters of robust mutexes if the mutex  
123814           is locked by any thread in the process. This behavior is defined in the standard and makes sense  
123815           because no thread other than the thread calling *exit()* has the chance to clean up its data.

123816           If a thread is terminated by cancellation or if it calls *pthread\_exit()*, the situation is different. In  
123817           both these situations the thread has the chance to clean up after itself by registering appropriate  
123818           cleanup handlers. There is no real reason to demand that other waiters for a robust mutex the  
123819           terminating thread owns are notified. The committee felt that this is actively encouraging bad  
123820           practice because programmers are tempted to rely on the robust mutex semantics instead of  
123821           correctly cleaning up after themselves.

123822           Therefore, the standard does not require notification of other waiters at the time a thread is  
123823           terminated while the process continues to run. The mutex is still recognized as being locked by  
123824           the process (with the thread gone it makes no sense to refer to the thread owning the mutex).  
123825           Therefore, a terminating process will cause notifications about the dead owner to be sent to all  
123826           waiters. This delay in the notification is not required, but programmers cannot rely on prompt  
123827           notification after a thread is terminated.

123828           For the same reason is it not required that an implementation supports robust mutexes that are  
123829           not shared between processes. If a robust mutex is used only within one process, all the cleanup  
123830           can be performed by the threads themselves by registering appropriate cleanup handlers. Fatal  
123831           signals are of no importance in this case because after the signal is delivered there is no thread  
123832           remaining to use the mutex.

123833           Some implementations might choose to support intra-process robust mutexes and they might  
123834           also send notification of a dead owner right after the previous owner died. But applications

123835 must not rely on this. Applications should only use robust mutexes for the purpose of handling  
123836 fatal signals in situations where inter-process mutexes are in use.

### 123837 **Supported Threads Functions**

123838 On POSIX-conforming systems, the following symbolic constants are always conforming:

123839 `_POSIX_READER_WRITER_LOCKS`  
123840 `_POSIX_THREADS`

123841 Therefore, the following threads functions are always supported:

123842 <code>pthread_atfork()</code>	<code>pthread_mutex_destroy()</code>
123843 <code>pthread_attr_destroy()</code>	<code>pthread_mutex_init()</code>
123844 <code>pthread_attr_getdetachstate()</code>	<code>pthread_mutex_lock()</code>
123845 <code>pthread_attr_getguardsize()</code>	<code>pthread_mutex_trylock()</code>
123846 <code>pthread_attr_getschedparam()</code>	<code>pthread_mutex_unlock()</code>
123847 <code>pthread_attr_init()</code>	<code>pthread_mutexattr_destroy()</code>
123848 <code>pthread_attr_setdetachstate()</code>	<code>pthread_mutexattr_getpshared()</code>
123849 <code>pthread_attr_setguardsize()</code>	<code>pthread_mutexattr_gettype()</code>
123850 <code>pthread_attr_setschedparam()</code>	<code>pthread_mutexattr_init()</code>
123851 <code>pthread_cancel()</code>	<code>pthread_mutexattr_setpshared()</code>
123852 <code>pthread_cleanup_pop()</code>	<code>pthread_mutexattr_settype()</code>
123853 <code>pthread_cleanup_push()</code>	<code>pthread_once()</code>
123854 <code>pthread_cond_broadcast()</code>	<code>pthread_rwlock_destroy()</code>
123855 <code>pthread_cond_destroy()</code>	<code>pthread_rwlock_init()</code>
123856 <code>pthread_cond_init()</code>	<code>pthread_rwlock_rdlock()</code>
123857 <code>pthread_cond_signal()</code>	<code>pthread_rwlock_tryrdlock()</code>
123858 <code>pthread_cond_timedwait()</code>	<code>pthread_rwlock_trywrlock()</code>
123859 <code>pthread_cond_wait()</code>	<code>pthread_rwlock_unlock()</code>
123860 <code>pthread_condattr_destroy()</code>	<code>pthread_rwlock_wrlock()</code>
123861 <code>pthread_condattr_getpshared()</code>	<code>pthread_rwlockattr_destroy()</code>
123862 <code>pthread_condattr_init()</code>	<code>pthread_rwlockattr_getpshared()</code>
123863 <code>pthread_condattr_setpshared()</code>	<code>pthread_rwlockattr_init()</code>
123864 <code>pthread_create()</code>	<code>pthread_rwlockattr_setpshared()</code>
123865 <code>pthread_detach()</code>	<code>pthread_self()</code>
123866 <code>pthread_equal()</code>	<code>pthread_setcancelstate()</code>
123867 <code>pthread_exit()</code>	<code>pthread_setcanceltype()</code>
123868 <code>pthread_getconcurrency()</code>	<code>pthread_setconcurrency()</code>
123869 <code>pthread_getspecific()</code>	<code>pthread_setspecific()</code>
123870 <code>pthread_join()</code>	<code>pthread_sigmask()</code>
123871 <code>pthread_key_create()</code>	<code>pthread_testcancel()</code>
123872 <code>pthread_key_delete()</code>	<code>sigwait()</code>
123873 <code>pthread_kill()</code>	

123874 On POSIX-conforming systems, the symbolic constant `_POSIX_THREAD_SAFE_FUNCTIONS` is  
 123875 always defined. Therefore, the following functions are always supported:

123876	<code>asctime_r()</code>	<code>getpwuid_r()</code>
123877	<code>ctime_r()</code>	<code>gmtime_r()</code>
123878	<code>flockfile()</code>	<code>localtime_r()</code>
123879	<code>ftrylockfile()</code>	<code>putc_unlocked()</code>
123880	<code>funlockfile()</code>	<code>putchar_unlocked()</code>
123881	<code>getc_unlocked()</code>	<code>rand_r()</code>
123882	<code>getchar_unlocked()</code>	<code>readdir_r()</code>
123883	<code>getgrgid_r()</code>	<code>strerror_r()</code>
123884	<code>getgrnam_r()</code>	<code>strtok_r()</code>
123885	<code>getpwnam_r()</code>	

## 123886 Threads Extensions

123887 The following extensions to the IEEE P1003.1c draft standard are now supported in  
 123888 POSIX.1-2008 as part of the alignment with the Single UNIX Specification:

- 123889 • Extended mutex attribute types
- 123890 • Read-write locks and attributes (also introduced by the IEEE Std 1003.1j-2000 amendment)
- 123891 • Thread concurrency level
- 123892 • Thread stack guard size
- 123893 • Parallel I/O
- 123894 • Robust mutexes

123895 These extensions carefully follow the threads programming model specified in POSIX.1c. As  
 123896 with POSIX.1c, all the new functions return zero if successful; otherwise, an error number is  
 123897 returned to indicate the error.

123898 The concept of attribute objects was introduced in POSIX.1c to allow implementations to extend  
 123899 POSIX.1-2008 without changing the existing interfaces. Attribute objects were defined for  
 123900 threads, mutexes, and condition variables. Attributes objects are defined as implementation-  
 123901 defined opaque types to aid extensibility, and functions are defined to allow attributes to be set  
 123902 or retrieved. This model has been followed when adding the new type attribute of  
 123903 **`pthread_mutexattr_t`** or the new read-write lock attributes object **`pthread_rwlockattr_t`**.

- 123904 • Extended Mutex Attributes

123905 POSIX.1c defines a mutex attributes object as an implementation-defined opaque object of  
 123906 type **`pthread_mutexattr_t`**, and specifies a number of attributes which this object must  
 123907 have and a number of functions which manipulate these attributes. These attributes  
 123908 include *detachstate*, *inheritsched*, *schedparm*, *schedpolicy*, *contentionscope*, *stackaddr*, and  
 123909 *stacksize*.

123910 The System Interfaces volume of POSIX.1-2008 specifies another mutex attribute called  
 123911 *type*. The *type* attribute allows applications to specify the behavior of mutex locking  
 123912 operations in situations where POSIX.1c behavior is undefined. The OSF DCE threads  
 123913 implementation, based on Draft 4 of POSIX.1c, specified a similar attribute. Note that the  
 123914 names of the attributes have changed somewhat from the OSF DCE threads  
 123915 implementation.

123916 The System Interfaces volume of POSIX.1-2008 also extends the specification of the  
 123917 following POSIX.1c functions which manipulate mutexes:

123918 `pthread_mutex_lock()`  
123919 `pthread_mutex_trylock()`  
123920 `pthread_mutex_unlock()`

123921 to take account of the new mutex attribute type and to specify behavior which was  
123922 declared as undefined in POSIX.1c. How a calling thread acquires or releases a mutex now  
123923 depends upon the mutex *type* attribute.

123924 The *type* attribute can have the following values:

123925 PTHREAD\_MUTEX\_NORMAL

123926 Basic mutex with no specific error checking built in. Does not report a deadlock error.

123927 PTHREAD\_MUTEX\_RECURSIVE

123928 Allows any thread to recursively lock a mutex. The mutex must be unlocked an equal  
123929 number of times to release the mutex.

123930 PTHREAD\_MUTEX\_ERRORCHECK

123931 Detects and reports simple usage errors; that is, an attempt to unlock a mutex that is  
123932 not locked by the calling thread or that is not locked at all, or an attempt to relock a  
123933 mutex the thread already owns.

123934 PTHREAD\_MUTEX\_DEFAULT

123935 The default mutex type. May be mapped to any of the above mutex types or may be  
123936 an implementation-defined type.

123937 *Normal* mutexes do not detect deadlock conditions; for example, a thread will hang if it  
123938 tries to relock a normal mutex that it already owns. Attempting to unlock a mutex locked  
123939 by another thread, or unlocking an unlocked mutex, results in undefined behavior. Normal  
123940 mutexes will usually be the fastest type of mutex available on a platform but provide the  
123941 least error checking.

123942 *Recursive* mutexes are useful for converting old code where it is difficult to establish clear  
123943 boundaries of synchronization. A thread can relock a recursive mutex without first  
123944 unlocking it. The relocking deadlock which can occur with normal mutexes cannot occur  
123945 with this type of mutex. However, multiple locks of a recursive mutex require the same  
123946 number of unlocks to release the mutex before another thread can acquire the mutex.  
123947 Furthermore, this type of mutex maintains the concept of an owner. Thus, a thread  
123948 attempting to unlock a recursive mutex which another thread has locked returns with an  
123949 error. A thread attempting to unlock a recursive mutex that is not locked returns with an  
123950 error. Never use a recursive mutex with condition variables because the implicit unlock  
123951 performed by `pthread_cond_wait()` or `pthread_cond_timedwait()` will not actually release the  
123952 mutex if it had been locked multiple times.

123953 *Errorcheck* mutexes provide error checking and are useful primarily as a debugging aid. A  
123954 thread attempting to relock an errorcheck mutex without first unlocking it returns with an  
123955 error. Again, this type of mutex maintains the concept of an owner. Thus, a thread  
123956 attempting to unlock an errorcheck mutex which another thread has locked returns with  
123957 an error. A thread attempting to unlock an errorcheck mutex that is not locked also returns  
123958 with an error. It should be noted that errorcheck mutexes will almost always be much  
123959 slower than normal mutexes due to the extra state checks performed.

123960 The default mutex type provides implementation-defined error checking. The default  
123961 mutex may be mapped to one of the other defined types or may be something entirely  
123962 different. This enables each vendor to provide the mutex semantics which the vendor feels  
123963 will be most useful to their target users. Most vendors will probably choose to make  
123964 normal mutexes the default so as to give applications the benefit of the fastest type of

123965 mutexes available on their platform. Check your implementation's documentation.

123966 An application developer can use any of the mutex types almost interchangeably as long  
 123967 as the application does not depend upon the implementation detecting (or failing to  
 123968 detect) any particular errors. Note that a recursive mutex can be used with condition  
 123969 variable waits as long as the application never recursively locks the mutex.

123970 Two functions are provided for manipulating the *type* attribute of a mutex attributes object.  
 123971 This attribute is set or returned in the *type* parameter of these functions. The  
 123972 *pthread\_mutexattr\_settype()* function is used to set a specific type value while  
 123973 *pthread\_mutexattr\_gettype()* is used to return the type of the mutex. Setting the *type*  
 123974 attribute of a mutex attributes object affects only mutexes initialized using that mutex  
 123975 attributes object. Changing the *type* attribute does not affect mutexes previously initialized  
 123976 using that mutex attributes object.

123977 • Read-Write Locks and Attributes

123978 The read-write locks introduced have been harmonized with those in IEEE Std  
 123979 1003.1j-2000; see also [Section B.2.9.6](#) (on page 3648).

123980 Read-write locks (also known as reader-writer locks) allow a thread to exclusively lock  
 123981 some shared data while updating that data, or allow any number of threads to have  
 123982 simultaneous read-only access to the data.

123983 Unlike a mutex, a read-write lock distinguishes between reading data and writing data. A  
 123984 mutex excludes all other threads. A read-write lock allows other threads access to the data,  
 123985 providing no thread is modifying the data. Thus, a read-write lock is less primitive than  
 123986 either a mutex-condition variable pair or a semaphore.

123987 Application developers should consider using a read-write lock rather than a mutex to  
 123988 protect data that is frequently referenced but seldom modified. Most threads (readers) will  
 123989 be able to read the data without waiting and will only have to block when some other  
 123990 thread (a writer) is in the process of modifying the data. Conversely a thread that wants to  
 123991 change the data is forced to wait until there are no readers. This type of lock is often used  
 123992 to facilitate parallel access to data on multi-processor platforms or to avoid context  
 123993 switches on single processor platforms where multiple threads access the same data.

123994 If a read-write lock becomes unlocked and there are multiple threads waiting to acquire  
 123995 the write lock, the implementation's scheduling policy determines which thread acquires  
 123996 the read-write lock for writing. If there are multiple threads blocked on a read-write lock  
 123997 for both read locks and write locks, it is unspecified whether the readers or a writer  
 123998 acquire the lock first. However, for performance reasons, implementations often favor  
 123999 writers over readers to avoid potential writer starvation.

124000 A read-write lock object is an implementation-defined opaque object of type  
 124001 **pthread\_rwlock\_t** as defined in **<pthread.h>**. There are two different sorts of locks  
 124002 associated with a read-write lock: a read lock and a write lock.

124003 The *pthread\_rwlockattr\_init()* function initializes a read-write lock attributes object with the  
 124004 default value for all the attributes defined in the implementation. After a read-write lock  
 124005 attributes object has been used to initialize one or more read-write locks, changes to the  
 124006 read-write lock attributes object, including destruction, do not affect previously initialized  
 124007 read-write locks.

124008 Implementations must provide at least the read-write lock attribute *process-shared*. This  
 124009 attribute can have the following values:



124010 PTHREAD\_PROCESS\_SHARED  
124011 Any thread of any process that has access to the memory where the read-write lock  
124012 resides can manipulate the read-write lock.

124013 PTHREAD\_PROCESS\_PRIVATE  
124014 Only threads created within the same process as the thread that initialized the read-  
124015 write lock can manipulate the read-write lock. This is the default value.

124016 The *pthread\_rwlockattr\_setpshared()* function is used to set the *process-shared* attribute of an  
124017 initialized read-write lock attributes object while the function  
124018 *pthread\_rwlockattr\_getpshared()* obtains the current value of the *process-shared* attribute.

124019 A read-write lock attributes object is destroyed using the *pthread\_rwlockattr\_destroy()*  
124020 function. The effect of subsequent use of the read-write lock attributes object is undefined.

124021 A thread creates a read-write lock using the *pthread\_rwlock\_init()* function. The attributes  
124022 of the read-write lock can be specified by the application developer; otherwise, the default  
124023 implementation-defined read-write lock attributes are used if the pointer to the read-write  
124024 lock attributes object is NULL. In cases where the default attributes are appropriate, the  
124025 PTHREAD\_RWLOCK\_INITIALIZER macro can be used to initialize read-write locks.

124026 A thread which wants to apply a read lock to the read-write lock can use either  
124027 *pthread\_rwlock\_rdlock()* or *pthread\_rwlock\_tryrdlock()*. If *pthread\_rwlock\_rdlock()* is used, the  
124028 thread acquires a read lock if a writer does not hold the write lock and there are no writers  
124029 blocked on the write lock. If a read lock is not acquired, the calling thread blocks until it  
124030 can acquire a lock. However, if *pthread\_rwlock\_tryrdlock()* is used, the function returns  
124031 immediately with the error [EBUSY] if any thread holds a write lock or there are blocked  
124032 writers waiting for the write lock.

124033 A thread which wants to apply a write lock to the read-write lock can use either of two  
124034 functions: *pthread\_rwlock\_wrlock()* or *pthread\_rwlock\_trywrlock()*. If *pthread\_rwlock\_wrlock()*  
124035 is used, the thread acquires the write lock if no other reader or writer threads hold the  
124036 read-write lock. If the write lock is not acquired, the thread blocks until it can acquire the  
124037 write lock. However, if *pthread\_rwlock\_trywrlock()* is used, the function returns  
124038 immediately with the error [EBUSY] if any thread is holding either a read or a write lock.

124039 The *pthread\_rwlock\_unlock()* function is used to unlock a read-write lock object held by the  
124040 calling thread. Results are undefined if the read-write lock is not held by the calling thread.  
124041 If there are other read locks currently held on the read-write lock object, the read-write  
124042 lock object remains in the read locked state but without the current thread as one of its  
124043 owners. If this function releases the last read lock for this read-write lock object, the read-  
124044 write lock object is put in the unlocked read state. If this function is called to release a write  
124045 lock for this read-write lock object, the read-write lock object is put in the unlocked state.

124046 • Thread Concurrency Level

124047 On threads implementations that multiplex user threads onto a smaller set of kernel  
124048 execution entities, the system attempts to create a reasonable number of kernel execution  
124049 entities for the application upon application startup.

124050 On some implementations, these kernel entities are retained by user threads that block in  
124051 the kernel. Other implementations do not *timeslice* user threads so that multiple compute-  
124052 bound user threads can share a kernel thread. On such implementations, some  
124053 applications may use up all the available kernel execution entities before their user-space  
124054 threads are used up. The process may be left with user threads capable of doing work for  
124055 the application but with no way to schedule them.

124056 The *pthread\_setconcurrency()* function enables an application to request more kernel

entities; that is, specify a desired concurrency level. However, this function merely provides a hint to the implementation. The implementation is free to ignore this request or to provide some other number of kernel entities. If an implementation does not multiplex user threads onto a smaller number of kernel execution entities, the *pthread\_setconcurrency()* function has no effect.

The *pthread\_setconcurrency()* function may also have an effect on implementations where the kernel mode and user mode schedulers cooperate to ensure that ready user threads are not prevented from running by other threads blocked in the kernel.

The *pthread\_getconcurrency()* function always returns the value set by a previous call to *pthread\_setconcurrency()*. However, if *pthread\_setconcurrency()* was not previously called, this function returns zero to indicate that the threads implementation is maintaining the concurrency level.

- Thread Stack Guard Size

DCE threads introduced the concept of a “thread stack guard size”. Most thread implementations add a region of protected memory to a thread’s stack, commonly known as a “guard region”, as a safety measure to prevent stack pointer overflow in one thread from corrupting the contents of another thread’s stack. The default size of the guard regions attribute is {PAGESIZE} bytes and is implementation-defined.

Some application developers may wish to change the stack guard size. When an application creates a large number of threads, the extra page allocated for each stack may strain system resources. In addition to the extra page of memory, the kernel’s memory manager has to keep track of the different protections on adjoining pages. When this is a problem, the application developer may request a guard size of 0 bytes to conserve system resources by eliminating stack overflow protection.

Conversely an application that allocates large data structures such as arrays on the stack may wish to increase the default guard size in order to detect stack overflow. If a thread allocates two pages for a data array, a single guard page provides little protection against thread stack overflows since the thread can corrupt adjoining memory beyond the guard page.

The System Interfaces volume of POSIX.1-2008 defines a new attribute of a thread attributes object; that is, the *guardsize* attribute which allows applications to specify the size of the guard region of a thread’s stack.

Two functions are provided for manipulating a thread’s stack guard size. The *pthread\_attr\_setguardsize()* function sets the thread *guardsize* attribute, and the *pthread\_attr\_getguardsize()* function retrieves the current value.

An implementation may round up the requested guard size to a multiple of the configurable system variable {PAGESIZE}. In this case, *pthread\_attr\_getguardsize()* returns the guard size specified by the previous *pthread\_attr\_setguardsize()* function call and not the rounded up value.

If an application is managing its own thread stacks using the *stackaddr* attribute, the *guardsize* attribute is ignored and no stack overflow protection is provided. In this case, it is the responsibility of the application to manage stack overflow along with stack allocation.

- Parallel I/O

Suppose two or more threads independently issue read requests on the same file. To read specific data from a file, a thread must first call *lseek()* to seek to the proper offset in the file, and then call *read()* to retrieve the required data. If more than one thread does this at the same time, the first thread may complete its seek call, but before it gets a chance to

124104 issue its read call a second thread may complete its seek call, resulting in the first thread  
124105 accessing incorrect data when it issues its read call. One workaround is to lock the file  
124106 descriptor while seeking and reading or writing, but this reduces parallelism and adds  
124107 overhead.

124108 Instead, the System Interfaces volume of POSIX.1-2008 provides two functions to make  
124109 seek/read and seek/write operations atomic. The file descriptor's current offset is  
124110 unchanged, thus allowing multiple read and write operations to proceed in parallel. This  
124111 improves the I/O performance of threaded applications. The *pread()* function is used to do  
124112 an atomic read of data from a file into a buffer. Conversely, the *pwrite()* function does an  
124113 atomic write of data from a buffer to a file.

#### 124114 B.2.9.1 Thread-Safety

124115 All functions required by POSIX.1-2008 need to be thread-safe. Implementations have to  
124116 provide internal synchronization when necessary in order to achieve this goal. In certain cases—  
124117 for example, most floating-point implementations—context switch code may have to manage  
124118 the writable shared state.

124119 While a read from a pipe of {PIPE\_BUF}\*2 bytes may not generate a single atomic and thread-  
124120 safe stream of bytes, it should generate “several” (individually atomic) thread-safe streams of  
124121 bytes. Similarly, while reading from a terminal device may not generate a single atomic and  
124122 thread-safe stream of bytes, it should generate some finite number of (individually atomic) and  
124123 thread-safe streams of bytes. That is, concurrent calls to read for a pipe, FIFO, or terminal device  
124124 are not allowed to result in corrupting the stream of bytes or other internal data. However,  
124125 *read()*, in these cases, is not required to return a single contiguous and atomic stream of bytes.

124126 It is not required that all functions provided by POSIX.1-2008 be either async-cancel-safe or  
124127 async-signal-safe.

124128 As it turns out, some functions are inherently not thread-safe; that is, their interface  
124129 specifications preclude async-signal-safety. For example, some functions (such as *asctime()*)  
124130 return a pointer to a result stored in memory space allocated by the function on a per-process  
124131 basis. Such a function is not thread-safe, because its result can be overwritten by successive  
124132 invocations. Other functions, while not inherently non-thread-safe, may be implemented in  
124133 ways that lead to them not being thread-safe. For example, some functions (such as *rand()*) store  
124134 state information (such as a seed value, which survives multiple function invocations) in  
124135 memory space allocated by the function on a per-process basis. The implementation of such a  
124136 function is not thread-safe if the implementation fails to synchronize invocations of the function  
124137 and thus fails to protect the state information. The problem is that when the state information is  
124138 not protected, concurrent invocations can interfere with one another (for example, applications  
124139 using *rand()* may see the same seed value).

#### 124140 Thread-Safety and Locking of Existing Functions

124141 Originally, POSIX.1 was not designed to work in a multi-threaded environment, and some  
124142 implementations of some existing functions will not work properly when executed concurrently.  
124143 To provide routines that will work correctly in an environment with threads (“thread-safe”), two  
124144 problems need to be solved:

- 124145 1. Routines that maintain or return pointers to static areas internal to the routine (which  
124146 may now be shared) need to be modified. The routines *ttyname()* and *localtime()* are  
124147 examples.
- 124148 2. Routines that access data space shared by more than one thread need to be modified. The  
124149 *malloc()* function and the *stdio* family routines are examples.



There are a variety of constraints on these changes. The first is compatibility with the existing versions of these functions—non-thread-safe functions will continue to be in use for some time, as the original interfaces are used by existing code. Another is that the new thread-safe versions of these functions represent as small a change as possible over the familiar interfaces provided by the existing non-thread-safe versions. The new interfaces should be independent of any particular threads implementation. In particular, they should be thread-safe without depending on explicit thread-specific memory. Finally, there should be minimal performance penalty due to the changes made to the functions.

It is intended that the list of functions from POSIX.1 that cannot be made thread-safe and for which corrected versions are provided be complete.

#### *Thread-Safety and Locking Solutions*

Many of the POSIX.1 functions were thread-safe and did not change at all. However, some functions (for example, the math functions typically found in **libm**) are not thread-safe because of writable shared global state. For instance, in IEEE Std 754-1985 floating-point implementations, the computation modes and flags are global and shared.

Some functions are not thread-safe because a particular implementation is not reentrant, typically because of a non-essential use of static storage. These require only a new implementation.

Thread-safe libraries are useful in a wide range of parallel (and asynchronous) programming environments, not just within pthreads. In order to be used outside the context of pthreads, however, such libraries still have to use some synchronization method. These could either be independent of the pthread synchronization operations, or they could be a subset of the pthread interfaces. Either method results in thread-safe library implementations that can be used without the rest of pthreads.

Some functions, such as the *stdio* family interface and dynamic memory allocation functions such as *malloc()*, are inter-dependent routines that share resources (for example, buffers) across related calls. These require synchronization to work correctly, but they do not require any change to their external (user-visible) interfaces.

In some cases, such as *getc()* and *putc()*, adding synchronization is likely to create an unacceptable performance impact. In this case, slower thread-safe synchronized functions are to be provided, but the original, faster (but unsafe) functions (which may be implemented as macros) are retained under new names. Some additional special-purpose synchronization facilities are necessary for these macros to be usable in multi-threaded programs. This also requires changes in **<stdio.h>**.

The other common reason that functions are unsafe is that they return a pointer to static storage, making the functions non-thread-safe. This has to be changed, and there are three natural choices:

1. Return a pointer to thread-specific storage

This could incur a severe performance penalty on those architectures with a costly implementation of the thread-specific data interface.

A variation on this technique is to use *malloc()* to allocate storage for the function output and return a pointer to this storage. This technique may also have an undesirable performance impact, however, and a simplistic implementation requires that the user program explicitly free the storage object when it is no longer needed. This technique is used by some existing POSIX.1 functions. With careful implementation for infrequently used functions, there may be little or no performance or storage penalty, and the maintenance of already-standardized interfaces is a significant benefit.

2. Return the actual value computed by the function

This technique can only be used with functions that return pointers to structures—routines that return character strings would have to wrap their output in an enclosing structure in order to return the output on the stack. There is also a negative performance impact inherent in this solution in that the output value has to be copied twice before it can be used by the calling function: once from the called routine's local buffers to the top of the stack, then from the top of the stack to the assignment target. Finally, many older compilers cannot support this technique due to a historical tendency to use internal static buffers to deliver the results of structure-valued functions.

3. Have the caller pass the address of a buffer to contain the computed value

The only disadvantage of this approach is that extra arguments have to be provided by the calling program. It represents the most efficient solution to the problem, however, and, unlike the *malloc()* technique, it is semantically clear.

There are some routines (often groups of related routines) whose interfaces are inherently non-thread-safe because they communicate across multiple function invocations by means of static memory locations. The solution is to redesign the calls so that they are thread-safe, typically by passing the needed data as extra parameters. Unfortunately, this may require major changes to the interface as well.

A floating-point implementation using IEEE Std 754-1985 is a case in point. A less problematic example is the *rand48* family of pseudo-random number generators. The functions *getgrgid()*, *getgrnam()*, *getpwnam()*, and *getpwuid()* are another such case.

The problems with *errno* are discussed in [Alternative Solutions for Per-Thread \*errno\*](#) (on page 3564).

Some functions can be thread-safe or not, depending on their arguments. These include the *tmpnam()* and *ctermid()* functions. These functions have pointers to character strings as arguments. If the pointers are not NULL, the functions store their results in the character string; however, if the pointers are NULL, the functions store their results in an area that may be static and thus subject to overwriting by successive calls. These should only be called by multi-thread applications when their arguments are non-NULL.

*Asynchronous Safety and Thread-Safety*

A floating-point implementation has many modes that effect rounding and other aspects of computation. Functions in some math library implementations may change the computation modes for the duration of a function call. If such a function call is interrupted by a signal or cancellation, the floating-point state is not required to be protected.

There is a significant cost to make floating-point operations async-cancel-safe or async-signal-safe; accordingly, neither form of async safety is required.

*Functions Returning Pointers to Static Storage*

For those functions that are not thread-safe because they return values in fixed size statically allocated structures, alternate “\_r” forms are provided that pass a pointer to an explicit result structure. Those that return pointers into library-allocated buffers have forms provided with explicit buffer and length parameters.

For functions that return pointers to library-allocated buffers, it makes sense to provide “\_r” versions that allow the application control over allocation of the storage in which results are returned. This allows the state used by these functions to be managed on an application-specific basis, supporting per-thread, per-process, or other application-specific sharing relationships.

Early proposals had provided “\_r” versions for functions that returned pointers to variable-size

124243 buffers without providing a means for determining the required buffer size. This would have  
 124244 made using such functions exceedingly clumsy, potentially requiring iteratively calling them  
 124245 with increasingly larger guesses for the amount of storage required. Hence, *sysconf()* variables  
 124246 have been provided for such functions that return the maximum required buffer size.

124247 Thus, the rule that has been followed by POSIX.1-2008 when adapting single-threaded non-  
 124248 thread-safe functions is as follows: all functions returning pointers to library-allocated storage  
 124249 should have “\_r” versions provided, allowing the application control over the storage  
 124250 allocation. Those with variable-sized return values accept both a buffer address and a length  
 124251 parameter. The *sysconf()* variables are provided to supply the appropriate buffer sizes when  
 124252 required. Implementors are encouraged to apply the same rule when adapting their own  
 124253 existing functions to a pthreads environment.

#### 124254 B.2.9.2 Thread IDs

124255 Separate applications should communicate through well-defined interfaces and should not  
 124256 depend on each other’s implementation. For example, if a programmer decides to rewrite the  
 124257 *sort* utility using multiple threads, it should be easy to do this so that the interface to the *sort*  
 124258 utility does not change. Consider that if the user causes SIGINT to be generated while the *sort*  
 124259 utility is running, keeping the same interface means that the entire *sort* utility is killed, not just  
 124260 one of its threads. As another example, consider a realtime application that manages a reactor.  
 124261 Such an application may wish to allow other applications to control the priority at which it  
 124262 watches the control rods. One technique to accomplish this is to write the ID of the thread  
 124263 watching the control rods into a file and allow other programs to change the priority of that  
 124264 thread as they see fit. A simpler technique is to have the reactor process accept IPCs  
 124265 (Interprocess Communication messages) from other processes, telling it at a semantic level what  
 124266 priority the program should assign to watching the control rods. This allows the programmer  
 124267 greater flexibility in the implementation. For example, the programmer can change the  
 124268 implementation from having one thread per rod to having one thread watching all of the rods  
 124269 without changing the interface. Having threads live inside the process means that the  
 124270 implementation of a process is invisible to outside processes (excepting debuggers and system  
 124271 management tools).

124272 Threads do not provide a protection boundary. Every thread model allows threads to share  
 124273 memory with other threads and encourages this sharing to be widespread. This means that one  
 124274 thread can wipe out memory that is needed for the correct functioning of other threads that are  
 124275 sharing its memory. Consequently, providing each thread with its own user and/or group IDs  
 124276 would not provide a protection boundary between threads sharing memory.

124277 Some applications make the assumption that the implementation can always detect invalid uses  
 124278 of thread IDs of type **pthread\_t**. This is an invalid assumption. Specifically, if **pthread\_t** is  
 124279 defined as a pointer type, no access check needs to be performed before using the ID.

124280 As with other interfaces that take pointer parameters, the outcome of passing an invalid  
 124281 parameter can result in an invalid memory reference or an attempt to access an undefined  
 124282 portion of a memory object, cause signals to be sent (SIGSEGV or SIGBUS) and possible  
 124283 termination of the process. This is a similar case to passing an invalid buffer pointer to *read()*.  
 124284 Some implementations might implement *read()* as a system call and set an [EFAULT] error  
 124285 condition. Other implementations might contain parts of *read()* at user level and the first  
 124286 attempt to access data at an invalid reference will cause a signal to be sent instead.

124287 If an implementation detects use of a thread ID after the end of its lifetime, it is recommended  
 124288 that the function should fail and report an [ESRCH] error. This does not imply that  
 124289 implementations are required to return in this case. It is legitimate behavior to send an “invalid  
 124290 memory reference” signal (SIGSEGV or SIGBUS). It is the application’s responsibility to use only

124291 valid thread IDs and to keep track of the lifetime of the underlying threads.

124292 B.2.9.3 *Thread Mutexes*

124293 There is no additional rationale provided for this section.

124294 B.2.9.4 *Thread Scheduling*

124295 • Scheduling Implementation Models

124296 The following scheduling implementation models are presented in terms of threads and  
124297 “kernel entities”. This is to simplify exposition of the models, and it does not imply that  
124298 an implementation actually has an identifiable “kernel entity”.

124299 A kernel entity is not defined beyond the fact that it has scheduling attributes that are used  
124300 to resolve contention with other kernel entities for execution resources. A kernel entity  
124301 may be thought of as an envelope that holds a thread or a separate kernel thread. It is not a  
124302 conventional process, although it shares with the process the attribute that it has a single  
124303 thread of control; it does not necessarily imply an address space, open files, and so on. It is  
124304 better thought of as a primitive facility upon which conventional processes and threads  
124305 may be constructed.

124306 — System Thread Scheduling Model

124307 This model consists of one thread per kernel entity. The kernel entity is solely  
124308 responsible for scheduling thread execution on one or more processors. This model  
124309 schedules all threads against all other threads in the system using the scheduling  
124310 attributes of the thread.

124311 — Process Scheduling Model

124312 A generalized process scheduling model consists of two levels of scheduling. A  
124313 threads library creates a pool of kernel entities, as required, and schedules threads to  
124314 run on them using the scheduling attributes of the threads. Typically, the size of the  
124315 pool is a function of the simultaneously runnable threads, not the total number of  
124316 threads. The kernel then schedules the kernel entities onto processors according to  
124317 their scheduling attributes, which are managed by the threads library. This set model  
124318 potentially allows a wide range of mappings between threads and kernel entities.

124319 • System and Process Scheduling Model Performance

124320 There are a number of important implications on the performance of applications using  
124321 these scheduling models. The process scheduling model potentially provides lower  
124322 overhead for making scheduling decisions, since there is no need to access kernel-level  
124323 information or functions and the set of schedulable entities is smaller (only the threads  
124324 within the process).

124325 On the other hand, since the kernel is also making scheduling decisions regarding the  
124326 system resources under its control (for example, CPU(s), I/O devices, memory), decisions  
124327 that do not take thread scheduling parameters into account can result in unspecified  
124328 delays for realtime application threads, causing them to miss maximum response time  
124329 limits.

124330 • Rate Monotonic Scheduling

124331 Rate monotonic scheduling was considered, but rejected for standardization in the context  
124332 of pthreads. A sporadic server policy is included.

## • Scheduling Options

In POSIX.1-2008, the basic thread scheduling functions are defined under the threads functionality, so that they are required of all threads implementations. However, there are no specific scheduling policies required by this functionality to allow for conforming thread implementations that are not targeted to realtime applications.

Specific standard scheduling policies are defined to be under the Thread Execution Scheduling option, and they are specifically designed to support realtime applications by providing predictable resource-sharing sequences. The name of this option was chosen to emphasize that this functionality is defined as appropriate for realtime applications that require simple priority-based scheduling.

It is recognized that these policies are not necessarily satisfactory for some multi-processor implementations, and work is ongoing to address a wider range of scheduling behaviors. The interfaces have been chosen to create abundant opportunity for future scheduling policies to be implemented and standardized based on this interface. In order to standardize a new scheduling policy, all that is required (from the standpoint of thread scheduling attributes) is to define a new policy name, new members of the thread attributes object, and functions to set these members when the scheduling policy is equal to the new value.

## Scheduling Contention Scope

In order to accommodate the requirement for realtime response, each thread has a scheduling contention scope attribute. Threads with a system scheduling contention scope have to be scheduled with respect to all other threads in the system. These threads are usually bound to a single kernel entity that reflects their scheduling attributes and are directly scheduled by the kernel.

Threads with a process scheduling contention scope need be scheduled only with respect to the other threads in the process. These threads may be scheduled within the process onto a pool of kernel entities. The implementation is also free to bind these threads directly to kernel entities and let them be scheduled by the kernel. Process scheduling contention scope allows the implementation the most flexibility and is the default if both contention scopes are supported and none is specified.

Thus, the choice by implementors to provide one or the other (or both) of these scheduling models is driven by the need of their supported application domains for worst-case (that is, realtime) response, or average-case (non-realtime) response.

## Scheduling Allocation Domain

The SCHED\_FIFO and SCHED\_RR scheduling policies take on different characteristics on a multi-processor. Other scheduling policies are also subject to changed behavior when executed on a multi-processor. The concept of scheduling allocation domain determines the set of processors on which the threads of an application may run. By considering the application's processor scheduling allocation domain for its threads, scheduling policies can be defined in terms of their behavior for varying processor scheduling allocation domain values. It is conceivable that not all scheduling allocation domain sizes make sense for all scheduling policies on all implementations. The concept of scheduling allocation domain, however, is a useful tool for the description of multi-processor scheduling policies.

The "process control" approach to scheduling obtains significant performance advantages from dynamic scheduling allocation domain sizes when it is applicable.

Non-Uniform Memory Access (NUMA) multi-processors may use a system scheduling structure



124379 that involves reassignment of threads among scheduling allocation domains. In NUMA  
124380 machines, a natural model of scheduling is to match scheduling allocation domains to clusters of  
124381 processors. Load balancing in such an environment requires changing the scheduling allocation  
124382 domain to which a thread is assigned.

### 124383 **Scheduling Documentation**

124384 Implementation-provided scheduling policies need to be completely documented in order to be  
124385 useful. This documentation includes a description of the attributes required for the policy, the  
124386 scheduling interaction of threads running under this policy and all other supported policies, and  
124387 the effects of all possible values for processor scheduling allocation domain. Note that for the  
124388 implementor wishing to be minimally-compliant, it is (minimally) acceptable to define the  
124389 behavior as undefined.

### 124390 **Scheduling Contention Scope Attribute**

124391 The scheduling contention scope defines how threads compete for resources. Within  
124392 POSIX.1-2008, scheduling contention scope is used to describe only how threads are scheduled  
124393 in relation to one another in the system. That is, either they are scheduled against all other  
124394 threads in the system ("system scope") or only against those threads in the process ("process  
124395 scope"). In fact, scheduling contention scope may apply to additional resources, including  
124396 virtual timers and profiling, which are not currently considered by POSIX.1-2008.

### 124397 **Mixed Scopes**

124398 If only one scheduling contention scope is supported, the scheduling decision is straightforward.  
124399 To perform the processor scheduling decision in a mixed scope environment, it is necessary to  
124400 map the scheduling attributes of the thread with process-wide contention scope to the same  
124401 attribute space as the thread with system-wide contention scope.

124402 Since a conforming implementation has to support one and may support both scopes, it is useful  
124403 to discuss the effects of such choices with respect to example applications. If an implementation  
124404 supports both scopes, mixing scopes provides a means of better managing system-level (that is,  
124405 kernel-level) and library-level resources. In general, threads with system scope will require the  
124406 resources of a separate kernel entity in order to guarantee the scheduling semantics. On the  
124407 other hand, threads with process scope can share the resources of a kernel entity while  
124408 maintaining the scheduling semantics.

124409 The application is free to create threads with dedicated kernel resources, and other threads that  
124410 multiplex kernel resources. Consider the example of a window server. The server allocates two  
124411 threads per widget: one thread manages the widget user interface (including drawing), while  
124412 the other thread takes any required application action. This allows the widget to be "active"  
124413 while the application is computing. A screen image may be built from thousands of widgets. If  
124414 each of these threads had been created with system scope, then most of the kernel-level  
124415 resources might be wasted, since only a few widgets are active at any one time. In addition,  
124416 mixed scope is particularly useful in a window server where one thread with high priority and  
124417 system scope handles the mouse so that it tracks well. As another example, consider a database  
124418 server. For each of the hundreds or thousands of clients supported by a large server, an  
124419 equivalent number of threads will have to be created. If each of these threads were system scope,  
124420 the consequences would be the same as for the window server example above. However, the  
124421 server could be constructed so that actual retrieval of data is done by several dedicated threads.  
124422 Dedicated threads that do work for all clients frequently justify the added expense of system  
124423 scope. If it were not permissible to mix system and process threads in the same process, this type  
124424 of solution would not be possible.

## Dynamic Thread Scheduling Parameters Access

In many time-constrained applications, there is no need to change the scheduling attributes dynamically during thread or process execution, since the general use of these attributes is to reflect directly the time constraints of the application. Since these time constraints are generally imposed to meet higher-level system requirements, such as accuracy or availability, they frequently should remain unchanged during application execution.

However, there are important situations in which the scheduling attributes should be changed. Generally, this will occur when external environmental conditions exist in which the time constraints change. Consider, for example, a space vehicle major mode change, such as the change from ascent to descent mode, or the change from the space environment to the atmospheric environment. In such cases, the frequency with which many of the sensors or actuators need to be read or written will change, which will necessitate a priority change. In other cases, even the existence of a time constraint might be temporary, necessitating not just a priority change, but also a policy change for ongoing threads or processes. For this reason, it is critical that the interface should provide functions to change the scheduling parameters dynamically, but, as with many of the other realtime functions, it is important that applications use them properly to avoid the possibility of unnecessarily degrading performance.

In providing functions for dynamically changing the scheduling behavior of threads, there were two options: provide functions to get and set the individual scheduling parameters of threads, or provide a single interface to get and set all the scheduling parameters for a given thread simultaneously. Both approaches have merit. Access functions for individual parameters allow simpler control of thread scheduling for simple thread scheduling parameters. However, a single function for setting all the parameters for a given scheduling policy is required when first setting that scheduling policy. Since the single all-encompassing functions are required, it was decided to leave the interface as minimal as possible. Note that simpler functions (such as *pthread\_setprio()* for threads running under the priority-based schedulers) can be easily defined in terms of the all-encompassing functions.

If the *pthread\_setschedparam()* function executes successfully, it will have set all of the scheduling parameter values indicated in *param*; otherwise, none of the scheduling parameters will have been modified. This is necessary to ensure that the scheduling of this and all other threads continues to be consistent in the presence of an erroneous scheduling parameter.

The [EPERM] error value is included in the list of possible *pthread\_setschedparam()* error returns as a reflection of the fact that the ability to change scheduling parameters increases risks to the implementation and application performance if the scheduling parameters are changed improperly. For this reason, and based on some existing practice, it was felt that some implementations would probably choose to define specific permissions for changing either a thread's own or another thread's scheduling parameters. POSIX.1-2008 does not include portable methods for setting or retrieving permissions, so any such use of permissions is completely unspecified.

## Mutex Initialization Scheduling Attributes

In a priority-driven environment, a direct use of traditional primitives like mutexes and condition variables can lead to unbounded priority inversion, where a higher priority thread can be blocked by a lower priority thread, or set of threads, for an unbounded duration of time. As a result, it becomes impossible to guarantee thread deadlines. Priority inversion can be bounded and minimized by the use of priority inheritance protocols. This allows thread deadlines to be guaranteed even in the presence of synchronization requirements.

Two useful but simple members of the family of priority inheritance protocols are the basic priority inheritance protocol and the priority ceiling protocol emulation. Under the Basic

124473 Priority Inheritance protocol (governed by the Non-Robust Mutex Priority Inheritance option), a  
124474 thread that is blocking higher priority threads executes at the priority of the highest priority  
124475 thread that it blocks. This simple mechanism allows priority inversion to be bounded by the  
124476 duration of critical sections and makes timing analysis possible.

124477 Under the Priority Ceiling Protocol Emulation protocol (governed by the Thread Priority  
124478 Protection option), each mutex has a priority ceiling, usually defined as the priority of the  
124479 highest priority thread that can lock the mutex. When a thread is executing inside critical  
124480 sections, its priority is unconditionally increased to the highest of the priority ceilings of all the  
124481 mutexes owned by the thread. This protocol has two very desirable properties in uni-processor  
124482 systems. First, a thread can be blocked by a lower priority thread for at most the duration of one  
124483 single critical section. Furthermore, when the protocol is correctly used in a single processor, and  
124484 if threads do not become blocked while owning mutexes, mutual deadlocks are prevented.

124485 The priority ceiling emulation can be extended to multiple processor environments, in which  
124486 case the values of the priority ceilings will be assigned depending on the kind of mutex that is  
124487 being used: local to only one processor, or global, shared by several processors. Local priority  
124488 ceilings will be assigned the usual way, equal to the priority of the highest priority thread that  
124489 may lock that mutex. Global priority ceilings will usually be assigned a priority level higher  
124490 than all the priorities assigned to any of the threads that reside in the involved processors to  
124491 avoid the effect called remote blocking.

#### 124492 **Change the Priority Ceiling of a Mutex**

124493 In order for the priority protect protocol to exhibit its desired properties of bounding priority  
124494 inversion and avoidance of deadlock, it is critical that the ceiling priority of a mutex be the same  
124495 as the priority of the highest thread that can ever hold it, or higher. Thus, if the priorities of the  
124496 threads using such mutexes never change dynamically, there is no need ever to change the  
124497 priority ceiling of a mutex.

124498 However, if a major system mode change results in an altered response time requirement for one  
124499 or more application threads, their priority has to change to reflect it. It will occasionally be the  
124500 case that the priority ceilings of mutexes held also need to change. While changing priority  
124501 ceilings should generally be avoided, it is important that POSIX.1-2008 provide these interfaces  
124502 for those cases in which it is necessary.

#### 124503 *B.2.9.5 Thread Cancellation*

124504 Many existing threads packages have facilities for canceling an operation or canceling a thread.  
124505 These facilities are used for implementing user requests (such as the CANCEL button in a  
124506 window-based application), for implementing OR parallelism (for example, telling the other  
124507 threads to stop working once one thread has found a forced mate in a parallel chess program), or  
124508 for implementing the ABORT mechanism in Ada.

124509 POSIX programs traditionally have used the signal mechanism combined with either *longjmp()*  
124510 or polling to cancel operations. Many POSIX programmers have trouble using these facilities to  
124511 solve their problems efficiently in a single-threaded process. With the introduction of threads,  
124512 these solutions become even more difficult to use.

124513 The main issues with implementing a cancellation facility are specifying the operation to be  
124514 canceled, cleanly releasing any resources allocated to that operation, controlling when the target  
124515 notices that it has been canceled, and defining the interaction between asynchronous signals and  
124516 cancellation.



## Specifying the Operation to Cancel

Consider a thread that calls through five distinct levels of program abstraction and then, inside the lowest-level abstraction, calls a function that suspends the thread. (An abstraction boundary is a layer at which the client of the abstraction sees only the service being provided and can remain ignorant of the implementation. Abstractions are often layered, each level of abstraction being a client of the lower-level abstraction and implementing a higher-level abstraction.) Depending on the semantics of each abstraction, one could imagine wanting to cancel only the call that causes suspension, only the bottom two levels, or the operation being done by the entire thread. Canceling operations at a finer grain than the entire thread is difficult because threads are active and they may be run in parallel on a multi-processor. By the time one thread can make a request to cancel an operation, the thread performing the operation may have completed that operation and gone on to start another operation whose cancellation is not desired. Thread IDs are not reused until the thread has exited, and either it was created with the *Attr detachstate* attribute set to *PTHREAD\_CREATE\_DETACHED* or the *pthread\_join()* or *pthread\_detach()* function has been called for that thread. Consequently, a thread cancellation will never be misdirected when the thread terminates. For these reasons, the canceling of operations is done at the granularity of the thread. Threads are designed to be inexpensive enough so that a separate thread may be created to perform each separately cancelable operation; for example, each possibly long running user request.

For cancellation to be used in existing code, cancellation scopes and handlers will have to be established for code that needs to release resources upon cancellation, so that it follows the programming discipline described in the text.

## A Special Signal Versus a Special Interface

Two different mechanisms were considered for providing the cancellation interfaces. The first was to provide an interface to direct signals at a thread and then to define a special signal that had the required semantics. The other alternative was to use a special interface that delivered the correct semantics to the target thread.

The solution using signals produced a number of problems. It required the implementation to provide cancellation in terms of signals whereas a perfectly valid (and possibly more efficient) implementation could have both layered on a low-level set of primitives. There were so many exceptions to the special signal (it cannot be used with *kill()*, no POSIX.1 interfaces can be used with it) that it was clearly not a valid signal. Its semantics on delivery were also completely different from any existing POSIX.1 signal. As such, a special interface that did not mandate the implementation and did not confuse the semantics of signals and cancellation was felt to be the better solution.

## Races Between Cancellation and Resuming Execution

Due to the nature of cancellation, there is generally no synchronization between the thread requesting the cancellation of a blocked thread and events that may cause that thread to resume execution. For this reason, and because excess serialization hurts performance, when both an event that a thread is waiting for has occurred and a cancellation request has been made and cancellation is enabled, POSIX.1-2008 explicitly allows the implementation to choose between returning from the blocking call or acting on the cancellation request.

## Interaction of Cancellation with Asynchronous Signals

A typical use of cancellation is to acquire a lock on some resource and to establish a cancellation cleanup handler for releasing the resource when and if the thread is canceled.

A correct and complete implementation of cancellation in the presence of asynchronous signals requires considerable care. An implementation has to push a cancellation cleanup handler on the cancellation cleanup stack while maintaining the integrity of the stack data structure. If an asynchronously-generated signal is posted to the thread during a stack operation, the signal handler cannot manipulate the cancellation cleanup stack. As a consequence, asynchronous signal handlers may not cancel threads or otherwise manipulate the cancellation state of a thread. Threads may, of course, be canceled by another thread that used a *sigwait()* function to wait synchronously for an asynchronous signal.

In order for cancellation to function correctly, it is required that asynchronous signal handlers not change the cancellation state. This requires that some elements of existing practice, such as using *longjmp()* to exit from an asynchronous signal handler implicitly, be prohibited in cases where the integrity of the cancellation state of the interrupt thread cannot be ensured.

## Thread Cancellation Overview

- Cancelability States

The three possible cancelability states (disabled, deferred, and asynchronous) are encoded into two separate bits ((disable, enable) and (deferred, asynchronous)) to allow them to be changed and restored independently. For instance, short code sequences that will not block sometimes disable cancelability on entry and restore the previous state upon exit. Likewise, long or unbounded code sequences containing no convenient explicit cancellation points will sometimes set the cancelability type to asynchronous on entry and restore the previous value upon exit.

- Cancellation Points

Cancellation points are points inside of certain functions where a thread has to act on any pending cancellation request when cancelability is enabled. For functions in the “shall occur” list, a cancellation check must be performed on every call regardless of whether, absent the cancellation, the call would have blocked. For functions in the “may occur” list, a cancellation check may be performed on some calls but not others; i.e., whether or not a cancellation point occurs when one of these functions is being executed can depend on current conditions.

The idea was considered of allowing implementations to define whether blocking calls such as *read()* should be cancellation points. It was decided that it would adversely affect the design of conforming applications if blocking calls were not cancellation points because threads could be left blocked in an uncancelable state.

There are several important blocking routines that are specifically not made cancellation points:

- *pthread\_mutex\_lock()*

If *pthread\_mutex\_lock()* were a cancellation point, every routine that called it would also become a cancellation point (that is, any routine that touched shared state would automatically become a cancellation point). For example, *malloc()*, *free()*, and *rand()* would become cancellation points under this scheme. Having too many cancellation points makes programming very difficult, leading to either much disabling and restoring of cancelability or much difficulty in trying to arrange for reliable cleanup at every possible place.

124605 Since `pthread_mutex_lock()` is not a cancellation point, threads could result in being  
124606 blocked uninterruptibly for long periods of time if mutexes were used as a general  
124607 synchronization mechanism. As this is normally not acceptable, mutexes should only  
124608 be used to protect resources that are held for small fixed lengths of time where not  
124609 being able to be canceled will not be a problem. Resources that need to be held  
124610 exclusively for long periods of time should be protected with condition variables.

124611 — `pthread_barrier_wait()`

124612 Canceling a barrier wait will render a barrier unusable. Similar to a barrier timeout  
124613 (which the standard developers rejected), there is no way to guarantee the  
124614 consistency of a barrier's internal data structures if a barrier wait is canceled.

124615 — `pthread_spin_lock()`

124616 As with mutexes, spin locks should only be used to protect resources that are held for  
124617 small fixed lengths of time where not being cancelable will not be a problem.

124618 Every library routine should specify whether or not it includes any cancellation points.  
124619 Typically, only those routines that may block or compute indefinitely need to include  
124620 cancellation points.

124621 Correctly coded routines only reach cancellation points after having set up a cancellation  
124622 cleanup handler to restore invariants if the thread is canceled at that point. Being  
124623 cancelable only at specified cancellation points allows programmers to keep track of  
124624 actions needed in a cancellation cleanup handler more easily. A thread should only be  
124625 made asynchronously cancelable when it is not in the process of acquiring or releasing  
124626 resources or otherwise in a state from which it would be difficult or impossible to recover.

124627 • Thread Cancellation Cleanup Handlers

124628 The cancellation cleanup handlers provide a portable mechanism, easy to implement, for  
124629 releasing resources and restoring invariants. They are easier to use than signal handlers  
124630 because they provide a stack of cancellation cleanup handlers rather than a single handler,  
124631 and because they have an argument that can be used to pass context information to the  
124632 handler.

124633 The alternative to providing these simple cancellation cleanup handlers (whose only use is  
124634 for cleaning up when a thread is canceled) is to define a general exception package that  
124635 could be used for handling and cleaning up after hardware traps and software-detected  
124636 errors. This was too far removed from the charter of providing threads to handle  
124637 asynchrony. However, it is an explicit goal of POSIX.1-2008 to be compatible with existing  
124638 exception facilities and languages having exceptions.

124639 The interaction of this facility and other procedure-based or language-level exception  
124640 facilities is unspecified in this version of POSIX.1-2008. However, it is intended that it be  
124641 possible for an implementation to define the relationship between these cancellation  
124642 cleanup handlers and Ada, C++, or other language-level exception handling facilities.

124643 It was suggested that the cancellation cleanup handlers should also be called when the  
124644 process exits or calls the `exec` function. This was rejected partly due to the performance  
124645 problem caused by having to call the cancellation cleanup handlers of every thread before  
124646 the operation could continue. The other reason was that the only state expected to be  
124647 cleaned up by the cancellation cleanup handlers would be the intraprocess state. Any  
124648 handlers that are to clean up the interprocess state would be registered with `atexit()`. There  
124649 is the orthogonal problem that the `exec` functions do not honor the `atexit()` handlers, but  
124650 resolving this is beyond the scope of POSIX.1-2008.

- Async-Cancel Safety

A function is said to be async-cancel-safe if it is written in such a way that entering the function with asynchronous cancelability enabled will not cause any invariants to be violated, even if a cancellation request is delivered at any arbitrary instruction. Functions that are async-cancel-safe are often written in such a way that they need to acquire no resources for their operation and the visible variables that they may write are strictly limited.

Any routine that gets a resource as a side-effect cannot be made async-cancel-safe (for example, *malloc()*). If such a routine were called with asynchronous cancelability enabled, it might acquire the resource successfully, but as it was returning to the client, it could act on a cancellation request. In such a case, the application would have no way of knowing whether the resource was acquired or not.

Indeed, because many interesting routines cannot be made async-cancel-safe, most library routines in general are not async-cancel-safe. Every library routine should specify whether or not it is async-cancel safe so that programmers know which routines can be called from code that is asynchronously cancelable.

IEEE Std 1003.1-2001/Cor 1-2002, item XSH/TC1/D6/8 is applied, adding the *pselect()* function to the list of functions with cancellation points.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/5 is applied, adding the *fdatasync()* function into the table of functions that shall have cancellation points.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/6 is applied, adding the numerous functions into the table of functions that may have cancellation points.

IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/7 is applied, clarifying the requirements in Thread Cancellation Cleanup Handlers.

#### B.2.9.6 Thread Read-Write Locks

##### Background

Read-write locks are often used to allow parallel access to data on multi-processors, to avoid context switches on uni-processors when multiple threads access the same data, and to protect data structures that are frequently accessed (that is, read) but rarely updated (that is, written). The in-core representation of a file system directory is a good example of such a data structure. One would like to achieve as much concurrency as possible when searching directories, but limit concurrent access when adding or deleting files.

Although read-write locks can be implemented with mutexes and condition variables, such implementations are significantly less efficient than is possible. Therefore, this synchronization primitive is included in POSIX.1-2008 for the purpose of allowing more efficient implementations in multi-processor systems.

##### Queuing of Waiting Threads

The *pthread\_rwlock\_unlock()* function description states that one writer or one or more readers must acquire the lock if it is no longer held by any thread as a result of the call. However, the function does not specify which thread(s) acquire the lock, unless the Thread Execution Scheduling option is supported.

The standard developers considered the issue of scheduling with respect to the queuing of threads blocked on a read-write lock. The question turned out to be whether POSIX.1-2008 should require priority scheduling of read-write locks for threads whose execution scheduling

124695 policy is priority-based (for example, SCHED\_FIFO or SCHED\_RR). There are tradeoffs  
 124696 between priority scheduling, the amount of concurrency achievable among readers, and the  
 124697 prevention of writer and/or reader starvation.

124698 For example, suppose one or more readers hold a read-write lock and the following threads  
 124699 request the lock in the listed order:

124700 pthread\_rwlock\_wrlock() - Low priority thread writer\_a  
 124701 pthread\_rwlock\_rdlock() - High priority thread reader\_a  
 124702 pthread\_rwlock\_rdlock() - High priority thread reader\_b  
 124703 pthread\_rwlock\_rdlock() - High priority thread reader\_c

124704 When the lock becomes available, should *writer\_a* block the high priority readers? Or, suppose a  
 124705 read-write lock becomes available and the following are queued:

124706 pthread\_rwlock\_rdlock() - Low priority thread reader\_a  
 124707 pthread\_rwlock\_rdlock() - Low priority thread reader\_b  
 124708 pthread\_rwlock\_rdlock() - Low priority thread reader\_c  
 124709 pthread\_rwlock\_wrlock() - Medium priority thread writer\_a  
 124710 pthread\_rwlock\_rdlock() - High priority thread reader\_d

124711 If priority scheduling is applied then *reader\_d* would acquire the lock and *writer\_a* would block  
 124712 the remaining readers. But should the remaining readers also acquire the lock to increase  
 124713 concurrency? The solution adopted takes into account that when the Thread Execution  
 124714 Scheduling option is supported, high priority threads may in fact starve low priority threads  
 124715 (the application developer is responsible in this case for designing the system in such a way that  
 124716 this starvation is avoided). Therefore, POSIX.1-2008 specifies that high priority readers take  
 124717 precedence over lower priority writers. However, to prevent writer starvation from threads of  
 124718 the same or lower priority, writers take precedence over readers of the same or lower priority.

124719 Priority inheritance mechanisms are non-trivial in the context of read-write locks. When a high  
 124720 priority writer is forced to wait for multiple readers, for example, it is not clear which subset of  
 124721 the readers should inherit the writer's priority. Furthermore, the internal data structures that  
 124722 record the inheritance must be accessible to all readers, and this implies some sort of  
 124723 serialization that could negate any gain in parallelism achieved through the use of multiple  
 124724 readers in the first place. Finally, existing practice does not support the use of priority  
 124725 inheritance for read-write locks. Therefore, no specification of priority inheritance or priority  
 124726 ceiling is attempted. If reliable priority-scheduled synchronization is absolutely required, it can  
 124727 always be obtained through the use of mutexes.

## 124728 **Comparison to fcntl() Locks**

124729 The read-write locks and the *fcntl()* locks in POSIX.1-2008 share a common goal: increasing  
 124730 concurrency among readers, thus increasing throughput and decreasing delay.

124731 However, the read-write locks have two features not present in the *fcntl()* locks. First, under  
 124732 priority scheduling, read-write locks are granted in priority order. Second, also under priority  
 124733 scheduling, writer starvation is prevented by giving writers preference over readers of equal or  
 124734 lower priority.

124735 Also, read-write locks can be used in systems lacking a file system, such as those conforming to  
 124736 the minimal realtime system profile of IEEE Std 1003.13-1998.

124737      **History of Resolution Issues**

124738      Based upon some balloting objections, early drafts specified the behavior of threads waiting on a  
124739      read-write lock during the execution of a signal handler, as if the thread had not called the lock  
124740      operation. However, this specified behavior would require implementations to establish  
124741      internal signal handlers even though this situation would be rare, or never happen for many  
124742      programs. This would introduce an unacceptable performance hit in comparison to the little  
124743      additional functionality gained. Therefore, the behavior of read-write locks and signals was  
124744      reverted back to its previous mutex-like specification.

124745      *B.2.9.7      Thread Interactions with Regular File Operations*

124746      There is no additional rationale provided for this section.

124747      *B.2.9.8      Use of Application-Managed Thread Stacks*

124748      IEEE Std 1003.1-2001/Cor 2-2004, item XSH/TC2/D6/8 is applied, adding this new section. It  
124749      was added to make it clear that the current standard does not allow an application to determine  
124750      when a stack can be reclaimed. This may be addressed in a future version.

124751      **B.2.10      Sockets**

124752      The base document for the sockets interfaces in POSIX.1-2008 is the XNS, Issue 5.2 specification.  
124753      This was primarily chosen as it aligns with IPv6. Additional material has been added from  
124754      IEEE Std 1003.1g-2000, notably socket concepts, raw sockets, the *pselect()* function, the  
124755      *socketmark()* function, and the **<sys/select.h>** header.

124756      *B.2.10.1      Address Families*

124757      There is no additional rationale provided for this section.

124758      *B.2.10.2      Addressing*

124759      There is no additional rationale provided for this section.

124760      *B.2.10.3      Protocols*

124761      There is no additional rationale provided for this section.

124762      *B.2.10.4      Routing*

124763      There is no additional rationale provided for this section.

124764      *B.2.10.5      Interfaces*

124765      There is no additional rationale provided for this section.



124766 B.2.10.6 *Socket Types*

124767 The type **socklen\_t** was invented to cover the range of implementations seen in the field. The  
 124768 intent of **socklen\_t** is to be the type for all lengths that are naturally bounded in size; that is, that  
 124769 they are the length of a buffer which cannot sensibly become of massive size: network addresses,  
 124770 host names, string representations of these, ancillary data, control messages, and socket options  
 124771 are examples. Truly boundless sizes are represented by **size\_t** as in *read()*, *write()*, and so on.

124772 All **socklen\_t** types were originally (in BSD UNIX) of type **int**. During the development of  
 124773 POSIX.1-2008, it was decided to change all buffer lengths to **size\_t**, which appears at face value  
 124774 to make sense. When dual mode 32/64-bit systems came along, this choice unnecessarily  
 124775 complicated system interfaces because **size\_t** (with **long**) was a different size under ILP32 and  
 124776 LP64 models. Reverting to **int** would have happened except that some implementations had  
 124777 already shipped 64-bit-only interfaces. The compromise was a type which could be defined to be  
 124778 any size by the implementation: **socklen\_t**.

124779 B.2.10.7 *Socket I/O Mode*

124780 There is no additional rationale provided for this section.

124781 B.2.10.8 *Socket Owner*

124782 There is no additional rationale provided for this section.

124783 B.2.10.9 *Socket Queue Limits*

124784 There is no additional rationale provided for this section.

124785 B.2.10.10 *Pending Error*

124786 There is no additional rationale provided for this section.

124787 B.2.10.11 *Socket Receive Queue*

124788 There is no additional rationale provided for this section.

124789 B.2.10.12 *Socket Out-of-Band Data State*

124790 There is no additional rationale provided for this section.

124791 B.2.10.13 *Connection Indication Queue*

124792 There is no additional rationale provided for this section.

124793 B.2.10.14 *Signals*

124794 There is no additional rationale provided for this section.

124795 *B.2.10.15 Asynchronous Errors*

124796 There is no additional rationale provided for this section.

124797 *B.2.10.16 Use of Options*

124798 There is no additional rationale provided for this section.

124799 *B.2.10.17 Use of Sockets for Local UNIX Connections*

124800 There is no additional rationale provided for this section.

124801 *B.2.10.18 Use of Sockets over Internet Protocols*

124802 A raw socket allows privileged users direct access to a protocol; for example, raw access to the  
124803 IP and ICMP protocols is possible through raw sockets. Raw sockets are intended for  
124804 knowledgeable applications that wish to take advantage of some protocol feature not directly  
124805 accessible through the other sockets interfaces.

124806 *B.2.10.19 Use of Sockets over Internet Protocols Based on IPv4*

124807 There is no additional rationale provided for this section.

124808 *B.2.10.20 Use of Sockets over Internet Protocols Based on IPv6*

124809 The Open Group Base Resolution bwg2001-012 is applied, clarifying that IPv6 implementations  
124810 are required to support use of AF\_INET6 sockets over IPv4.

124811 **B.2.11 Tracing**

124812 The organization of the tracing rationale differs from the traditional rationale in that this tracing  
124813 rationale text is written against the trace interface as a whole, rather than against the individual  
124814 components of the trace interface or the normative section in which those components are  
124815 defined. Therefore the sections below do not parallel the sections of normative text in  
124816 POSIX.1-2008.

124817 *B.2.11.1 Objectives*

124818 The intended uses of tracing are application-system debugging during system development, as a  
124819 “flight recorder” for maintenance of fielded systems, and as a performance measurement tool.  
124820 In all of these intended uses, the vendor-supplied computer system and its software are, for this  
124821 discussion, assumed error-free; the intent being to debug the user-written and/or third-party  
124822 application code, and their interactions. Clearly, problems with the vendor-supplied system and  
124823 its software will be uncovered from time to time, but this is a byproduct of the primary activity,  
124824 debugging user code.

124825 Another need for defining a trace interface in POSIX stems from the objective to provide an  
124826 efficient portable way to perform benchmarks. Existing practice shows that such interfaces are  
124827 commonly used in a variety of systems but with little commonality. As part of the benchmarking  
124828 needs, two aspects within the trace interface must be considered.

124829 The first, and perhaps more important one, is the qualitative aspect.



The second is the quantitative aspect.

- Qualitative Aspect

To better understand this aspect, let us consider an example. Suppose that you want to organize a number of actions to be performed during the day. Some of these actions are known at the beginning of the day. Some others, which may be more or less important, will be triggered by reading your mail. During the day you will make some phone calls and synchronously receive some more information. Finally you will receive asynchronous phone calls that also will trigger actions. If you, or somebody else, examines your day at work, you, or he, can discover that you have not efficiently organized your work. For instance, relative to the phone calls you made, would it be preferable to make some of these early in the morning? Or to delay some others until the end of the day? Relative to the phone calls you have received, you might find that somebody you called in the morning has called you 10 times while you were performing some important work. To examine, afterwards, your day at work, you record in sequence all the trace events relative to your work. This should give you a chance of organizing your next day at work.

This is the qualitative aspect of the trace interface. The user of a system needs to keep a trace of particular points the application passes through, so that he can eventually make some changes in the application and/or system configuration, to give the application a chance of running more efficiently.

- Quantitative Aspect

This aspect concerns primarily realtime applications, where missed deadlines can be undesirable. Although there are, in POSIX.1-2008, some interfaces useful for such applications (timeouts, execution time monitoring, and so on), there are no APIs to aid in the tuning of a realtime application's behavior (**timespec** in timeouts, length of message queues, duration of driver interrupt service routine, and so on). The tuning of an application needs a means of recording timestamped important trace events during execution in order to analyze offline, and eventually, to tune some realtime features (redesign the system with less functionalities, readjust timeouts, redesign driver interrupts, and so on).

### Detailed Objectives

Objectives were defined to build the trace interface and are kept for historical interest. Although some objectives are not fully respected in this trace interface, the concept of the POSIX trace interface assumes the following points:

1. It must be possible to trace both system and user trace events concurrently.
2. It must be possible to trace per-process trace events and also to trace system trace events which are unrelated to any particular process. A per-process trace event is either user-initiated or system-initiated.
3. It must be possible to control tracing on a per-process basis from either inside or outside the process.
4. It must be possible to control tracing on a per-thread basis from inside the enclosing process.
5. Trace points must be controllable by trace event type ID from inside and outside of the process. Multiple trace points can have the same trace event type ID, and will be controlled jointly.

- 124874 6. Recording of trace events is dependent on both trace event type ID and the  
124875 process/thread. Both must be enabled in order to record trace events. System trace events  
124876 may or may not be handled differently.
- 124877 7. The API must not mandate the ability to control tracing for more than one process at the  
124878 same time.
- 124879 8. There is no objective for trace control on anything bigger than a process; for example,  
124880 group or session.
- 124881 9. Trace propagation and control:
- 124882     a. Trace propagation across *fork()* is optional; the default is to not trace a child  
124883 process.
- 124884     b. Trace control must span *pthread\_create()* operations; that is, if a process is being  
124885 traced, any thread will be traced as well if this thread allows tracing. The default is  
124886 to allow tracing.
- 124887 10. Trace control must not span *exec* or *posix\_spawn()* operations.
- 124888 11. A triggering API is not required. The triggering API is the ability to command or stop  
124889 tracing based on the occurrence of a specific trace event other than a  
124890 POSIX\_TRACE\_START trace event or a POSIX\_TRACE\_STOP trace event.
- 124891 12. Trace log entries must have timestamps of implementation-defined resolution.  
124892 Implementations are exhorted to support at least microsecond resolution. When a trace  
124893 log entry is retrieved, it must have timestamp, PC address, PID, and TID of the entity that  
124894 generated the trace event.
- 124895 13. Independently developed code should be able to use trace facilities without coordination  
124896 and without conflict.
- 124897 14. Even if the trace points in the trace calls are not unique, the trace log entries (after any  
124898 processing) must be uniquely identified as to trace point.
- 124899 15. There must be a standard API to read the trace stream.
- 124900 16. The format of the trace stream and the trace log is opaque and unspecified.
- 124901 17. It must be possible to read a completed trace, if recorded on some suitable non-volatile  
124902 storage, even subsequent to a power cycle or subsequent cold boot of the system.
- 124903 18. Support of analysis of a trace log while it is being formed is implementation-defined.
- 124904 19. The API must allow the application to write trace stream identification information into  
124905 the trace stream and to be able to retrieve it, without it being overwritten by trace entries,  
124906 even if the trace stream is full.
- 124907 20. It must be possible to specify the destination of trace data produced by trace events.
- 124908 21. It must be possible to have different trace streams, and for the tracing enabled by one  
124909 trace stream to be completely independent of the tracing of another trace stream.
- 124910 22. It must be possible to trace events from threads in different CPUs.
- 124911 23. The API must support one or more trace streams per-system, and one or more trace  
124912 streams per-process, up to an implementation-defined set of per-system and per-process  
124913 maximums.
- 124914 24. It must be possible to determine the order in which the trace events happened, without  
124915 necessarily depending on the clock, up to an implementation-defined time resolution.

- 124916 25. For performance reasons, the trace event point call(s) must be implementable as a macro  
124917 (see the ISO POSIX-1: 1996 standard, 1.3.4, Statement 2).
- 124918 26. POSIX.1-2008 must not define the trace points which a conforming system must  
124919 implement, except for trace points used in the control of tracing.
- 124920 27. The APIs must be thread-safe, and trace points should be lock-free (that is, not require a  
124921 lock to gain exclusive access to some resource).
- 124922 28. The user-provided information associated with a trace event is variable-sized, up to some  
124923 maximum size.
- 124924 29. Bounds on record and trace stream sizes:
- 124925 a. The API must permit the application to declare the upper bounds on the length of  
124926 an application data record. The system must return the limit it used. The limit used  
124927 may be smaller than requested.
- 124928 b. The API must permit the application to declare the upper bounds on the size of  
124929 trace streams. The system must return the limit it used. The limit used may be  
124930 different, either larger or smaller, than requested.
- 124931 30. The API must be able to pass any fundamental data type, and a structured data type  
124932 composed only of fundamental types. The API must be able to pass data by reference,  
124933 given only as an address and a length. Fundamental types are the POSIX.1 types (see the  
124934 `<sys/types.h>` header) plus those defined in the ISO C standard.
- 124935 31. The API must apply the POSIX notions of ownership and permission to recorded trace  
124936 data, corresponding to the sources of that data.

#### 124937 **Comments on Objectives**

124938 **Note:** In the following comments, numbers in square brackets refer to the above objectives.

124939 It is necessary to be able to obtain a trace stream for a complete activity. Thus there is a  
124940 requirement to be able to trace both application and system trace events. A per-process trace  
124941 event is either user-initiated, like the *write()* function, or system-initiated, like a timer expiration.  
124942 There is also a need to be able to trace the activity of an entire process even when it has threads  
124943 in multiple CPUs. To avoid excess trace activity, it is necessary to be able to control tracing on a  
124944 trace event type basis.  
124945 [Objectives 1,2,5,22]

124946 There is a need to be able to control tracing on a per-process basis, both from inside and outside  
124947 the process; that is, a process can start a trace activity on itself or any other process. There is also  
124948 the perceived need to allow the definition of a maximum number of trace streams per system.  
124949 [Objectives 3,23]

124950 From within a process, it is necessary to be able to control tracing on a per-thread basis. This  
124951 provides an additional filtering capability to keep the amount of traced data to a minimum. It  
124952 also allows for less ambiguity as to the origin of trace events. It is recognized that thread-level  
124953 control is only valid from within the process itself. It is also desirable to know the maximum  
124954 number of trace streams per process that can be started. The API should not require thread  
124955 synchronization or mandate priority inversions that would cause the thread to block. However,  
124956 the API must be thread-safe.  
124957 [Objectives 4,23,24,27]

124958 There was no perceived objective to control tracing on anything larger than a process; for  
124959 example, a group or session. Also, the ability to start or stop a trace activity on multiple  
124960 processes atomically may be very difficult or cumbersome in some implementations.  
124961 [Objectives 6,8]

124962 It is also necessary to be able to control tracing by trace event type identifier, sometimes called a  
124963 trace hook ID. However, there is no mandated set of system trace events, since such trace points  
124964 are implementation-defined. The API must not require from the operating system facilities that  
124965 are not standard.  
124966 [Objectives 6,26]

124967 Trace control must span *fork()* and *pthread\_create()*. If not, there will be no way to ensure that an  
124968 application's activity is entirely traced. The newly forked child would not be able to turn on its  
124969 tracing until after it obtained control after the fork, and trace control externally would be even  
124970 more problematic.  
124971 [Objective 9]

124972 Since *exec* and *posix\_spawn()* represent a complete change in the execution of a task (a new  
124973 program), trace control need not persist over an *exec* or *posix\_spawn()*.  
124974 [Objective 10]

124975 Where trace activities are started on multiple processes, these trace activities should not interfere  
124976 with each other.  
124977 [Objective 21]

124978 There is no need for a triggering objective, primarily for performance reasons; see also [Section](#)  
124979 [B.2.11.8](#) (on page 3675), rationale on triggering.  
124980 [Objective 11]

124981 It must be possible to determine the origin of each traced event. The process and thread  
124982 identifiers for each trace event are needed. Also there was a perceived need for a user-specifiable  
124983 origin, but it was felt that this would create too much overhead.  
124984 [Objectives 12,14]

124985 An allowance must be made for trace points to come embedded in software components from  
124986 several different sources and vendors without requiring coordination.  
124987 [Objective 13]

124988 There is a requirement to be able to uniquely identify trace points that may have the same trace  
124989 stream identifier. This is only necessary when a trace report is produced.  
124990 [Objectives 12,14]

124991 Tracing is a very performance-sensitive activity, and will therefore likely be implemented at a  
124992 low level within the system. Hence the interface must not mandate any particular buffering or  
124993 storage method. Therefore, a standard API is needed to read a trace stream. Also the interface  
124994 must not mandate the format of the trace data, and the interface must not assume a trace storage  
124995 method. Due to the possibility of a monolithic kernel and the possible presence of multiple  
124996 processes capable of running trace activities, the two kinds of trace events may be stored in two  
124997 separate streams for performance reasons. A mandatory dump mechanism, common in some  
124998 existing practice, has been avoided to allow the implementation of this set of functions on small  
124999 realtime profiles for which the concept of a file system is not defined. The trace API calls should  
125000 be implemented as macros.  
125001 [Objectives 15,16,25,30]

125002 Since a trace facility is a valuable service tool, the output (or log) of a completed trace stream  
125003 that is written to permanent storage must be readable on other systems of the type that  
125004 produced the trace log. Note that there is no objective to be able to interpret a trace log that was  
125005 not successfully completed.  
125006 [Objectives 17,18,19]

125007 For trace streams written to permanent storage, a way to specify the destination of the trace  
125008 stream is needed.  
125009 [Objective 20]

There is a requirement to be able to depend on the ordering of trace events up to some implementation-defined time interval. For example, there is a need to know the time period during which, if trace events are closer together, their ordering is unspecified. Events that occur within an interval smaller than this resolution may or may not be read back in the correct order. [Objective 24]

The application should be able to know how much data can be traced. When trace event types can be filtered, the application should be able to specify the approximate maximum amount of data that will be traced in a trace event so resources can be more efficiently allocated. [Objectives 28,29]

Users should not be able to trace data to which they would not normally have access. System trace events corresponding to a process/thread should be associated with the ownership of that process/thread. [Objective 31]

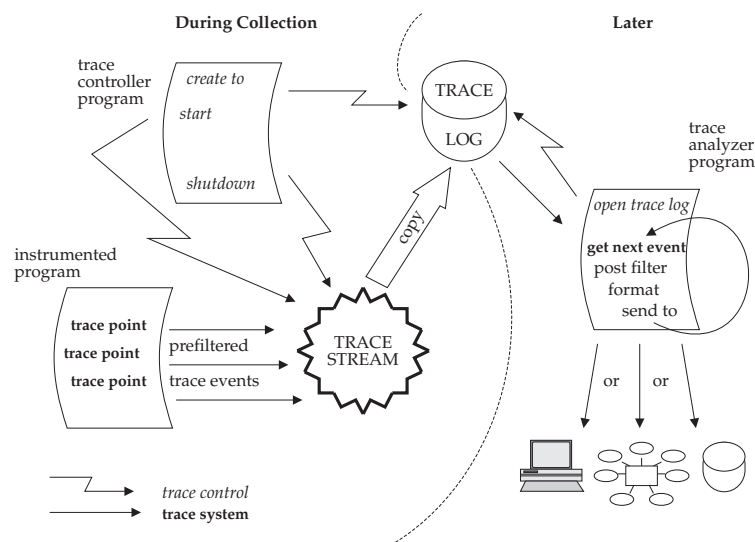
#### B.2.11.2 Trace Model

##### Introduction

The model is based on two base entities: the “Trace Stream” and the “Trace Log”, and a recorded unit called the “Trace Event”. The possibility of using Trace Streams and Trace Logs separately gives two use dimensions and solves both the performance issue and the full-information system issue. In the case of a trace stream without log, specific information, although reduced in quantity, is required to be registered, in a possibly small realtime system, with as little overhead as possible. The Trace Log option has been added for small realtime systems. In the case of a trace stream with log, considerable complex application-specific information needs to be collected.

##### Trace Model Description

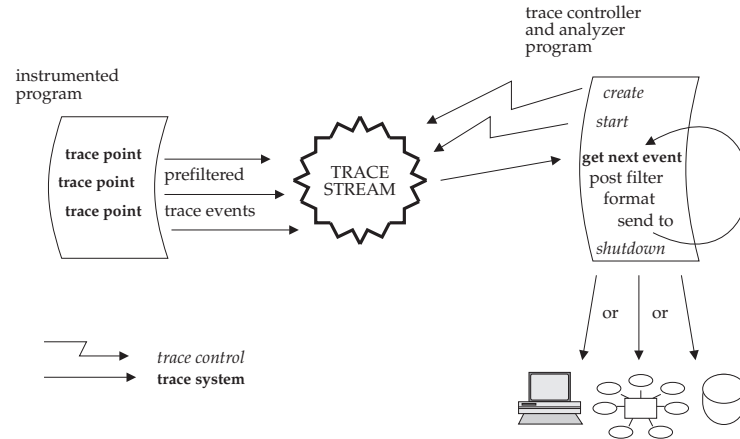
The trace model can be examined for three different subfunctions: Application Instrumentation, Trace Operation Control, and Trace Analysis.



**Figure B-2** Trace System Overview: for Offline Analysis

- 125037 Each of these subfunctions requires specific characteristics of the trace mechanism API.
- 125038 • Application Instrumentation
- 125039 When instrumenting an application, the programmer is not concerned about the future use  
 125040 of the trace events in the trace stream or the trace log, the full policy of the trace stream, or  
 125041 the eventual pre-filtering of trace events. But he is concerned about the correct  
 125042 determination of the specific trace event type identifier, regardless of how many  
 125043 independent libraries are used in the same user application; see [Figure B-2](#) (on page 3657)  
 125044 and [Figure B-3](#) (on page 3659).
- 125045 This trace API provides the necessary operations to accomplish this subfunction. This is  
 125046 done by providing functions to associate a programmer-defined name with an  
 125047 implementation-defined trace event type identifier (see the *posix\_trace\_eventid\_open()*  
 125048 function), and to send this trace event into a potential trace stream (see the  
 125049 *posix\_trace\_event()* function).
- 125050 • Trace Operation Control
- 125051 When controlling the recording of trace events in a trace stream, the programmer is  
 125052 concerned with the correct initialization of the trace mechanism (that is, the sizing of the  
 125053 trace stream), the correct retention of trace events in a permanent storage, the correct  
 125054 dynamic recording of trace events, and so on.
- 125055 This trace API provides the necessary material to permit this efficiently. This is done by  
 125056 providing functions to initialize a new trace stream, and optionally a trace log:
- 125057 — Trace Stream Attributes Object Initialization (see *posix\_trace\_attr\_init()*)
  - 125058 — Functions to Retrieve or Set Information About a Trace Stream (see  
 125059 *posix\_trace\_attr\_getgenversion()*)
  - 125060 — Functions to Retrieve or Set the Behavior of a Trace Stream (see  
 125061 *posix\_trace\_attr\_getinherited()*)
  - 125062 — Functions to Retrieve or Set Trace Stream Size Attributes (see  
 125063 *posix\_trace\_attr\_getmaxusereventsize()*)
  - 125064 — Trace Stream Initialization, Flush, and Shutdown from a Process (see  
 125065 *posix\_trace\_create()*)
  - 125066 — Clear Trace Stream and Trace Log (see *posix\_trace\_clear()*)
- 125067 To select the trace event types that are to be traced:
- 125068 — Manipulate Trace Event Type Identifier (see *posix\_trace\_trid\_eventid\_open()*)
  - 125069 — Iterate over a Mapping of Trace Event Type (see *posix\_trace\_eventtypelist\_getnext\_id()*)
  - 125070 — Manipulate Trace Event Type Sets (see *posix\_trace\_eventset\_empty()*)
  - 125071 — Set Filter of an Initialized Trace Stream (see *posix\_trace\_set\_filter()*)
- 125072 To control the execution of an active trace stream:
- 125073 — Trace Start and Stop (see *posix\_trace\_start()*)
  - 125074 — Functions to Retrieve the Trace Attributes or Trace Statuses (see  
 125075 *posix\_trace\_get\_attr()*)





**Figure B-3** Trace System Overview: for Online Analysis

### • Trace Analysis

Once correctly recorded, on permanent storage or not, an ultimate activity consists of the analysis of the recorded information. If the recorded data is on permanent storage, a specific open operation is required to associate a trace stream to a trace log.

The first intent of the group was to request the presence of a system identification structure in the trace stream attribute. This was, for the application, to allow some portable way to process the recorded information. However, there is no requirement that the **utsname** structure, on which this system identification was based, be portable from one machine to another, so the contents of the attribute cannot be interpreted correctly by an application conforming to POSIX.1-2008.

This modification has been incorporated and requests that some unspecified information be recorded in the trace log in order to fail opening it if the analysis process and the controller process were running in different types of machine, but does not request that this information be accessible to the application. This modification has implied a modification in the `posix_trace_open()` function error code returns.

This trace API provides functions to:

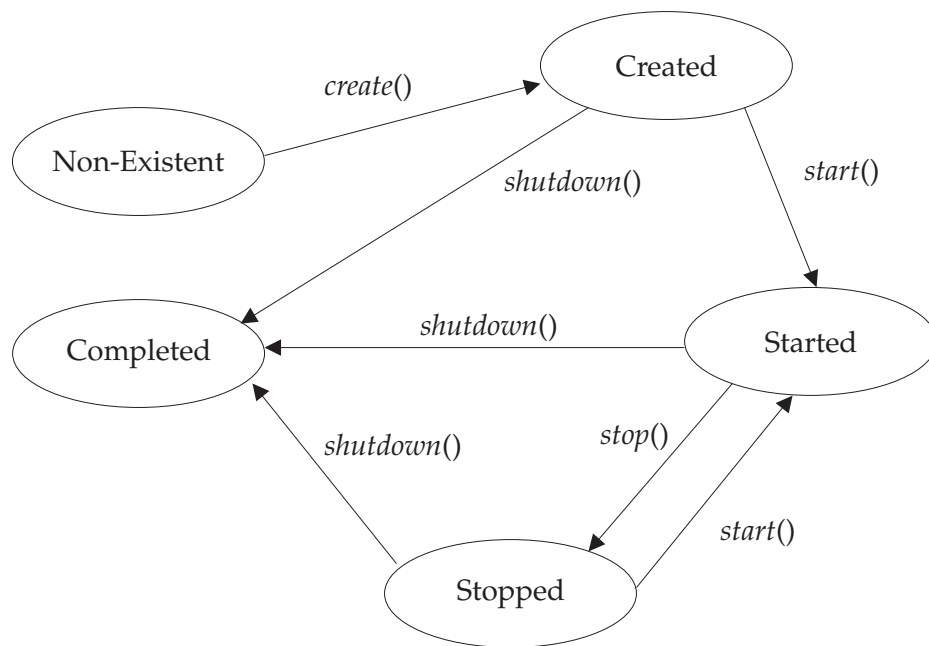
- Extract trace stream identification attributes (see `posix_trace_attr_getgenversion()`)
- Extract trace stream behavior attributes (see `posix_trace_attr_getinherited()`)
- Extract trace event, stream, and log size attributes (see `posix_trace_attr_getmaxusereventsize()`)
- Look up trace event type names (see `posix_trace_eventid_get_name()`)
- Iterate over trace event type identifiers (see `posix_trace_eventtypelist_getnext_id()`)
- Open, rewind, and close a trace log (see `posix_trace_open()`)
- Read trace stream attributes and status (see `posix_trace_get_attr()`)
- Read trace events (see `posix_trace_getnext_event()`)

Due to the following two reasons:

1. The requirement that the trace system must not add unacceptable overhead to the traced process and so that the trace event point execution must be fast
2. The traced application does not care about tracing errors

the trace system cannot return any internal error to the application. Internal error conditions can range from unrecoverable errors that will force the active trace stream to abort, to small errors that can affect the quality of tracing without aborting the trace stream. The group decided to define a system trace event to report to the analysis process such internal errors. It is not the intention of POSIX.1-2008 to require an implementation to report an internal error that corrupts or terminates tracing operation. The implementor is free to decide which internal documented errors, if any, the trace system is able to report.

### States of a Trace Stream



**Figure B-4** Trace System Overview: States of a Trace Stream

Figure B-4 shows the different states an active trace stream passes through. After the `posix_trace_create()` function call, a trace stream becomes CREATED and a trace stream is associated for the future collection of trace events. The status of the trace stream is `POSIX_TRACE_SUSPENDED`. The state becomes STARTED after a call to the `posix_trace_start()` function, and the status becomes `POSIX_TRACE_RUNNING`. In this state, all trace events that are not filtered out will be stored into the trace stream. After a call to `posix_trace_stop()`, the trace stream becomes STOPPED (and the status `POSIX_TRACE_SUSPENDED`). In this state, no new trace events will be recorded in the trace stream, but previously recorded trace events may continue to be read.

After a call to `posix_trace_shutdown()`, the trace stream is in the state COMPLETED. The trace stream no longer exists but, if the Trace Log option is supported, all the information contained in it has been logged. If a log object has not been associated with the trace stream at the creation, it



is the responsibility of the trace controller process to not shut the trace stream down while trace events remain to be read in the stream.

### Tracing All Processes

Some implementations have a tracing subsystem with the ability to trace all processes. This is useful to debug some types of device drivers such as those for ATM or X25 adapters. These types of adapters are used by several independent processes, that are not issued from the same process.

The POSIX trace interface does not define any constant or option to create a trace stream tracing all processes. POSIX.1 does not prevent this type of implementation and an implementor is free to add this capability. Nevertheless, the trace interface allows tracing of all the system trace events and all the processes issued from the same process.

If such a tracing system capability has to be implemented, when a trace stream is created, it is recommended that a constant named `POSIX_TRACE_ALLPROC` be used instead of the process identifier in the argument of the `posix_trace_create()` or `posix_trace_create_withlog()` function. A possible value for `POSIX_TRACE_ALLPROC` may be `-1` instead of a real process identifier.

The implementor has to be aware that there is some impact on the tracing behavior as defined in the POSIX trace interface. For example:

- If the default value for the inheritance attribute is set to `POSIX_TRACE_CLOSE_FOR_CHILD`, the implementation has to stop tracing for the child process.
- The trace controller which is creating this type of trace stream must have the appropriate privilege to trace all the processes.

### Trace Storage

The model is based on two types of trace events: system trace events and user-defined trace events. The internal representation of trace events is implementation-defined, and so the implementor is free to choose the more suitable, practical, and efficient way to design the internal management of trace events. For the timestamping operation, the model does not impose the `CLOCK_REALTIME` or any other clock. The buffering allocation and operation follow the same principle. The implementor is free to use one or more buffers to record trace events; the interface assumes only a logical trace stream of sequentially recorded trace events. Regarding flushing of trace events, the interface allows the definition of a trace log object which typically can be a file. But the group was also aware of defining functions to permit the use of this interface in small realtime systems, which may not have general file system capabilities. For instance, the three functions `posix_trace_getnext_event()` (blocking), `posix_trace_timedgetnext_event()` (blocking with timeout), and `posix_trace_trygetnext_event()` (non-blocking) are proposed to read the recorded trace events.

The policy to be used when the trace stream becomes full also relies on common practice:

- For an active trace stream, the `POSIX_TRACE_LOOP` trace stream policy permits automatic overrun (overwrite of oldest trace events) while waiting for some user-defined condition to cause tracing to stop. By contrast, the `POSIX_TRACE_UNTIL_FULL` trace stream policy requires the system to stop tracing when the trace stream is full. However, if the trace stream that is full is at least partially emptied by a call to the `posix_trace_flush()` function or by calls to the `posix_trace_getnext_event()` function, the trace system will automatically resume tracing.

If the Trace Log option is supported, the operation of the `POSIX_TRACE_FLUSH` policy is an extension of the `POSIX_TRACE_UNTIL_FULL` policy. The automatic free operation (by

125173 flushing to the associated trace log) is added.

- 125174 • If a log is associated with the trace stream and this log is a regular file, these policies also
- 125175 apply for the log. One more policy, POSIX\_TRACE\_APPEND, is defined to allow
- 125176 indefinite extension of the log. Since the log destination can be any device or pseudo-
- 125177 device, the implementation may not be able to manipulate the destination as required by
- 125178 POSIX.1-2008. For this reason, the behavior of the log full policy may be unspecified
- 125179 depending on the trace log type.

125180 The current trace interface does not define a service to preallocate space for a trace log file,

125181 because this space can be preallocated by means of a call to the *posix\_fallocate()* function.

125182 This function could be called after the file has been opened, but before the trace stream is

125183 created. The *posix\_fallocate()* function ensures that any required storage for regular file data

125184 is allocated on the file system storage media. If *posix\_fallocate()* returns successfully,

125185 subsequent writes to the specified file data will not fail due to the lack of free space on the

125186 file system storage media. Besides trace events, a trace stream also includes trace attributes

125187 and the mapping from trace event names to trace event type identifiers. The implementor

125188 is free to choose how to store the trace attributes and the trace event type map, but must

125189 ensure that this information is not lost when a trace stream overrun occurs.

### 125190 B.2.11.3 Trace Programming Examples

125191 Several programming examples are presented to show the code of the different possible

125192 subfunctions using a trace subsystem. All these programs need to include the **<trace.h>** header.

125193 In the examples shown, error checking is omitted for more simplicity.

### 125194 Trace Operation Control

125195 These examples show the creation of a trace stream for another process; one which is already

125196 trace instrumented. All the default trace stream attributes are used to simplify programming in

125197 the first example. The second example shows more possibilities.

### 125198 First Example

```

125199 /* Caution. Error checks omitted */
125200 {
125201     trace_attr_t attr;
125202     pid_t pid = traced_process_pid;
125203     int fd;
125204     trace_id_t trid;
125205
125206     - - - - -
125207     /* Initialize trace stream attributes */
125208     posix_trace_attr_init(&attr);
125209     /* Open a trace log */
125210     fd=open("/tmp/mytracelog",...);
125211     /*
125212      * Create a new trace associated with a log
125213      * and with default attributes
125214      */
125215     posix_trace_create_withlog(pid, &attr, fd, &trid);
125216     /* Trace attribute structure can now be destroyed */
125217     posix_trace_attr_destroy(&attr);
125218     /* Start of trace event recording */

```

```

125218         posix_trace_start(trid);
125219         - - - - -
125220         - - - - -
125221         /* Duration of tracing */
125222         - - - - -
125223         - - - - -
125224         /* Stop and shutdown of trace activity */
125225         posix_trace_shutdown(trid);
125226         - - - - -
125227     }

```

## Second Example

Between the initialization of the trace stream attributes and the creation of the trace stream, these trace stream attributes may be modified; see [Trace Stream Attribute Manipulation](#) (on page 3667) for a specific programming example. Between the creation and the start of the trace stream, the event filter may be set; after the trace stream is started, the event filter may be changed. The setting of an event set and the change of a filter is shown in [Create a Trace Event Type Set and Change the Trace Event Type Filter](#) (on page 3667).

```

125235     /* Caution. Error checks omitted */
125236     {
125237         trace_attr_t attr;
125238         pid_t pid = traced_process_pid;
125239         int fd;
125240         trace_id_t trid;
125241         - - - - -
125242         /* Initialize trace stream attributes */
125243         posix_trace_attr_init(&attr);
125244         /* Attr default may be changed at this place; see example */
125245         - - - - -
125246         /* Create and open a trace log with R/W user access */
125247         fd=open("/tmp/mytracelog",O_WRONLY|O_CREAT,S_IRUSR|S_IWUSR);
125248         /* Create a new trace associated with a log */
125249         posix_trace_create_withlog(pid, &attr, fd, &trid);
125250         /*
125251          * If the Trace Filter option is supported
125252          * trace event type filter default may be changed at this place;
125253          * see example about changing the trace event type filter
125254          */
125255         posix_trace_start(trid);
125256         - - - - -
125257
125258         /*
125259          * If you have an uninteresting part of the application
125260          * you can stop temporarily.
125261          *
125262          * posix_trace_stop(trid);
125263          * - - - - -
125264          * - - - - -
125265          * posix_trace_start(trid);
125266          */
125267         - - - - -
125268     }

```

```

125268     * If the Trace Filter option is supported
125269     * the current trace event type filter can be changed
125270     * at any time (see example about how to set
125271     * a trace event type filter)
125272     */
125273     - - - - -
125274     /* Stop the recording of trace events */
125275     posix_trace_stop(trid);
125276     /* Shutdown the trace stream */
125277     posix_trace_shutdown(trid);
125278     /*
125279     * Destroy trace stream attributes; attr structure may have
125280     * been used during tracing to fetch the attributes
125281     */
125282     posix_trace_attr_destroy(&attr);
125283     - - - - -
125284 }

```

## 125285 Application Instrumentation

125286 This example shows an instrumented application. The code is included in a block of instructions,  
125287 perhaps a function from a library. Possibly in an initialization part of the instrumented  
125288 application, two user trace event names are mapped to two trace event type identifiers  
125289 (function *posix\_trace\_eventid\_open()*). Then two trace points are programmed.

```

125290 /* Caution. Error checks omitted */
125291 {
125292     trace_event_id_t eventidl, eventid2;
125293     - - - - -
125294     /* Initialization of two trace event type ids */
125295     posix_trace_eventid_open("my_first_event",&eventidl);
125296     posix_trace_eventid_open("my_second_event",&eventid2);
125297     - - - - -
125298     - - - - -
125299     - - - - -
125300     /* Trace point */
125301     posix_trace_event(eventidl,NULL,0);
125302     - - - - -
125303     /* Trace point */
125304     posix_trace_event(eventid2,NULL,0);
125305     - - - - -
125306 }

```

## 125307 Trace Analyzer

125308 This example shows the manipulation of a trace log resulting from the dumping of a completed  
125309 trace stream. All the default attributes are used to simplify programming, and data associated  
125310 with a trace event is not shown in the first example. The second example shows more  
125311 possibilities.

### First Example

```

/* Caution. Error checks omitted */
{
    int fd;
    trace_id_t trid;
    posix_trace_event_info trace_event;
    char trace_event_name[TRACE_EVENT_NAME_MAX];
    int return_value;
    size_t returndatasize;
    int lost_event_number;

    - - - - -

    /* Open an existing trace log */
    fd=open("/tmp/tracelog", O_RDONLY);
    /* Open a trace stream on the open log */
    posix_trace_open(fd, &trid);
    /* Read a trace event */
    posix_trace_getnext_event(trid, &trace_event,
        NULL, 0, &returndatasize,&return_value);

    /* Read and print all trace event names out in a loop */
    while (return_value == NULL)
    {
        /*
         * Get the name of the trace event associated
         * with trid trace ID
         */
        posix_trace_eventid_get_name(trid, trace_event.event_id,
            trace_event_name);
        /* Print the trace event name out */
        printf("%s\n",trace_event_name);
        /* Read a trace event */
        posix_trace_getnext_event(trid, &trace_event,
            NULL, 0, &returndatasize,&return_value);
    }

    /* Close the trace stream */
    posix_trace_close(trid);
    /* Close the trace log */
    close(fd);
}

```

### Second Example

The complete example includes the two other examples in [Retrieve Information from a Trace Log](#) (on page 3668) and in [Retrieve the List of Trace Event Types Used in a Trace Log](#) (on page 3669). For example, the *maxdatasize* variable is set in [Retrieve the List of Trace Event Types Used in a Trace Log](#) (on page 3669).

```

/* Caution. Error checks omitted */
{
    int fd;
    trace_id_t trid;
    posix_trace_event_info trace_event;

```

```

125360     char trace_event_name[TRACE_EVENT_NAME_MAX];
125361     char * data;
125362     size_t maxdatasize=1024, returndatasize;
125363     int return_value;
125364     - - - - -
125365     /* Open an existing trace log */
125366     fd=open("/tmp/tracelog", O_RDONLY);
125367     /* Open a trace stream on the open log */
125368     posix_trace_open( fd, &trid);
125369     /*
125370      * Retrieve information about the trace stream which
125371      * was dumped in this trace log (see example)
125372      */
125373     - - - - -
125374     /* Allocate a buffer for trace event data */
125375     data=(char *)malloc(maxdatasize);
125376     /*
125377      * Retrieve the list of trace events used in this
125378      * trace log (see example)
125379      */
125380     - - - - -
125381     /* Read and print all trace event names and data out in a loop */
125382     while (1)
125383     {
125384     posix_trace_getnext_event(trid, &trace_event,
125385         data, maxdatasize, &returndatasize,&return_value);
125386         if (return_value != NULL) break;
125387         /*
125388          * Get the name of the trace event type associated
125389          * with trid trace ID
125390          */
125391         posix_trace_eventid_get_name(trid, trace_event.event_id,
125392             trace_event_name);
125393         {
125394         int i;
125395
125396         /* Print the trace event name out */
125397         printf("%s: ", trace_event_name);
125398         /* Print the trace event data out */
125399         for (i=0; i<returndatasize, i++) printf("%02.2X",
125400             (unsigned char)data[i]);
125401         printf("\n");
125402         }
125403     }
125404     /* Close the trace stream */
125405     posix_trace_close(trid);
125406     /* The buffer data is deallocated */
125407     free(data);
125408     /* Now the file can be closed */
125409     close(fd);
125409 }

```

## Several Programming Manipulations

The following examples show some typical sets of operations needed in some contexts.

### Trace Stream Attribute Manipulation

This example shows the manipulation of a trace stream attribute object in order to change the default value provided by a previous *posix\_trace\_attr\_init()* call.

```
/* Caution. Error checks omitted */
{
    trace_attr_t attr;
    size_t logsize=100000;
    - - - - -
    /* Initialize trace stream attributes */
    posix_trace_attr_init(&attr);
    /* Set the trace name in the attributes structure */
    posix_trace_attr_setname(&attr, "my_trace");
    /* Set the trace full policy */
    posix_trace_attr_setstreamfullpolicy(&attr, POSIX_TRACE_LOOP);
    /* Set the trace log size */
    posix_trace_attr_setlogsize(&attr, logsize);
    - - - - -
}
```

### Create a Trace Event Type Set and Change the Trace Event Type Filter

This example is valid only if the Trace Event Filter option is supported. This example shows the manipulation of a trace event type set in order to change the trace event type filter for an existing active trace stream, which may be just-created, running, or suspended. Some sets of trace event types are well-known, such as the set of trace event types not associated with a process, some trace event types are just-built trace event types for this trace stream; one trace event type is the predefined trace event error type which is deleted from the trace event type set.

```
/* Caution. Error checks omitted */
{
    trace_id_t trid = existing_trace;
    trace_event_set_t set;
    trace_event_id_t trace_event1, trace_event2;
    - - - - -
    /* Initialize to an empty set of trace event types */
    /* (not strictly required because posix_trace_event_set_fill() */
    /* will ignore the prior contents of the event set.) */
    posix_trace_eventset_emptyset(&set);
    /*
     * Fill the set with all system trace events
     * not associated with a process
     */
    posix_trace_eventset_fill(&set, POSIX_TRACE_WOPID_EVENTS);

    /*
     * Get the trace event type identifier of the known trace event name
     * my_first_event for the trid trace stream
     */
    posix_trace_trid_eventid_open(trid, "my_first_event", &trace_event1);
}
```

```

125457      /* Add the set with this trace event type identifier */
125458      posix_trace_eventset_add_event(trace_event1, &set);
125459      /*
125460      * Get the trace event type identifier of the known trace event name
125461      * my_second_event for the trid trace stream
125462      */
125463      posix_trace_trid_eventid_open(trid, "my_second_event", &trace_event2);
125464      /* Add the set with this trace event type identifier */
125465      posix_trace_eventset_add_event(trace_event2, &set);
125466      - - - - -
125467      /* Delete the system trace event POSIX_TRACE_ERROR from the set */
125468      posix_trace_eventset_del_event(POSIX_TRACE_ERROR, &set);
125469      - - - - -
125470
125471      /* Modify the trace stream filter making it equal to the new set */
125472      posix_trace_set_filter(trid, &set, POSIX_TRACE_SET_EVENTSET);
125473      - - - - -
125474      /*
125475      * Now trace_event1, trace_event2, and all system trace event types
125476      * not associated with a process, except for the POSIX_TRACE_ERROR
125477      * system trace event type, are filtered out of (not recorded in) the
125478      * existing trace stream.
125479      */
125480      }

```

## Retrieve Information from a Trace Log

This example shows how to extract information from a trace log, the dump of a trace stream. This code:

```

125483      • Asks if the trace stream has lost trace events
125484
125485      • Extracts the information about the version of the trace subsystem which generated this
125486      trace log
125487
125488      • Retrieves the maximum size of trace event data; this may be used to dynamically allocate
125489      an array for extracting trace event data from the trace log without overflow
125490
125491      /* Caution. Error checks omitted */
125492      {
125493          struct posix_trace_status_info statusinfo;
125494          trace_attr_t attr;
125495          trace_id_t trid = existing_trace;
125496          size_t maxdatasize;
125497          char genversion[TRACE_NAME_MAX];
125498          - - - - -
125499          /* Get the trace stream status */
125500          posix_trace_get_status(trid, &statusinfo);
125501          /* Detect an overrun condition */
125502          if (statusinfo.posix_stream_overrun_status == POSIX_TRACE_OVERRUN)
125503              printf("trace events have been lost\n");
125504
125505          /* Get attributes from the trid trace stream */
125506          posix_trace_get_attr(trid, &attr);
125507          /* Get the trace generation version from the attributes */

```



```

125504     posix_trace_attr_getgenversion(&attr, genversion);
125505     /* Print the trace generation version out */
125506     printf("Information about Trace Generator:%s\n",genversion);

125507     /* Get the trace event max data size from the attributes */
125508     posix_trace_attr_getmaxdatasize(&attr, &maxdatasize);
125509     /* Print the trace event max data size out */
125510     printf("Maximum size of associated data:%d\n",maxdatasize);
125511     /* Destroy the trace stream attributes */
125512     posix_trace_attr_destroy(&attr);
125513 }

```

### Retrieve the List of Trace Event Types Used in a Trace Log

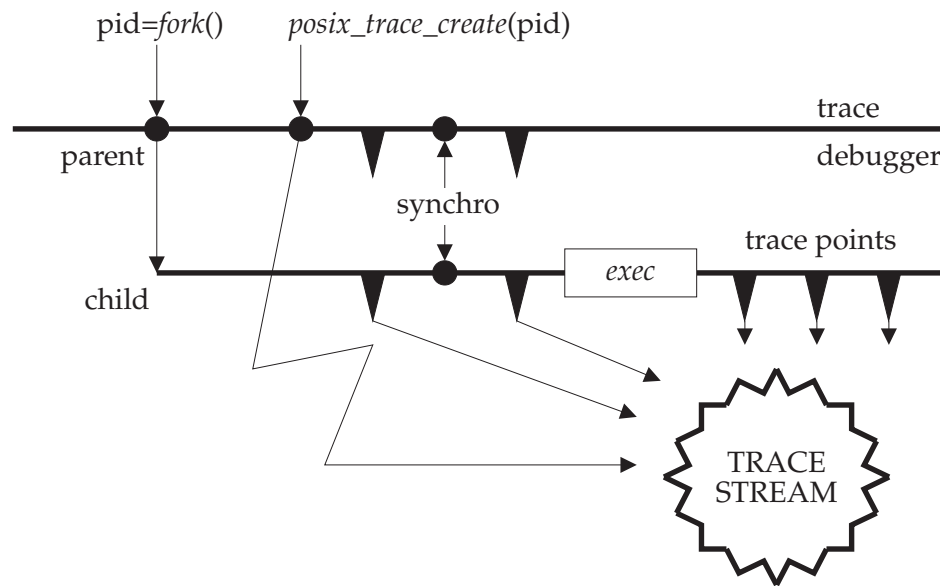
This example shows the retrieval of a trace stream's trace event type list. This operation may be very useful if you are interested only in tracking the type of trace events in a trace log.

```

125517 /* Caution. Error checks omitted */
125518 {
125519     trace_id_t trid = existing_trace;
125520     trace_event_id_t event_id;
125521     char event_name[TRACE_EVENT_NAME_MAX];
125522     int return_value;
125523     - - - - -
125524     /*
125525      * In a loop print all existing trace event names out
125526      * for the trid trace stream
125527      */
125528     while (1)
125529     {
125530         posix_trace_eventtypelist_getnext_id(trid, &event_id
125531             &return_value);
125532         if (return_value != NULL) break;
125533         /*
125534          * Get the name of the trace event associated
125535          * with trid trace ID
125536          */
125537         posix_trace_eventid_get_name(trid, event_id, event_name);
125538         /* Print the name out */
125539         printf("%s\n", event_name);
125540     }
125541 }

```

125542 B.2.11.4 Rationale on Trace for Debugging



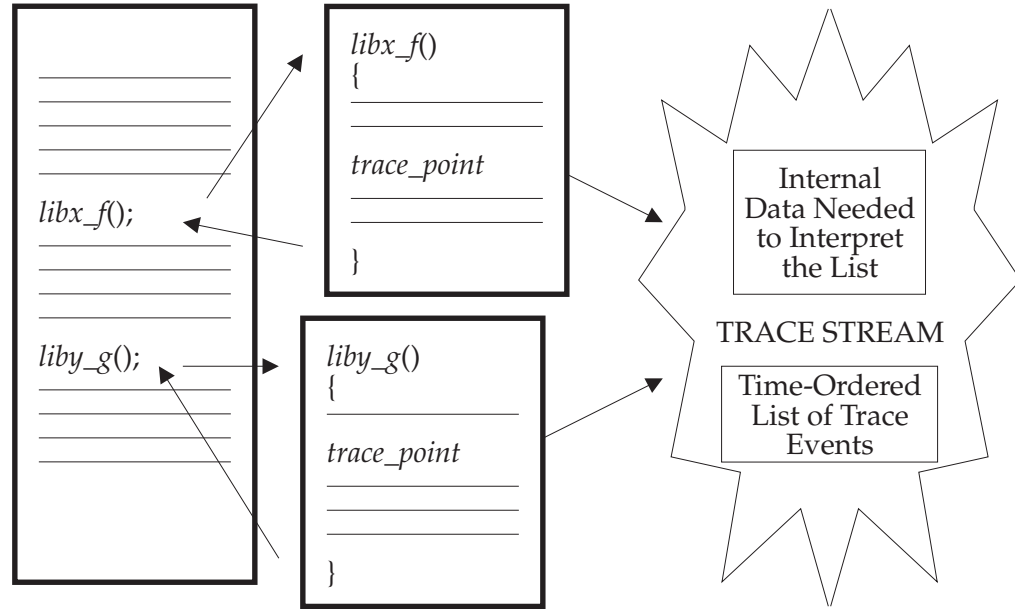
125543 **Figure B-5** Trace Another Process

125544 Among the different possibilities offered by the trace interface defined in POSIX.1-2008, the  
 125545 debugging of an application is the most interesting one. Typical operations in the controlling  
 125546 debugger process are to filter trace event types, to get trace events from the trace stream, to stop  
 125547 the trace stream when the debugged process is executing uninteresting code, to start the trace  
 125548 stream when some interesting point is reached, and so on. The interface defined in POSIX.1-2008  
 125549 should define all the necessary base functions to allow this dynamic debug handling.

125550 **Figure B-5** shows an example in which the trace stream is created after the call to the `fork()`  
 125551 function. If the user does not want to lose trace events, some synchronization mechanism  
 125552 (represented in the figure) may be needed before calling the `exec()` function, to give the parent a  
 125553 chance to create the trace stream before the child begins the execution of its trace points.

125554 B.2.11.5 Rationale on Trace Event Type Name Space

125555 At first, the working group was in favor of the representation of a trace event type by an integer  
 125556 (`event_name`). It seems that existing practice shows the weakness of such a representation. The  
 125557 collision of trace event types is the main problem that cannot be simply resolved using this sort  
 125558 of representation. Suppose, for example, that a third party designs an instrumented library. The  
 125559 user does not have the source of this library and wants to trace his application which uses in  
 125560 some part the third-party library. There is no means for him to know what are the trace event  
 125561 types used in the instrumented library so he has some chance of duplicating some of them and  
 125562 thus to obtain a contaminated tracing of his application.



**Figure B-6** Trace Name Space Overview: With Third-Party Library

There are requirements to allow program images containing pieces from various vendors to be traced without also requiring those of any other vendors to coordinate their uses of the trace facility, and especially the naming of their various trace event types and trace point IDs. The chosen solution is to provide a very large name space, large enough so that the individual vendors can give their trace types and tracepoint IDs sufficiently long and descriptive names making the occurrence of collisions quite unlikely. The probability of collision is thus made sufficiently low so that the problem may, as a practical matter, be ignored. By requirement, the consequence of collisions will be a slight ambiguity in the trace streams; tracing will continue in spite of collisions and ambiguities. “The show must go on”. The *posix\_prog\_address* member of the **posix\_trace\_event\_info** structure is used to allow trace streams to be unambiguously interpreted, despite the fact that trace event types and trace event names need not be unique.

The *posix\_trace\_eventid\_open()* function is required to allow the instrumented third-party library to get a valid trace event type identifier for its trace event names. This operation is, somehow, an allocation, and the group was aware of proposing some deallocation mechanism which the instrumented application could use to recover the resources used by a trace event type identifier. This would have given the instrumented application the benefit of being capable of reusing a possible minimum set of trace event type identifiers, but also the inconvenience to have, possibly in the same trace stream, one trace event type identifier identifying two different trace event types. After some discussions the group decided to not define such a function which would make this API thicker for little benefit, the user having always the possibility of adding identification information in the *data* member of the trace event structure.

The set of the trace event type identifiers the controlling process wants to filter out is initialized in the trace mechanism using the function *posix\_trace\_set\_filter()*, setting the arguments according to the definitions explained in *posix\_trace\_set\_filter()*. This operation can be done statically (when the trace is in the STOPPED state) or dynamically (when the trace is in the STARTED state). The preparation of the filter is normally done using the function defined in *posix\_trace\_eventtypelist\_getnext\_id()* and eventually the function *posix\_trace\_eventtypelist\_rewind()* in order to know (before the recording) the list of the potential

125592 set of trace event types that can be recorded. In the case of an active trace stream, this list may  
125593 not be exhaustive. Actually, the target process may not have yet called the function  
125594 *posix\_trace\_eventid\_open()*. But it is a common practice, for a controlling process, to prepare the  
125595 filtering of a future trace stream before its start. Therefore the user must have a way to get the  
125596 trace event type identifier corresponding to a well-known trace event name before its future  
125597 association by the pre-cited function. This is done by calling the *posix\_trace\_trid\_eventid\_open()*  
125598 function, given the trace stream identifier and the trace name, and described hereafter. Because  
125599 this trace event type identifier is associated with a trace stream identifier, where a unique  
125600 process has initialized two or more traces, the implementation is expected to return the same  
125601 trace event type identifier for successive calls to *posix\_trace\_trid\_eventid\_open()* with different  
125602 trace stream identifiers. The *posix\_trace\_eventid\_get\_name()* function is used by the controller  
125603 process to identify, by the name, the trace event type returned by a call to the  
125604 *posix\_trace\_eventtypelist\_getnext\_id()* function.

125605 Afterwards, the set of trace event types is constructed using the functions defined in  
125606 *posix\_trace\_eventset\_empty()*, *posix\_trace\_eventset\_fill()*, *posix\_trace\_eventset\_add()*, and  
125607 *posix\_trace\_eventset\_del()*.

125608 A set of functions is provided devoted to the manipulation of the trace event type identifier and  
125609 names for an active trace stream. All these functions require the trace stream identifier argument  
125610 as the first parameter. The opacity of the trace event type identifier implies that the user cannot  
125611 associate directly its well-known trace event name with the system-associated trace event type  
125612 identifier.

125613 The *posix\_trace\_trid\_eventid\_open()* function allows the application to get the system trace event  
125614 type identifier back from the system, given its well-known trace event name. This function is  
125615 useful only when a controlling process needs to specify specific events to be filtered.

125616 The *posix\_trace\_eventid\_get\_name()* function allows the application to obtain a trace event name  
125617 given its trace event type identifier. One possible use of this function is to identify the type of a  
125618 trace event retrieved from the trace stream, and print it. The easiest way to implement this  
125619 requirement, is to use a single trace event type map for all the processes whose maps are  
125620 required to be identical. A more difficult way is to attempt to keep multiple maps identical at  
125621 every call to *posix\_trace\_eventid\_open()* and *posix\_trace\_trid\_eventid\_open()*.

#### 125622 B.2.11.6 Rationale on Trace Events Type Filtering

125623 The most basic rationale for runtime and pre-registration filtering (selection/rejection) of trace  
125624 event types is to prevent choking of the trace collection facility, and/or overloading of the  
125625 computer system. Any worthwhile trace facility can bring even the largest computer to its  
125626 knees. Otherwise, everything would be recorded and filtered after the fact; it would be much  
125627 simpler, but impractical.

125628 To achieve debugging, measurement, or whatever the purpose of tracing, the filtering of trace  
125629 event types is an important part of trace analysis. Due to the fact that the trace events are put  
125630 into a trace stream and probably logged afterwards into a file, different levels of filtering—that  
125631 is, rejection of trace event types—are possible.

## **Filtering of Trace Event Types Before Tracing**

This function, represented by the `posix_trace_set_filter()` function in POSIX.1-2008 (see `posix_trace_set_filter()`), selects, before or during tracing, the set of trace event types to be filtered out. It should be possible also (as OSF suggested in their ETAP trace specifications) to select the kernel trace event types to be traced in a system-wide fashion. These two functionalities are called the pre-filtering of trace event types.

The restriction on the actual type used for the `trace_event_set_t` type is intended to guarantee that these objects can always be assigned, have their address taken, and be passed by value as parameters. It is not intended that this type be a structure including pointers to other data structures, as that could impact the portability of applications performing such operations. A reasonable implementation could be a structure containing an array of integer types.

## **Filtering of Trace Event Types at Runtime**

It is possible to build this functionality using the `posix_trace_set_filter()` function. A privileged process or a privileged thread can get trace events from the trace stream of another process or thread, and thus specify the type of trace events to record into a file, using implementation-defined methods and interfaces. This functionality, called inline filtering of trace event types, is used for runtime analysis of trace streams.

## **Post-Mortem Filtering of Trace Event Types**

The word “post-mortem” is used here to indicate that some unanticipated situation occurs during execution that does not permit a pre or inline filtering of trace events and that it is necessary to record all trace event types to have a chance to discover the problem afterwards. When the program stops, all the trace events recorded previously can be analyzed in order to find the solution. This functionality could be named the post-filtering of trace event types.

## **Discussions about Trace Event Type Filtering**

After long discussions with the parties involved in the process of defining the trace interface, it seems that the sensitivity to the filtering problem is different, but everybody agrees that the level of the overhead introduced during the tracing operation depends on the filtering method elected. If the time that it takes the trace event to be recorded can be neglected, the overhead introduced by the filtering process can be classified as follows:

Pre-filtering      System and process/thread-level overhead

Inline-filtering    Process/thread-level overhead

Post-filtering      No overhead; done offline

The pre-filtering could be named “critical realtime” filtering in the sense that the filtering of trace event type is manageable at the user level so the user can lower to a minimum the filtering overhead at some user selected level of priority for the inline filtering, or delay the filtering to after execution for the post-filtering. The counterpart of this solution is that the size of the trace stream must be sufficient to record all the trace events. The advantage of the pre-filtering is that the utilization of the trace stream is optimized.

Only pre-filtering is defined by POSIX.1-2008. However, great care must be taken in specifying pre-filtering, so that it does not impose unacceptable overhead. Moreover, it is necessary to isolate all the functionality relative to the pre-filtering.

The result of this rationale is to define a new option, the Trace Event Filter option, not necessarily implemented in small realtime systems, where system overhead is minimized to the extent possible.

#### 125676 B.2.11.7 Tracing, pthread API

125677 The objective to be able to control tracing for individual threads may be in conflict with the  
 125678 efficiency expected in threads with a *contentionscope* attribute of PTHREAD\_SCOPE\_PROCESS.  
 125679 For these threads, context switches from one thread that has tracing enabled to another thread  
 125680 that has tracing disabled may require a kernel call to inform the kernel whether it has to trace  
 125681 system events executed by that thread or not. For this reason, it was proposed that the ability to  
 125682 enable or disable tracing for PTHREAD\_SCOPE\_PROCESS threads be made optional, through  
 125683 the introduction of a Trace Scope Process option. A trace implementation which did not  
 125684 implement the Trace Scope Process option would not honor the tracing-state attribute of a thread  
 125685 with PTHREAD\_SCOPE\_PROCESS; it would, however, honor the tracing-state attribute of a  
 125686 thread with PTHREAD\_SCOPE\_SYSTEM. This proposal was rejected as:

- 125687 1. Removing desired functionality (per-thread trace control)
- 125688 2. Introducing counter-intuitive behavior for the tracing-state attribute
- 125689 3. Mixing logically orthogonal ideas (thread scheduling and thread tracing)
- 125690 [Objective 4]

125691 Finally, to solve this complex issue, this API does not provide *pthread\_gettracingstate()*,  
 125692 *pthread\_settracingstate()*, *pthread\_attr\_gettracingstate()*, and *pthread\_attr\_settracingstate()*  
 125693 interfaces. These interfaces force the thread implementation to add to the weight of the thread  
 125694 and cause a revision of the threads libraries, just to support tracing. Worse yet,  
 125695 *posix\_trace\_event()* must always test this per-thread variable even in the common case where it is  
 125696 not used at all. Per-thread tracing is easy to implement using existing interfaces where  
 125697 necessary; see the following example.

#### 125698 Example

```

125699 /* Caution. Error checks omitted */
125700 static pthread_key_t my_key;
125701 static trace_event_id_t my_event_id;
125702 static pthread_once_t my_once = PTHREAD_ONCE_INIT;

125703 void my_init(void)
125704 {
125705     (void) pthread_key_create(&my_key, NULL);
125706     (void) posix_trace_eventid_open("my", &my_event_id);
125707 }

125708 int get_trace_flag(void)
125709 {
125710     pthread_once(&my_once, my_init);
125711     return (pthread_getspecific(my_key) != NULL);
125712 }

125713 void set_trace_flag(int f)
125714 {
125715     pthread_once(&my_once, my_init);
125716     pthread_setspecific(my_key, f? &my_event_id: NULL);
125717 }

125718 fn()
125719 {
125720     if (get_trace_flag())
125721         posix_trace_event(my_event_id, ...)
125722 }
```



- 125723 The above example does not implement third-party state setting.
- 125724 Lastly, per-thread tracing works poorly for threads with PTHREAD\_SCOPE\_PROCESS  
125725 contention scope. These “library” threads have minimal interaction with the kernel and would  
125726 have to explicitly set the attributes whenever they are context switched to a new kernel thread in  
125727 order to trace system events. Such state was explicitly avoided in POSIX threads to keep  
125728 PTHREAD\_SCOPE\_PROCESS threads lightweight.
- 125729 The reason that keeping PTHREAD\_SCOPE\_PROCESS threads lightweight is important is that  
125730 such threads can be used not just for simple multi-processors but also for co-routine style  
125731 programming (such as discrete event simulation) without inventing a new threads paradigm.  
125732 Adding extra runtime cost to thread context switches will make using POSIX threads less  
125733 attractive in these situations.
- 125734 *B.2.11.8 Rationale on Triggering*
- 125735 The ability to start or stop tracing based on the occurrence of specific trace event types has been  
125736 proposed as a parallel to similar functionality appearing in logic analyzers. Such triggering, in  
125737 order to be very useful, should be based not only on the trace event type, but on trace event-  
125738 specific data, including tests of user-specified fields for matching or threshold values.
- 125739 Such a facility is unnecessary where the buffering of the stream is not a constraint, since such  
125740 checks can be performed offline during post-mortem analysis.
- 125741 For example, a large system could incorporate a daemon utility to collect the trace records from  
125742 memory buffers and spool them to secondary storage for later analysis. In the instances where  
125743 resources are truly limited, such as embedded applications, the application incorporation of  
125744 application code to test the circumstances of a trace event and call the trace point only if needed  
125745 is usually straightforward.
- 125746 For performance reasons, the *posix\_trace\_event()* function should be implemented using a macro,  
125747 so if the trace is inactive, the trace event point calls are latent code and must cost no more than a  
125748 scalar test.
- 125749 The API proposed in POSIX.1-2008 does not include any triggering functionality.
- 125750 *B.2.11.9 Rationale on Timestamp Clock*
- 125751 It has been suggested that the tracing mechanism should include the possibility of specifying the  
125752 clock to be used in timestamping the trace events. When application trace events must be  
125753 correlated to remote trace events, such a facility could provide a global time reference not  
125754 available from a local clock. Further, the application may be driven by timers based on a clock  
125755 different from that used for the timestamp, and the correlation of the trace to those untraced  
125756 timer activities could be an important part of the analysis of the application.
- 125757 However, the tracing mechanism needs to be fast and just the provision of such an option can  
125758 materially affect its performance. Leaving aside the performance costs of reading some clocks,  
125759 this notion is also ill-defined when kernel trace events are to be traced by two applications  
125760 making use of different tracing clocks. This can even happen within a single application where  
125761 different parts of the application are served by different clocks. Another complication can occur  
125762 when a clock is maintained strictly at the user level and is unavailable at the kernel level.
- 125763 It is felt that the benefits of a selectable trace clock do not match its costs. Applications that wish  
125764 to correlate clocks other than the default tracing clock can include trace events with sample  
125765 values of those other clocks, allowing correlation of timestamps from the various independent  
125766 clocks. In any case, such a technique would be required when applications are sensitive to

125767 multiple clocks.

125768 *B.2.11.10 Rationale on Different Overrun Conditions*

125769 The analysis of the dynamic behavior of the trace mechanism shows that different overrun  
125770 conditions may occur. The API must provide a means to manage such conditions in a portable  
125771 way.

125772 **Overrun in Trace Streams Initialized with POSIX\_TRACE\_LOOP Policy**

125773 In this case, the user of the trace mechanism is interested in using the trace stream with  
125774 POSIX\_TRACE\_LOOP policy to record trace events continuously, but ideally without losing any  
125775 trace events. The online analyzer process must get the trace events at a mean speed equivalent to  
125776 the recording speed. Should the trace stream become full, a trace stream overrun occurs. This  
125777 condition is detected by getting the status of the active trace stream (function  
125778 *posix\_trace\_get\_status()*) and looking at the member *posix\_stream\_overrun\_status* of the read  
125779 **posix\_stream\_status** structure. In addition, two predefined trace event types are defined:

- 125780 1. The beginning of a trace overflow, to locate the beginning of an overflow when reading a  
125781 trace stream
- 125782 2. The end of a trace overflow, to locate the end of an overflow, when reading a trace stream

125783 As a timestamp is associated with these predefined trace events, it is possible to know the  
125784 duration of the overflow.

125785 **Overrun in Dumping Trace Streams into Trace Logs**

125786 The user lets the trace mechanism dump the trace stream initialized with  
125787 POSIX\_TRACE\_FLUSH policy automatically into a trace log. If the dump operation is slower  
125788 than the recording of trace events, the trace stream can overrun. This condition is detected by  
125789 getting the status of the active trace stream (the *posix\_trace\_get\_status()* function) and looking at  
125790 the member *posix\_stream\_overrun\_status* of the read **posix\_stream\_status** structure. This overrun  
125791 indicates that the trace mechanism is not able to operate in this mode at this speed. It is the  
125792 responsibility of the user to modify one of the trace parameters (the stream size or the trace  
125793 event type filter, for instance) to avoid such overrun conditions, if overruns are to be prevented.  
125794 The same already predefined trace event types (see [Overrun in Trace Streams Initialized with  
125795 POSIX\\_TRACE\\_LOOP Policy](#)) are used to detect and to know the duration of an overflow.

125796 **Reading an Active Trace Stream**

125797 Although this trace API allows one to read an active trace stream with log while it is tracing, this  
125798 feature can lead to false overflow origin interpretation: the trace log or the reader of the trace  
125799 stream. Reading from an active trace stream with log is thus non-portable, and has been left  
125800 unspecified.



## 125801 B.2.12 Data Types

### 125802 B.2.12.1 Defined Types

125803 The requirement that additional types defined in this section end in “\_t” was prompted by the  
125804 problem of name space pollution. It is difficult to define a type (where that type is not one  
125805 defined by POSIX.1-2008) in one header file and use it in another without adding symbols to the  
125806 name space of the program. To allow implementors to provide their own types, all conforming  
125807 applications are required to avoid symbols ending in “\_t”, which permits the implementor to  
125808 provide additional types. Because a major use of types is in the definition of structure members,  
125809 which can (and in many cases must) be added to the structures defined in POSIX.1-2008, the  
125810 need for additional types is compelling.

125811 The types, such as **ushort** and **ulong**, which are in common usage, are not defined in  
125812 POSIX.1-2008 (although **ushort\_t** would be permitted as an extension). They can be added to  
125813 **<sys/types.h>** using a feature test macro (see [Section B.2.2.1](#), on page 3556). A suggested symbol  
125814 for these is **\_SYSIII**. Similarly, the types like **u\_short** would probably be best controlled by **\_BSD**.

125815 Some of these symbols may appear in other headers; see [Section B.2.2.2](#) (on page 3557).

125816 **dev\_t** This type may be made large enough to accommodate host-locality considerations  
125817 of networked systems.

125818 This type must be arithmetic. Earlier proposals allowed this to be non-arithmetic  
125819 (such as a structure) and provided a *samefile()* function for comparison.

125820 **gid\_t** Some implementations had separated **gid\_t** from **uid\_t** before POSIX.1 was  
125821 completed. It would be difficult for them to coalesce them when it was  
125822 unnecessary. Additionally, it is quite possible that user IDs might be different from  
125823 group IDs because the user ID might wish to span a heterogeneous network,  
125824 where the group ID might not.

125825 For current implementations, the cost of having a separate **gid\_t** will be only  
125826 lexical.

125827 **mode\_t** This type was chosen so that implementations could choose the appropriate  
125828 integer type, and for compatibility with the ISO C standard. 4.3 BSD uses  
125829 **unsigned short** and the SVID uses **ushort**, which is the same. Historically, only the  
125830 low-order sixteen bits are significant.

125831 **nlink\_t** This type was introduced in place of **short** for *st\_nlink* (see the **<sys/stat.h>** header)  
125832 in response to an objection that **short** was too small.

125833 **off\_t** This type is used to represent a file offset or file size. On systems supporting large  
125834 files, **off\_t** is larger than 32 bits in at least one programming environment. Other  
125835 programming environments may use different sizes for **off\_t**, for compatibility or  
125836 other reasons.

125837 **pid\_t** The inclusion of this symbol was controversial because it is tied to the issue of the  
125838 representation of a process ID as a number. From the point of view of a  
125839 conforming application, process IDs should be “magic cookies”<sup>8</sup> that are produced  
125840 by calls such as *fork()*, used by calls such as *waitpid()* or *kill()*, and not otherwise

---

125841 8. An historical term meaning: “An opaque object, or token, of determinate size, whose significance is known only to the entity which  
125842 created it. An entity receiving such a token from the generating entity may only make such use of the ‘cookie’ as is defined and permitted  
125843 by the supplying entity.”

125844 analyzed (except that the sign is used as a flag for certain operations).

125845 The concept of a {PID\_MAX} value interacted with this in early proposals. Treating  
 125846 process IDs as an opaque type both removes the requirement for {PID\_MAX} and  
 125847 allows systems to be more flexible in providing process IDs that span a large range  
 125848 of values, or a small one.

125849 Since the values in **uid\_t**, **gid\_t**, and **pid\_t** will be numbers generally, and  
 125850 potentially both large in magnitude and sparse, applications that are based on  
 125851 arrays of objects of this type are unlikely to be fully portable in any case. Solutions  
 125852 that treat them as magic cookies will be portable.

125853 {CHILD\_MAX} precludes the possibility of a “toy implementation”, where there  
 125854 would only be one process.

125855 **ssize\_t** This is intended to be a signed analog of **size\_t**. The wording is such that an  
 125856 implementation may either choose to use a longer type or simply to use the signed  
 125857 version of the type that underlies **size\_t**. All functions that return **ssize\_t** (*read()*  
 125858 and *write()*) describe as “implementation-defined” the result of an input exceeding  
 125859 {SSIZE\_MAX}. It is recognized that some implementations might have **ints** that  
 125860 are smaller than **size\_t**. A conforming application would be constrained not to  
 125861 perform I/O in pieces larger than {SSIZE\_MAX}, but a conforming application  
 125862 using extensions would be able to use the full range if the implementation  
 125863 provided an extended range, while still having a single type-compatible interface.

125864 The symbols **size\_t** and **ssize\_t** are also required in **<unistd.h>** to minimize the  
 125865 changes needed for calls to *read()* and *write()*. Implementors are reminded that it  
 125866 must be possible to include both **<sys/types.h>** and **<unistd.h>** in the same  
 125867 program (in either order) without error.

125868 **uid\_t** Before the addition of this type, the data types used to represent these values  
 125869 varied throughout early proposals. The **<sys/stat.h>** header defined these values  
 125870 as type **short**, the **<passwd.h>** file (now **<pwd.h>** and **<grp.h>**) used an **int**, and  
 125871 *getuid()* returned an **int**. In response to a strong objection to the inconsistent  
 125872 definitions, all the types were switched to **uid\_t**.

125873 In practice, those historical implementations that use varying types of this sort can  
 125874 typedef **uid\_t** to **short** with no serious consequences.

125875 The problem associated with this change concerns object compatibility after  
 125876 structure size changes. Since most implementations will define **uid\_t** as a short, the  
 125877 only substantive change will be a reduction in the size of the **passwd** structure.  
 125878 Consequently, implementations with an overriding concern for object  
 125879 compatibility can pad the structure back to its current size. For that reason, this  
 125880 problem was not considered critical enough to warrant the addition of a separate  
 125881 type to POSIX.1.

125882 The types **uid\_t** and **gid\_t** are magic cookies. There is no {UID\_MAX} defined by  
 125883 POSIX.1, and no structure imposed on **uid\_t** and **gid\_t** other than that they be  
 125884 positive arithmetic types. (In fact, they could be **unsigned char**.) There is no  
 125885 maximum or minimum specified for the number of distinct user or group IDs.

125886 B.2.12.2 *The char Type*

125887 POSIX.1-2008 explicitly requires that a **char** type is exactly one byte (8 bits).

125888 **B.2.13 Status Information**

125889 POSIX.1-2008 does not require all matching WNOWAIT threads (threads in a matching call to  
 125890 *waitid()* with the WNOWAIT flag set) to obtain a child's status information because the status  
 125891 information might be discarded (consumed or replaced) before one of the matching WNOWAIT  
 125892 threads is scheduled. If the status information is not discarded, it will remain available, so all of  
 125893 the matching WNOWAIT threads will (eventually) obtain the status information.

125894 **B.2.14 File Descriptor Allocation**

125895 Functions such as *pipe()* and *socketpair()* which allocate two file descriptors are permitted to  
 125896 perform the two allocations independently. This means that other threads or signal handlers  
 125897 may perform operations on file descriptors in between the two allocations and this can result in  
 125898 the two file descriptors not having adjacent values or in the second allocation producing a lower  
 125899 value than the first.

125900 **B.3 System Interfaces**

125901 See the RATIONALE sections on the individual reference pages.

125902 **B.3.1 System Interfaces Removed in this Version**

125903 The following section contains a list of the interfaces removed in POSIX.1-2008, together with  
 125904 advice for application developers on the alternative interfaces that should be used for maximum  
 125905 portability.

125906 B.3.1.1 *bcmp()*

125907 Applications are recommended to use the *memcmp()* function instead of this function.  
 125908 For maximum portability, it is recommended to replace the function call to *bcmp()* as follows:  
 125909 `#define bcmp(b1,b2,len) memcmp((b1), (b2), (size_t)(len))`

125910 B.3.1.2 *bcopy()*

125911 Applications are recommended to use the *memmove()* function instead of this function.  
 125912 The following are approximately equivalent (note the order of the arguments):  
 125913 `bcopy(s1,s2,n) ≈ memmove(s2,s1,n)`  
 125914 For maximum portability, it is recommended to replace the function call to *bcopy()* as follows:  
 125915 `#define bcopy(b1,b2,len) (void)(memmove((b2), (b1), (len)))`

- 125916 B.3.1.3 *bsd\_signal()*
- 125917 Applications are recommended to use the *sigaction()* function instead of this function.
- 125918 The *bsd\_signal()* function was supplied as a migration path for the BSD *signal()* function for  
125919 simple applications that installed a single-argument signal handler function.
- 125920 Historically, the *bsd\_signal()* function differs from *signal()* in that the SA\_RESTART flag is set  
125921 and the SA\_RESETHAND flag is clear when *bsd\_signal()* is used. The state of these flags is not  
125922 specified for *signal()*.
- 125923 B.3.1.4 *bzero()*
- 125924 Applications are recommended to use the *memset()* function instead of this function.
- 125925 For maximum portability, it is recommended to replace the function call to *bzero()* as follows:
- 125926 `#define bzero(b,len) (void)(memset((b), '\0', (len)))`
- 125927 B.3.1.5 *ecvt()*, *fcvt()*, *gcvt()*
- 125928 Applications are recommended to use the *sprintf()* function instead of these functions.
- 125929 The *sprintf()* function is required by ISO C and is thus more portable.
- 125930 B.3.1.6 *ftime()*
- 125931 Applications are recommended to use the *time()* function to determine the current time.  
125932 Realtime applications should use *clock\_gettime()* to determine the current time.
- 125933 B.3.1.7 *getcontext()*, *makecontext()*, *swapcontext()*
- 125934 Due to portability issues with these functions, especially with the manipulation of contexts,  
125935 applications are recommended to be rewritten to use POSIX threads.
- 125936 B.3.1.8 *gethostbyaddr()*, *gethostbyname()*
- 125937 Applications are recommended to use the *getaddrinfo()* and *getnameinfo()* functions instead of  
125938 these functions.
- 125939 The *gethostbyaddr()* and *gethostbyname()* functions may return pointers to static data, which may  
125940 be overwritten by subsequent calls to any of these functions. The suggested replacements do not  
125941 have this problem and are also IPv6-capable.
- 125942 B.3.1.9 *getwd()*
- 125943 Applications are recommended to use the *getcwd()* function to determine the current working  
125944 directory.

125945 B.3.1.10 *h\_errno*

125946 Applications are recommended not to use this error return code. Previously it was set by the  
 125947 *gethostbyaddr()* and *gethostbyname()* functions.

125948 B.3.1.11 *index()*

125949 Applications are recommended to use the *strchr()* function instead of this function.

125950 For maximum portability, it is recommended to replace the function call to *index()* as follows:

125951 `#define index(a,b) strchr((a),(b))`

125952 B.3.1.12 *makecontext()*

125953 Applications using the *getcontext()*, *makecontext()*, and *swapcontext()* functions should be  
 125954 rewritten to use POSIX threads.

125955 B.3.1.13 *mktemp()*

125956 Applications are recommended to use the *mkstemp()* function instead of this function.

125957 The *mktemp()* function makes an application vulnerable to possible security problems since  
 125958 between the time a pathname is created and the file opened, it is possible for some other process  
 125959 to create a file with the same name. The *mkstemp()* function does not have this vulnerability.

125960 B.3.1.14 *pthread\_attr\_getstackaddr()*, *pthread\_attr\_setstackaddr()*

125961 Applications are recommended to use the *pthread\_attr\_setstack()* and *pthread\_attr\_getstack()*  
 125962 functions instead of these functions.

125963 There are a number of ambiguities in the specification of the *stackaddr* attribute that makes  
 125964 portable use of these interfaces impossible.

125965 B.3.1.15 *rindex()*

125966 Applications are recommended to use the *strrchr()* function instead of this function.

125967 For maximum portability, it is recommended to replace the function call to *rindex()* as follows:

125968 `#define rindex(a,b) strrchr((a),(b))`

125969 B.3.1.16 *scalb()*

125970 Applications are recommended to use either *scalbln()*, *scalblnf()*, or *scalblnl()* instead of these  
 125971 functions.

125972 The behavior for the *scalb()* function was only defined when the *n* argument is an integer, a  
 125973 NaN, or Inf. The behavior of other values for the *n* argument was unspecified.

125974 B.3.1.17 *ualarm()*

125975 Applications are recommended to use *timer\_create()*, *timer\_delete()*, *timer\_getoverrun()*,  
125976 *timer\_gettime()*, or *timer\_settime()* instead of this function.

125977 B.3.1.18 *usleep()*

125978 Applications are recommended to use the *nanosleep()* function instead of this function.

125979 B.3.1.19 *vfork()*

125980 Applications are recommended to use the *fork()* function instead of this function.

125981 The *vfork()* function was previously under-specified.

125982 B.3.1.20 *wcswcs()*

125983 Applications are recommended to use the *wcsstr()* function instead of this function.

125984 The *wcsstr()* function is technically equivalent and is portable across all ISO C implementations.

## 125985 B.3.2 System Interfaces Removed in the Previous Version

125986 The following system interfaces, headers, and external variables were removed in the previous  
125987 version of this standard:

125988	<i>advance()</i>	<i>getdtablesize()</i>	<i>re_exec()</i>	<i>ttyslot()</i>	<i>loc1</i>
125989	<i>brk()</i>	<i>getpagesize()</i>	<i>regcmp()</i>	<i>valloc()</i>	<i>__loc1</i>
125990	<i>chroot()</i>	<i>getpass()</i>	<i>regex()</i>	<i>wait3()</i>	<i>loc2</i>
125991	<i>compile()</i>	<i>getw()</i>	<i>sbrk()</i>	<b>&lt;re_comp.h&gt;</b>	<i>locs</i>
125992	<i>cuserid()</i>	<i>putw()</i>	<i>sigstack()</i>	<b>&lt;regexp.h&gt;</b>	
125993	<i>gamma()</i>	<i>re_comp()</i>	<i>step()</i>	<b>&lt;varargs.h&gt;</b>	

## 125994 B.3.3 Examples for Spawn

125995 The following long examples are provided in the Rationale (Informative) volume of  
125996 POSIX.1-2008 as a supplement to the reference page for *posix\_spawn()*.

### 125997 Example Library Implementation of Spawn

125998 The *posix\_spawn()* or *posix\_spawnnp()* functions provide the following:

- 125999 • Simply start a process executing a process image. This is the simplest application for  
126000 process creation, and it may cover most executions of *fork()*.
- 126001 • Support I/O redirection, including pipes.
- 126002 • Run the child under a user and group ID in the domain of the parent.
- 126003 • Run the child at any priority in the domain of the parent.

126004 The *posix\_spawn()* or *posix\_spawnnp()* functions do not cover every possible use of the *fork()*  
126005 function, but they do span the common applications: typical use by a shell and a login utility.

126006 The price for an application is that before it calls *posix\_spawn()* or *posix\_spawnnp()*, the parent  
126007 must adjust to a state that *posix\_spawn()* or *posix\_spawnnp()* can map to the desired state for the

126008 child. Environment changes require the parent to save some of its state and restore it afterwards.  
 126009 The effective behavior of a successful invocation of *posix\_spawn()* is as if the operation were  
 126010 implemented with POSIX operations as follows:

```

126011 #include <sys/types.h>
126012 #include <stdlib.h>
126013 #include <stdio.h>
126014 #include <unistd.h>
126015 #include <sched.h>
126016 #include <fcntl.h>
126017 #include <signal.h>
126018 #include <errno.h>
126019 #include <string.h>
126020 #include <signal.h>

126021 /* #include <spawn.h> */
126022 /*****
126023  * Things that could be defined in spawn.h */
126024  *****/
126025 typedef struct
126026 {
126027     short posix_attr_flags;
126028     #define POSIX_SPAWN_SETPGROUP      0x1
126029     #define POSIX_SPAWN_SETSIGMASK    0x2
126030     #define POSIX_SPAWN_SETSIGDEF     0x4
126031     #define POSIX_SPAWN_SETSCHEDULER  0x8
126032     #define POSIX_SPAWN_SETSCHEDPARAM 0x10
126033     #define POSIX_SPAWN_RESETIDS      0x20
126034     pid_t posix_attr_pgroup;
126035     sigset_t posix_attr_sigmask;
126036     sigset_t posix_attr_sigdefault;
126037     int posix_attr_schedpolicy;
126038     struct sched_param posix_attr_schedparam;
126039 } posix_spawnattr_t;

126040 typedef char *posix_spawn_file_actions_t;

126041 int posix_spawn_file_actions_init(
126042     posix_spawn_file_actions_t *file_actions);
126043 int posix_spawn_file_actions_destroy(
126044     posix_spawn_file_actions_t *file_actions);
126045 int posix_spawn_file_actions_addclose(
126046     posix_spawn_file_actions_t *file_actions, int fildes);
126047 int posix_spawn_file_actions_adddup2(
126048     posix_spawn_file_actions_t *file_actions, int fildes,
126049     int newfildes);
126050 int posix_spawn_file_actions_addopen(
126051     posix_spawn_file_actions_t *file_actions, int fildes,
126052     const char *path, int oflag, mode_t mode);
126053 int posix_spawnattr_init(posix_spawnattr_t *attr);
126054 int posix_spawnattr_destroy(posix_spawnattr_t *attr);
126055 int posix_spawnattr_getflags(const posix_spawnattr_t *attr,
126056     short *lags);
126057 int posix_spawnattr_setflags(posix_spawnattr_t *attr, short flags);
126058 int posix_spawnattr_getpgroup(const posix_spawnattr_t *attr,
```



```

126059     pid_t *pgroup);
126060 int posix_spawnattr_setpgroup(posix_spawnattr_t *attr, pid_t pgroup);
126061 int posix_spawnattr_getschedpolicy(const posix_spawnattr_t *attr,
126062     int *schedpolicy);
126063 int posix_spawnattr_setschedpolicy(posix_spawnattr_t *attr,
126064     int schedpolicy);
126065 int posix_spawnattr_getschedparam(const posix_spawnattr_t *attr,
126066     struct sched_param *schedparam);
126067 int posix_spawnattr_setschedparam(posix_spawnattr_t *attr,
126068     const struct sched_param *schedparam);
126069 int posix_spawnattr_getsigmask(const posix_spawnattr_t *attr,
126070     sigset_t *sigmask);
126071 int posix_spawnattr_setsigmask(posix_spawnattr_t *attr,
126072     const sigset_t *sigmask);
126073 int posix_spawnattr_getdefault(const posix_spawnattr_t *attr,
126074     sigset_t *sigdefault);
126075 int posix_spawnattr_setsigdefault(posix_spawnattr_t *attr,
126076     const sigset_t *sigdefault);
126077 int posix_spawn(pid_t *pid, const char *path,
126078     const posix_spawn_file_actions_t *file_actions,
126079     const posix_spawnattr_t *attrp, char *const argv[],
126080     char *const envp[]);
126081 int posix_spawnnp(pid_t *pid, const char *file,
126082     const posix_spawn_file_actions_t *file_actions,
126083     const posix_spawnattr_t *attrp, char *const argv[],
126084     char *const envp[]);

126085 /*****
126086  /* Example posix_spawn() library routine */
126087  *****/
126088 int posix_spawn(pid_t *pid,
126089     const char *path,
126090     const posix_spawn_file_actions_t *file_actions,
126091     const posix_spawnattr_t *attrp,
126092     char *const argv[],
126093     char *const envp[])
126094 {
126095     /* Create process */
126096     if ((*pid = fork()) == (pid_t) 0)
126097     {
126098         /* This is the child process */
126099         /* Worry about process group */
126100         if (attrp->posix_attr_flags & POSIX_SPAWN_SETPGROUP)
126101         {
126102             /* Override inherited process group */
126103             if (setpgid(0, attrp->posix_attr_pgroup) != 0)
126104             {
126105                 /* Failed */
126106                 exit(127);
126107             }
126108         }

126109         /* Worry about thread signal mask */
126110         if (attrp->posix_attr_flags & POSIX_SPAWN_SETSIGMASK)

```



```

126111     {
126112         /* Set the signal mask (can't fail) */
126113         sigprocmask(SIG_SETMASK, &attrp->posix_attr_sigmask, NULL);
126114     }
126115     /* Worry about resetting effective user and group IDs */
126116     if (attrp->posix_attr_flags & POSIX_SPAWN_RESETPIDS)
126117     {
126118         /* None of these can fail for this case. */
126119         setuid(getuid());
126120         setgid(getgid());
126121     }
126122     /* Worry about defaulted signals */
126123     if (attrp->posix_attr_flags & POSIX_SPAWN_SETSIGDEF)
126124     {
126125         struct sigaction deflt;
126126         sigset_t all_signals;
126127
126128         int s;
126129
126130         /* Construct default signal action */
126131         deflt.sa_handler = SIG_DFL;
126132         deflt.sa_flags = 0;
126133
126134         /* Construct the set of all signals */
126135         sigfillset(&all_signals);
126136
126137         /* Loop for all signals */
126138         for (s = 0; sigismember(&all_signals, s); s++)
126139         {
126140             /* Signal to be defaulted? */
126141             if (sigismember(&attrp->posix_attr_sigdefault, s))
126142             {
126143                 /* Yes; default this signal */
126144                 if (sigaction(s, &deflt, NULL) == -1)
126145                 {
126146                     /* Failed */
126147                     exit(127);
126148                 }
126149             }
126150         }
126151     }
126152     /* Worry about the fds if they are to be mapped */
126153     if (file_actions != NULL)
126154     {
126155         /* Loop for all actions in object file_actions */
126156         /* (implementation dives beneath abstraction) */
126157         char *p = *file_actions;
126158
126159         while (*p != '\0')
126160         {
126161             if (strncmp(p, "close(", 6) == 0)
126162             {
126163                 int fd;

```

```

126159         if (sscanf(p + 6, "%d", &fd) != 1)
126160         {
126161             exit(127);
126162         }
126163         if (close(fd) == -1)
126164             exit(127);
126165     }
126166     else if (strncmp(p, "dup2(", 5) == 0)
126167     {
126168         int fd, newfd;
126169
126169         if (sscanf(p + 5, "%d,%d", &fd, &newfd) != 2)
126170         {
126171             exit(127);
126172         }
126173         if (dup2(fd, newfd) == -1)
126174             exit(127);
126175     }
126176     else if (strncmp(p, "open(", 5) == 0)
126177     {
126178         int fd, oflag;
126179         mode_t mode;
126180         int tempfd;
126181         char path[1000];    /* Should be dynamic */
126182         char *q;
126183
126183         if (sscanf(p + 5, "%d", &fd) != 1)
126184         {
126185             exit(127);
126186         }
126187         p = strchr(p, ',') + 1;
126188         q = strchr(p, '*');
126189         if (q == NULL)
126190             exit(127);
126191         strncpy(path, p, q - p);
126192         path[q - p] = '\0';
126193         if (sscanf(q + 1, "%o,%o", &oflag, &mode) != 2)
126194         {
126195             exit(127);
126196         }
126197         if (close(fd) == -1)
126198         {
126199             if (errno != EBADF)
126200                 exit(127);
126201         }
126202         tempfd = open(path, oflag, mode);
126203         if (tempfd == -1)
126204             exit(127);
126205         if (tempfd != fd)
126206         {
126207             if (dup2(tempfd, fd) == -1)
126208             {
126209                 exit(127);
126210             }

```

```

126211         if (close(tempfd) == -1)
126212         {
126213             exit(127);
126214         }
126215     }
126216 }
126217 else
126218 {
126219     exit(127);
126220 }
126221 p = strchr(p, ')') + 1;
126222 }
126223 }
126224 /* Worry about setting new scheduling policy and parameters */
126225 if (attrp->posix_attr_flags & POSIX_SPAWN_SETSCHEDULER)
126226 {
126227     if (sched_setscheduler(0, attrp->posix_attr_schedpolicy,
126228         &attrp->posix_attr_schedparam) == -1)
126229     {
126230         exit(127);
126231     }
126232 }
126233 /* Worry about setting only new scheduling parameters */
126234 if (attrp->posix_attr_flags & POSIX_SPAWN_SETSCHEDPARAM)
126235 {
126236     if (sched_setparam(0, &attrp->posix_attr_schedparam) == -1)
126237     {
126238         exit(127);
126239     }
126240 }
126241 /* Now execute the program at path */
126242 /* Any fd that still has FD_CLOEXEC set will be closed */
126243 execve(path, argv, envp);
126244 exit(127);          /* exec failed */
126245 }
126246 else
126247 {
126248     /* This is the parent (calling) process */
126249     if (*pid == (pid_t) - 1)
126250         return errno;
126251     return 0;
126252 }
126253 }
126254
126255 /* *****
126256 /* Here is a crude but effective implementation of the */
126257 /* file action object operators which store actions as */
126258 /* concatenated token-separated strings.          */
126259 /* *****
126259 /* Create object with no actions. */
126260 int posix_spawn_file_actions_init(
126261     posix_spawn_file_actions_t *file_actions)

```

```
126262     {
126263         *file_actions = malloc(sizeof(char));
126264         if (*file_actions == NULL)
126265             return ENOMEM;
126266         strcpy(*file_actions, "");
126267         return 0;
126268     }
126269
126270     /* Free object storage and make invalid. */
126271     int posix_spawn_file_actions_destroy(
126272         posix_spawn_file_actions_t *file_actions)
126273     {
126274         free(*file_actions);
126275         *file_actions = NULL;
126276         return 0;
126277     }
126278
126279     /* Add a new action string to object. */
126280     static int add_to_file_actions(
126281         posix_spawn_file_actions_t *file_actions, char *new_action)
126282     {
126283         *file_actions = realloc
126284             (*file_actions, strlen(*file_actions) + strlen(new_action) + 1);
126285         if (*file_actions == NULL)
126286             return ENOMEM;
126287         strcat(*file_actions, new_action);
126288         return 0;
126289     }
126290
126291     /* Add a close action to object. */
126292     int posix_spawn_file_actions_addclose(
126293         posix_spawn_file_actions_t *file_actions, int fildes)
126294     {
126295         char temp[100];
126296
126297         sprintf(temp, "close(%d)", fildes);
126298         return add_to_file_actions(file_actions, temp);
126299     }
126300
126301     /* Add a dup2 action to object. */
126302     int posix_spawn_file_actions_adddup2(
126303         posix_spawn_file_actions_t *file_actions, int fildes,
126304         int newfildes)
126305     {
126306         char temp[100];
126307
126308         sprintf(temp, "dup2(%d,%d)", fildes, newfildes);
126309         return add_to_file_actions(file_actions, temp);
126310     }
126311
126312     /* Add an open action to object. */
126313     int posix_spawn_file_actions_addopen(
126314         posix_spawn_file_actions_t *file_actions, int fildes,
126315         const char *path, int oflag, mode_t mode)
126316     {
126317         char temp[100];
```

```

126311         sprintf(temp, "open(%d,%s*%o,%o)", fildes, path, oflag, mode);
126312         return add_to_file_actions(file_actions, temp);
126313     }

126314     /*****
126315     /* Here is a crude but effective implementation of the */
126316     /* spawn attributes object functions which manipulate */
126317     /* the individual attributes. */
126318     *****/
126319     /* Initialize object with default values. */
126320     int posix_spawnattr_init(posix_spawnattr_t *attr)
126321     {
126322         attr->posix_attr_flags = 0;
126323         attr->posix_attr_pgroup = 0;
126324         /* Default value of signal mask is the parent's signal mask; */
126325         /* other values are also allowed */
126326         sigprocmask(0, NULL, &attr->posix_attr_sigmask);
126327         sigemptyset(&attr->posix_attr_sigdefault);
126328         /* Default values of scheduling attr inherited from the parent; */
126329         /* other values are also allowed */
126330         attr->posix_attr_schedpolicy = sched_getscheduler(0);
126331         sched_getparam(0, &attr->posix_attr_schedparam);
126332         return 0;
126333     }

126334     int posix_spawnattr_destroy(posix_spawnattr_t *attr)
126335     {
126336         /* No action needed */
126337         return 0;
126338     }

126339     int posix_spawnattr_getflags(const posix_spawnattr_t *attr,
126340                                short *flags)
126341     {
126342         *flags = attr->posix_attr_flags;
126343         return 0;
126344     }
126345     int posix_spawnattr_setflags(posix_spawnattr_t *attr, short flags)
126346     {
126347         attr->posix_attr_flags = flags;
126348         return 0;
126349     }

126350     int posix_spawnattr_getpgroup(const posix_spawnattr_t *attr,
126351                                pid_t *pgroup)
126352     {
126353         *pgroup = attr->posix_attr_pgroup;
126354         return 0;
126355     }

126356     int posix_spawnattr_setpgroup(posix_spawnattr_t *attr, pid_t pgroup)
126357     {
126358         attr->posix_attr_pgroup = pgroup;
126359         return 0;
126360     }

```

```
126361     int posix_spawnattr_getschedpolicy(const posix_spawnattr_t *attr,
126362         int *schedpolicy)
126363     {
126364         *schedpolicy = attr->posix_attr_schedpolicy;
126365         return 0;
126366     }
126367     int posix_spawnattr_setschedpolicy(posix_spawnattr_t *attr,
126368         int schedpolicy)
126369     {
126370         attr->posix_attr_schedpolicy = schedpolicy;
126371         return 0;
126372     }
126373     int posix_spawnattr_getschedparam(const posix_spawnattr_t *attr,
126374         struct sched_param *schedparam)
126375     {
126376         *schedparam = attr->posix_attr_schedparam;
126377         return 0;
126378     }
126379     int posix_spawnattr_setschedparam(posix_spawnattr_t *attr,
126380         const struct sched_param *schedparam)
126381     {
126382         attr->posix_attr_schedparam = *schedparam;
126383         return 0;
126384     }
126385     int posix_spawnattr_getsigmask(const posix_spawnattr_t *attr,
126386         sigset_t *sigmask)
126387     {
126388         *sigmask = attr->posix_attr_sigmask;
126389         return 0;
126390     }
126391     int posix_spawnattr_setsigmask(posix_spawnattr_t *attr,
126392         const sigset_t *sigmask)
126393     {
126394         attr->posix_attr_sigmask = *sigmask;
126395         return 0;
126396     }
126397     int posix_spawnattr_getsigdefault(const posix_spawnattr_t *attr,
126398         sigset_t *sigdefault)
126399     {
126400         *sigdefault = attr->posix_attr_sigdefault;
126401         return 0;
126402     }
126403     int posix_spawnattr_setsigdefault(posix_spawnattr_t *attr,
126404         const sigset_t *sigdefault)
126405     {
126406         attr->posix_attr_sigdefault = *sigdefault;
126407         return 0;
126408     }
```

**I/O Redirection with Spawn**

I/O redirection with `posix_spawn()` or `posix_spawnnp()` is accomplished by crafting a `file_actions` argument to effect the desired redirection. Such a redirection follows the general outline of the following example:

```
/* To redirect new standard output (fd 1) to a file, */  
/* and redirect new standard input (fd 0) from my fd socket_pair[1], */  
/* and close my fd socket_pair[0] in the new process. */  
posix_spawn_file_actions_t file_actions;  
posix_spawn_file_actions_init(&file_actions);  
posix_spawn_file_actions_addopen(&file_actions, 1, "newout", ...);  
posix_spawn_file_actions_dup2(&file_actions, socket_pair[1], 0);  
posix_spawn_file_actions_close(&file_actions, socket_pair[0]);  
posix_spawn_file_actions_close(&file_actions, socket_pair[1]);  
posix_spawn(..., &file_actions, ...);  
posix_spawn_file_actions_destroy(&file_actions);
```

**Spawning a Process Under a New User ID**

Spawning a process under a new user ID follows the outline shown in the following example:

```
Save = getuid();  
setuid(newid);  
posix_spawn(...);  
setuid(Save);
```





126430

# *Rationale (Informative)*

126431

## **Part C:**

126432

## **Shell and Utilities**

126433

*The Open Group*

126434

*The Institute of Electrical and Electronics Engineers, Inc.*



126435

Appendix C

126436

# Rationale for Shell and Utilities

## C.1 Introduction

### C.1.1 Change History

The change history is provided as an informative section, to track changes from earlier versions of this standard.

The following sections describe changes made to the Shell and Utilities volume of POSIX.1-2008 since Issue 6 of the base document. The CHANGE HISTORY section for each utility describes technical changes made to that utility from Issue 5. Changes between earlier versions of the base document and Issue 5 are not included.

#### Changes from Issue 6 to Issue 7 (POSIX.1-2008)

The following list summarizes the major changes that were made in the Shell and Utilities volume of POSIX.1-2008 from Issue 6 to Issue 7:

- Austin Group defect reports, IEEE Interpretations against IEEE Std 1003.1, and responses to ISO/IEC defect reports against ISO/IEC 9945 are applied.
- The Open Group corrigenda and resolutions are applied.
- Features, marked legacy or obsolescent in the base document, have been considered for removal in this version.
- A review of the use of fixed pathnames within the standard has been undertaken; for example, the *at*, *batch*, and *crontab* utilities previously had a requirement for use of the directory */usr/lib/cron*.
- The options within the standard have been revised.
  - The Batch Environment Services and Utilities option is marked obsolescent.
  - The UUCP utilities option is added.
  - The User Portability Utilities option is revised so that only the *bg*, *ex*, *fc*, *fg*, *jobs*, *more*, *talk*, and *vi* utilities are included, the rest being moved to the Base.

**126461 New Features in Issue 7**

126462 There are no new utilities in Issue 7.

**126463 C.1.2 Relationship to Other Documents****126464 C.1.2.1 System Interfaces**

126465 It has been pointed out that the Shell and Utilities volume of POSIX.1-2008 assumes that a great  
126466 deal of functionality from the System Interfaces volume of POSIX.1-2008 is present, but never  
126467 states exactly how much (and strictly does not need to since both are mandated on a conforming  
126468 system). This section is an attempt to clarify the assumptions.

**126469 File Read, Write, and Creation**

126470 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/2 is applied, updating Table 1-1.

**126471 File Removal**

126472 This is intended to be a summary of the *unlink()* and *rmdir()* requirements. Note that it is  
126473 possible using the *unlink()* function for item 4. to occur.

**126474 C.1.2.2 Concepts Derived from the ISO C Standard**

126475 This section was introduced to address the issue that there was insufficient detail presented by  
126476 such utilities as *awk* or *sh* about their procedural control statements and their methods of  
126477 performing arithmetic functions.

126478 The ISO C standard was selected as a model because most historical implementations of the  
126479 standard utilities were written in C. Thus, it was more likely that they would act in the desired  
126480 manner without modification.

126481 Using the ISO C standard is primarily a notational convenience so that the many procedural  
126482 languages in the Shell and Utilities volume of POSIX.1-2008 would not have to be rigorously  
126483 described in every aspect. Its selection does not require that the standard utilities be written in  
126484 Standard C; they could be written in Common Usage C, Ada, Pascal, assembler language, or  
126485 anything else.

126486 The sizes of the various numeric values refer to C-language data types that are allowed to be  
126487 different sizes by the ISO C standard. Thus, like a C-language application, a shell application  
126488 cannot rely on their exact size. However, it can rely on their minimum sizes expressed in the  
126489 ISO C standard, such as {LONG\_MAX} for a **long** type.

126490 The behavior on overflow is undefined for ISO C standard arithmetic. Therefore, the standard  
126491 utilities can use “bignum” representation for integers so that there is no fixed maximum unless  
126492 otherwise stated in the utility description. Similarly, standard utilities can use infinite-precision  
126493 representations for floating-point arithmetic, as long as these representations exceed the ISO C  
126494 standard requirements.

126495 This section addresses only the issue of semantics; it is not intended to specify syntax. For  
126496 example, the ISO C standard requires that 0L be recognized as an integer constant equal to zero,  
126497 but utilities such as *awk* and *sh* are not required to recognize 0L (though they are allowed to, as  
126498 an extension).

126499 The ISO C standard requires that a C compiler must issue a diagnostic for constants that are too  
 126500 large to represent. Most standard utilities are not required to issue these diagnostics; for  
 126501 example, the command:

126502 `diff -C 2147483648 file1 file2`

126503 has undefined behavior, and the *diff* utility is not required to issue a diagnostic even if the  
 126504 number 2 147 483 648 cannot be represented.

### 126505 C.1.3 Utility Limits

126506 This section grew out of an idea that originated with the original POSIX.1, in the tables of system  
 126507 limits for the *sysconf()* and *pathconf()* functions. The idea being that a conforming application  
 126508 can be written to use the most restrictive values that a minimal system can provide, but it should  
 126509 not have to. The values provided represent compromises so that some vendors can use  
 126510 historically limited versions of UNIX system utilities. They are the highest values that a strictly  
 126511 conforming application can assume, given no other information.

126512 However, by using the *getconf* utility or the *sysconf()* function, the elegant application can be  
 126513 tailored to more liberal values on some of the specific instances of specific implementations.

126514 There is no explicitly stated requirement that an implementation provide finite limits for any of  
 126515 these numeric values; the implementation is free to provide essentially unbounded capabilities  
 126516 (where it makes sense), stopping only at reasonable points such as {ULONG\_MAX} (from the  
 126517 ISO C standard). Therefore, applications desiring to tailor themselves to the values on a  
 126518 particular implementation need to be ready for possibly huge values; it may not be a good idea  
 126519 to allocate blindly a buffer for an input line based on the value of {LINE\_MAX}, for instance.  
 126520 However, unlike the System Interfaces volume of POSIX.1-2008, there is no set of limits that  
 126521 return a special indication meaning “unbounded”. The implementation should always return an  
 126522 actual number, even if the number is very large.

126523 The statement:

126524 “It is not guaranteed that the application ...”

126525 is an indication that many of these limits are designed to ensure that implementors design their  
 126526 utilities without arbitrary constraints related to unimaginative programming. There are certainly  
 126527 conditions under which combinations of options can cause failures that would not render an  
 126528 implementation non-conforming. For example, {EXPR\_NEST\_MAX} and {ARG\_MAX} could  
 126529 collide when expressions are large; combinations of {BC\_SCALE\_MAX} and {BC\_DIM\_MAX}  
 126530 could exceed virtual memory.

126531 In the Shell and Utilities volume of POSIX.1-2008, the notion of a limit being guaranteed for the  
 126532 process lifetime, as it is in the System Interfaces volume of POSIX.1-2008, is not as useful to a  
 126533 shell script. The *getconf* utility is probably a process itself, so the guarantee would be without  
 126534 value. Therefore, the Shell and Utilities volume of POSIX.1-2008 requires the guarantee to be for  
 126535 the session lifetime. This will mean that many vendors will either return very conservative  
 126536 values or possibly implement *getconf* as a built-in.

126537 It may seem confusing to have limits that apply only to a single utility grouped into one global  
 126538 section. However, the alternative, which would be to disperse them out into their utility  
 126539 description sections, would cause great difficulty when *sysconf()* and *getconf* were described.  
 126540 Therefore, the standard developers chose the global approach.

126541 Each language binding could provide symbol names that are slightly different from those shown  
 126542 here. For example, the C-Language Binding option adds a leading <underscore> to the symbols  
 126543 as a prefix.

126544 The following comments describe selection criteria for the symbols and their values:

126545 {ARG\_MAX}

126546 This is defined by the System Interfaces volume of POSIX.1-2008. Unfortunately, it is very

126547 difficult for a conforming application to deal with this value, as it does not know how much

126548 of its argument space is being consumed by the environment variables of the user.

126549 {BC\_BASE\_MAX}

126550 {BC\_DIM\_MAX}

126551 {BC\_SCALE\_MAX}

126552 These were originally one value, {BC\_SCALE\_MAX}, but it was unreasonable to link all

126553 three concepts into one limit.

126554 {CHILD\_MAX}

126555 This is defined by the System Interfaces volume of POSIX.1-2008.

126556 {COLL\_WEIGHTS\_MAX}

126557 The weights assigned to **order** can be considered as “passes” through the collation

126558 algorithm.

126559 {EXPR\_NEST\_MAX}

126560 The value for expression nesting was borrowed from the ISO C standard.

126561 {LINE\_MAX}

126562 This is a global limit that affects all utilities, unless otherwise noted. The {MAX\_CANON}

126563 value from the System Interfaces volume of POSIX.1-2008 may further limit input lines from

126564 terminals. The {LINE\_MAX} value was the subject of much debate and is a compromise

126565 between those who wished to have unlimited lines and those who understood that many

126566 historical utilities were written with fixed buffers. Frequently, utility writers selected the

126567 UNIX system constant BUFSIZ to allocate these buffers; therefore, some utilities were

126568 limited to 512 bytes for I/O lines, while others achieved 4 096 bytes or greater.

126569 It should be noted that {LINE\_MAX} applies only to input line length; there is no

126570 requirement in POSIX.1-2008 that limits the length of output lines. Utilities such as *awk*, *sed*,

126571 and *paste* could theoretically construct lines longer than any of the input lines they received,

126572 depending on the options used or the instructions from the application. They are not

126573 required to truncate their output to {LINE\_MAX}. It is the responsibility of the application

126574 to deal with this. If the output of one of those utilities is to be piped into another of the

126575 standard utilities, line length restrictions will have to be considered; the *fold* utility, among

126576 others, could be used to ensure that only reasonable line lengths reach utilities or

126577 applications.

126578 {LINK\_MAX}

126579 This is defined by the System Interfaces volume of POSIX.1-2008.

126580 {MAX\_CANON}

126581 {MAX\_INPUT}

126582 {NAME\_MAX}

126583 {NGROUPS\_MAX}

126584 {OPEN\_MAX}

126585 {PATH\_MAX}

126586 {PIPE\_BUF}

126587 These limits are defined by the System Interfaces volume of POSIX.1-2008. Note that the

126588 byte lengths described by some of these values continue to represent bytes, even if the

126589 applicable character set uses a multi-byte encoding.

{RE\_DUP\_MAX}

The value selected is consistent with historical practice. Although the name implies that it applies to all REs, only BREs use the interval notation  $\{m,n\}$  addressed by this limit.

{POSIX2\_SYMLINKS}

The {POSIX2\_SYMLINKS} variable indicates that the underlying operating system supports the creation of symbolic links in specific directories. Many of the utilities defined in POSIX.1-2008 that deal with symbolic links do not depend on this value. For example, a utility that follows symbolic links (or does not, as the case may be) will only be affected by a symbolic link if it encounters one. Presumably, a file system that does not support symbolic links will not contain any. This variable does affect such utilities as *ln -s* and *pax* that attempt to create symbolic links.

There are different limits associated with command lines and input to utilities, depending on the method of invocation. In the case of a C program *exec*-ing a utility, {ARG\_MAX} is the underlying limit. In the case of the shell reading a script and *exec*-ing a utility, {LINE\_MAX} limits the length of lines the shell is required to process, and {ARG\_MAX} will still be a limit. If a user is entering a command on a terminal to the shell, requesting that it invoke the utility, {MAX\_INPUT} may restrict the length of the line that can be given to the shell to a value below {LINE\_MAX}.

When an option is supported, *getconf* returns a value of 1. For example, when C development is supported:

```
if [ "$(getconf POSIX2_C_DEV)" -eq 1 ]; then
    echo C supported
fi
```

The *sysconf()* function in the C-Language Binding option would return 1.

The following comments describe selection criteria for the symbols and their values:

POSIX2\_C\_BIND  
 POSIX2\_C\_DEV  
 POSIX2\_FORT\_DEV  
 POSIX2\_FORT\_RUN  
 POSIX2\_SW\_DEV  
 POSIX2\_UPE

It is possible for some (usually privileged) operations to remove utilities that support these options or otherwise to render these options unsupported. The header files, the *sysconf()* function, or the *getconf* utility will not necessarily detect such actions, in which case they should not be considered as rendering the implementation non-conforming. A test suite should not attempt tests such as:

```
rm /usr/bin/c99
getconf POSIX2_C_DEV
```

POSIX2\_LOCALEDEF

This symbol was introduced to allow implementations to restrict supported locales to only those supplied by the implementation.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/2 is applied, deleting the entry for {POSIX2\_VERSION} since it is not a utility limit minimum value.

IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/3 is applied, changing the text in Utility Limits from: “utility (see *getconf*) through the *sysconf()* function defined in the System Interfaces volume of POSIX.1-2008. The literal names shown in Table 1-3 apply only to the *getconf* utility; the high-level language binding describes the exact form of each name to be used by the

126637 interfaces in that binding.” to: “utility (see *getconf*).”.

#### 126638 **C.1.4 Grammar Conventions**

126639 There is no additional rationale provided for this section.

#### 126640 **C.1.5 Utility Description Defaults**

126641 This section is arranged with headings in the same order as all the utility descriptions. It is a  
126642 collection of related and unrelated information concerning:

- 126643 1. The default actions of utilities
- 126644 2. The meanings of notations used in POSIX.1-2008 that are specific to individual utility  
126645 sections

126646 Although this material may seem out of place here, it is important that this information appear  
126647 before any of the utilities to be described later.

#### 126648 **NAME**

126649 There is no additional rationale provided for this section.

#### 126650 **SYNOPSIS**

126651 There is no additional rationale provided for this section.

#### 126652 **DESCRIPTION**

126653 There is no additional rationale provided for this section.

#### 126654 **OPTIONS**

126655 Although it has not always been possible, the standard developers tried to avoid repeating  
126656 information to reduce the risk that duplicate explanations could each be modified differently.

126657 The need to recognize `--` is required because conforming applications need to shield their  
126658 operands from any arbitrary options that the implementation may provide as an extension. For  
126659 example, if the standard utility *foo* is listed as taking no options, and the application needed to  
126660 give it a pathname with a leading <hyphen-minus>, it could safely do it as:

126661 `foo -- -myfile`

126662 and avoid any problems with `-m` used as an extension.

#### 126663 **OPERANDS**

126664 The usage of `-` is never shown in the SYNOPSIS. Similarly, the usage of `--` is never shown.

126665 The requirement for processing operands in command-line order is to avoid a “WeirdNIX”  
126666 utility that might choose to sort the input files alphabetically, by size, or by directory order.  
126667 Although this might be acceptable for some utilities, in general the programmer has a right to  
126668 know exactly what order will be chosen.

126669 Some of the standard utilities take multiple *file* operands and act as if they were processing the  
126670 concatenation of those files. For example:

126671 `asa file1 file2`



126672 and:

126673 `cat file1 file2 | asa`

126674 have similar results when questions of file access, errors, and performance are ignored. Other  
 126675 utilities such as *grep* or *wc* have completely different results in these two cases. This latter type of  
 126676 utility is always identified in its DESCRIPTION or OPERANDS sections, whereas the former is  
 126677 not. Although it might be possible to create a general assertion about the former case, the  
 126678 following points must be addressed:

- 126679 • Access times for the files might be different in the operand case *versus* the *cat* case.
- 126680 • The utility may have error messages that are cognizant of the input filename, and this  
 126681 added value should not be suppressed. (As an example, *awk* sets a variable with the  
 126682 filename at each file boundary.)

## 126683 STDIN

126684 There is no additional rationale provided for this section.

## 126685 INPUT FILES

126686 A conforming application cannot assume the following three commands are equivalent:

```
126687 tail -n +2 file
126688 (sed -n 1q; cat) < file
126689 cat file | (sed -n 1q; cat)
```

126690 The second command is equivalent to the first only when the file is seekable. In the third  
 126691 command, if the file offset in the open file description were not unspecified, *sed* would have to  
 126692 be implemented so that it read from the pipe 1 byte at a time or it would have to employ some  
 126693 method to seek backwards on the pipe. Such functionality is not defined currently in POSIX.1  
 126694 and does not exist on all historical systems. Other utilities, such as *head*, *read*, and *sh*, have similar  
 126695 properties, so the restriction is described globally in this section.

126696 The definition of “text file” is strictly enforced for input to the standard utilities; very few of  
 126697 them list exceptions to the undefined results called for here. (Of course, “undefined” here does  
 126698 not mean that historical implementations necessarily have to change to start indicating error  
 126699 conditions. Conforming applications cannot rely on implementations succeeding or failing when  
 126700 non-text files are used.)

126701 The utilities that allow line continuation are generally those that accept input languages, rather  
 126702 than pure data. It would be unusual for an input line of this type to exceed {LINE\_MAX} bytes  
 126703 and unreasonable to require that the implementation allow unlimited accumulation of multiple  
 126704 lines, each of which could reach {LINE\_MAX}. Thus, for a conforming application the total of all  
 126705 the continued lines in a set cannot exceed {LINE\_MAX}.

126706 The format description is intended to be sufficiently rigorous to allow other applications to  
 126707 generate these input files. However, since <blank> characters can legitimately be included in  
 126708 some of the fields described by the standard utilities, particularly in locales other than the POSIX  
 126709 locale, this intent is not always realized.

**126710 ENVIRONMENT VARIABLES**

126711 There is no additional rationale provided for this section.

**126712 ASYNCHRONOUS EVENTS**

126713 Because there is no language prohibiting it, a utility is permitted to catch a signal, perform some  
126714 additional processing (such as deleting temporary files), restore the default signal action (or  
126715 action inherited from the parent process), and resignal itself.

**126716 STDOUT**

126717 The format description is intended to be sufficiently rigorous to allow post-processing of output  
126718 by other programs, particularly by an *awk* or *lex* parser.

**126719 STDERR**

126720 This section does not describe error messages that refer to incorrect operation of the utility.  
126721 Consider a utility that processes program source code as its input. This section is used to  
126722 describe messages produced by a correctly operating utility that encounters an error in the  
126723 program source code on which it is processing. However, a message indicating that the utility  
126724 had insufficient memory in which to operate would not be described.

126725 Some utilities have traditionally produced warning messages without returning a non-zero exit  
126726 status; these are specifically noted in their sections. Other utilities shall not write to standard  
126727 error if they complete successfully, unless the implementation provides some sort of extension to  
126728 increase the verbosity or debugging level.

126729 The format descriptions are intended to be sufficiently rigorous to allow post-processing of  
126730 output by other programs.

**126731 OUTPUT FILES**

126732 The format description is intended to be sufficiently rigorous to allow post-processing of output  
126733 by other programs, particularly by an *awk* or *lex* parser.

126734 Receipt of the SIGQUIT signal should generally cause termination (unless in some debugging  
126735 mode) that would bypass any attempted recovery actions.

**126736 EXTENDED DESCRIPTION**

126737 There is no additional rationale provided for this section.

**126738 EXIT STATUS**

126739 Note the additional discussion of exit values in *Exit Status for Commands* in the *sh* utility. It  
126740 describes requirements for returning exit values greater than 125.

126741 A utility may list zero as a successful return, 1 as a failure for a specific reason, and greater than  
126742 1 as “an error occurred”. In this case, unspecified conditions may cause a 2 or 3, or other value,  
126743 to be returned. A strictly conforming application should be written so that it tests for successful  
126744 exit status values (zero in this case), rather than relying upon the single specific error value listed  
126745 in POSIX.1-2008. In that way, it will have maximum portability, even on implementations with  
126746 extensions.

126747 The standard developers are aware that the general non-enumeration of errors makes it difficult  
126748 to write test suites that test the *incorrect* operation of utilities. There are some historical  
126749 implementations that have expended effort to provide detailed status messages and a helpful  
126750 environment to bypass or explain errors, such as prompting, retrying, or ignoring unimportant

126751 syntax errors; other implementations have not. Since there is no realistic way to mandate system  
 126752 behavior in cases of undefined application actions or system problems—in a manner acceptable  
 126753 to all cultures and environments—attention has been limited to the correct operation of utilities  
 126754 by the conforming application. Furthermore, the conforming application does not need detailed  
 126755 information concerning errors that it caused through incorrect usage or that it cannot correct.

126756 There is no description of defaults for this section because all of the standard utilities specify  
 126757 something (or explicitly state “Unspecified”) for exit status.

## 126758 CONSEQUENCES OF ERRORS

126759 Several actions are possible when a utility encounters an error condition, depending on the  
 126760 severity of the error and the state of the utility. Included in the possible actions of various  
 126761 utilities are: deletion of temporary or intermediate work files; deletion of incomplete files; and  
 126762 validity checking of the file system or directory.

126763 The text about recursive traversing is meant to ensure that utilities such as *find* process as many  
 126764 files in the hierarchy as they can. They should not abandon all of the hierarchy at the first error  
 126765 and resume with the next command-line operand, but should attempt to keep going.

126766 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0001 [150] is applied.

## 126767 APPLICATION USAGE

126768 This section provides additional caveats, issues, and recommendations to the developer.

## 126769 EXAMPLES

126770 This section provides sample usage.

## 126771 RATIONALE

126772 There is no additional rationale provided for this section.

## 126773 FUTURE DIRECTIONS

126774 FUTURE DIRECTIONS sections act as pointers to related work that may impact the interface in  
 126775 the future, and often cautions the developer to architect the code to account for a change in this  
 126776 area. Note that a future directions statement should not be taken as a commitment to adopt a  
 126777 feature or interface in the future.

## 126778 SEE ALSO

126779 There is no additional rationale provided for this section.

## 126780 CHANGE HISTORY

126781 There is no additional rationale provided for this section.

**C.1.6 Considerations for Utilities in Support of Files of Arbitrary Size**

This section is intended to clarify the requirements for utilities in support of large files.

The utilities listed in this section are utilities which are used to perform administrative tasks such as to create, move, copy, remove, change the permissions, or measure the resources of a file. They are useful both as end-user tools and as utilities invoked by applications during software installation and operation.

The *chgrp*, *chmod*, *chown*, *ln*, and *rm* utilities probably require use of large file-capable versions of *stat()*, *lstat()*, *ftw()*, and the **stat** structure.

The *cat*, *cksum*, *cmp*, *cp*, *dd*, *mv*, *sum*, and *touch* utilities probably require use of large file-capable versions of *creat()*, *open()*, and *fopen()*.

The *cat*, *cksum*, *cmp*, *dd*, *df*, *du*, *ls*, and *sum* utilities may require writing large integer values. For example:

- The *cat* utility might have a **-n** option which counts <newline> characters.
- The *cksum* and *ls* utilities report file sizes.
- The *cmp* utility reports the line number at which the first difference occurs, and also has a **-l** option which reports file offsets.
- The *dd*, *df*, *du*, *ls*, and *sum* utilities report block counts.

The *dd*, *find*, and *test* utilities may need to interpret command arguments that contain 64-bit values. For *dd*, the arguments include *skip=n*, *seek=n*, and *count=n*. For *find*, the arguments include **-sizen**. For *test*, the arguments are those associated with algebraic comparisons.

The *df* utility might need to access large file systems with *statvfs()*.

The *ulimit* utility will need to use large file-capable versions of *getrlimit()* and *setrlimit()* and be able to read and write large integer values.

**C.1.7 Built-In Utilities**

All of these utilities can be *exec*-ed. There is no requirement that these utilities are actually built into the shell itself, but many shells need the capability to do so because XCU [Section 2.9.1.1](#) (on page 2357) requires that they be found prior to the *PATH* search. The shell could satisfy its requirements by keeping a list of the names and directly accessing the file-system versions regardless of *PATH*. Providing all of the required functionality for those such as *cd* or *read* would be more difficult.

There were originally three justifications for allowing the omission of *exec*-able versions:

1. It would require wasting space in the file system, at the expense of very small systems. However, it has been pointed out that all 16 utilities in the table can be provided with 16 links to a single-line shell script:

```
$0 "$@"
```

2. It is not logical to require invocation of utilities such as *cd* because they have no value outside the shell environment or cannot be useful in a child process. However, counter-examples always seemed to be available for even the most unusual cases:

```
find . -type d -exec cd {} \; -exec foo {} \;
      (which invokes "foo" on accessible directories)
```

```
ps ... | sed ... | xargs kill
```

126823 `find . -exec true \; -a ...`

126824 (where “true” is used for temporary debugging)

- 126825 3. It is confusing to have a utility such as *kill* that can easily be in the file system in the base  
 126826 standard, but that requires built-in status for the User Portability Utilities option (for the  
 126827 % job control job ID notation). It was decided that it was more appropriate to describe the  
 126828 required functionality (rather than the implementation) to the system implementors and  
 126829 let them decide how to satisfy it.

126830 On the other hand, it was realized that any distinction like this between utilities was not useful  
 126831 to applications, and that the cost to correct it was small. These arguments were ultimately the  
 126832 most effective.

126833 There were varying reasons for including utilities in the table of built-ins:

126834 *alias, fc, unalias*

126835 The functionality of these utilities is performed more simply within the shell itself and that  
 126836 is the model most historical implementations have used.

126837 *bg, fg, jobs*

126838 All of the job control-related utilities are eligible for built-in status because that is the model  
 126839 most historical implementations have used.

126840 *cd, getopts, newgrp, read, umask, wait*

126841 The functionality of these utilities is performed more simply within the context of the  
 126842 current process. An example can be taken from the usage of the *cd* utility. The purpose of  
 126843 the *cd* utility is to change the working directory for subsequent operations. The actions of *cd*  
 126844 affect the process in which *cd* is executed and all subsequent child processes of that process.  
 126845 Based on the POSIX standard process model, changes in the process environment of a child  
 126846 process have no effect on the parent process. If the *cd* utility were executed from a child  
 126847 process, the working directory change would be effective only in the child process. Child  
 126848 processes initiated subsequent to the child process that executed the *cd* utility would not  
 126849 have a changed working directory relative to the parent process.

126850 *command*

126851 This utility was placed in the table primarily to protect scripts that are concerned about  
 126852 their *PATH* being manipulated. The “secure” shell script example in the *command* utility in  
 126853 the Shell and Utilities volume of POSIX.1-2008 would not be possible if a *PATH* change  
 126854 retrieved an alien version of *command*. (An alternative would have been to implement  
 126855 *getconf* as a built-in, but the standard developers considered that it carried too many  
 126856 changing configuration strings to require in the shell.)

126857 *kill* Since *kill* provides optional job control functionality using shell notation (%1, %2, and so on),  
 126858 some implementations would find it extremely difficult to provide this outside the shell.

126859 *true, false*

126860 These are in the table as a courtesy to programmers who wish to use the “while true”  
 126861 shell construct without protecting *true* from *PATH* searches. (It is acknowledged that  
 126862 “while :” also works, but the idiom with *true* is historically pervasive.)

126863 All utilities, including those in the table, are accessible via the *system()* and *popen()* functions in  
 126864 the System Interfaces volume of POSIX.1-2008. There are situations where the return  
 126865 functionality of *system()* and *popen()* is not desirable. Applications that require the exit status of  
 126866 the invoked utility will not be able to use *system()* or *popen()*, since the exit status returned is  
 126867 that of the command language interpreter rather than that of the invoked utility. The alternative  
 126868 for such applications is the use of the *exec* family.

## 126869 C.2 Shell Command Language

### 126870 C.2.1 Shell Introduction

126871 The System V shell was selected as the starting point for the Shell and Utilities volume of  
126872 POSIX.1-2008. The BSD C shell was excluded from consideration for the following reasons:

- 126873 • Most historically portable shell scripts assume the Version 7 Bourne shell, from which the  
126874 System V shell is derived.
- 126875 • The majority of tutorial materials on shell programming assume the System V shell.

126876 The construct "#!" is reserved for implementations wishing to provide that extension. If it were  
126877 not reserved, the Shell and Utilities volume of POSIX.1-2008 would disallow it by forcing it to be  
126878 a comment. As it stands, a strictly conforming application must not use "#!" as the first two  
126879 characters of the file.

### 126880 C.2.2 Quoting

126881 There is no additional rationale provided for this section.

#### 126882 C.2.2.1 Escape Character (Backslash)

126883 There is no additional rationale provided for this section.

#### 126884 C.2.2.2 Single-Quotes

126885 A <backslash> cannot be used to escape a single-quote in a single-quoted string. An embedded  
126886 quote can be created by writing, for example: "'a'\''b'", which yields "a'b". (See XCU  
126887 [Section 2.6.5](#) (on page 2349) for a better understanding of how portions of words are either split  
126888 into fields or remain concatenated.) A single token can be made up of concatenated partial  
126889 strings containing all three kinds of quoting or escaping, thus permitting any combination of  
126890 characters.

#### 126891 C.2.2.3 Double-Quotes

126892 The escaped <newline> used for line continuation is removed entirely from the input and is not  
126893 replaced by any white space. Therefore, it cannot serve as a token separator.

126894 In double-quoting, if a <backslash> is immediately followed by a character that would be  
126895 interpreted as having a special meaning, the <backslash> is deleted and the subsequent  
126896 character is taken literally. If a <backslash> does not precede a character that would have a  
126897 special meaning, it is left in place unmodified and the character immediately following it is also  
126898 left unmodified. Thus, for example:

126899 "\$" -> \$

126900 "\a" -> \a

126901 It would be desirable to include the statement "The characters from an enclosed "\${ to the  
126902 matching '}' shall not be affected by the double-quotes", similar to the one for "\$()".  
126903 However, historical practice in the System V shell prevents this.



126904 The requirement that double-quotes be matched inside "\${...}" within double-quotes and the  
126905 rule for finding the matching '}' in XCU [Section 2.6.2](#) (on page 2344) eliminate several subtle  
126906 inconsistencies in expansion for historical shells in rare cases; for example:

126907 "\$ {foo-bar} "

126908 yields **bar** when **foo** is not defined, and is an invalid substitution when **foo** is defined, in many  
126909 historical shells. The differences in processing the "\${...}" form have led to inconsistencies  
126910 between historical systems. A consequence of this rule is that single-quotes cannot be used to  
126911 quote the '}' within "\${...}"; for example:

126912 unset bar  
126913 foo="\$ {bar-' }' "

126914 is invalid because the "\${...}" substitution contains an unpaired unescaped single-quote. The  
126915 <backslash> can be used to escape the '}' in this example to achieve the desired result:

126916 unset bar  
126917 foo="\$ {bar-\} }" "

126918 The differences in processing the "\${...}" form have led to inconsistencies between the  
126919 historical System V shell, BSD, and KornShells, and the text in the Shell and Utilities volume of  
126920 POSIX.1-2008 is an attempt to converge them without breaking too many applications. The only  
126921 alternative to this compromise between shells would be to make the behavior unspecified  
126922 whenever the literal characters single-quote, '{', '}', and '"' appear within "\${...}". To  
126923 write a portable script that uses these values, a user would have to assign variables; for example:

126924 squote='\ ' dquote=\" lbrace='{ ' rbrace=' }'  
126925 \${foo-\$squote\$rbrace\$squote}

126926 rather than:

126927 \${foo-" ' }' ' ' }

126928 Some implementations have allowed the end of the word to terminate the backquoted command  
126929 substitution, such as in:

126930 "`echo hello"

126931 This usage is undefined; the matching backquote is required by the Shell and Utilities volume of  
126932 POSIX.1-2008. The other undefined usage can be illustrated by the example:

126933 sh -c '` echo "foo`'

126934 The description of the recursive actions involving command substitution can be illustrated with  
126935 an example. Upon recognizing the introduction of command substitution, the shell parses input  
126936 (in a new context), gathering the source for the command substitution until an unbalanced ')' or  
126937 or '`' is located. For example, in the following:

126938 echo "\$(date; echo "  
126939 one" )"

126940 the double-quote following the *echo* does not terminate the first double-quote; it is part of the  
126941 command substitution script. Similarly, in:

126942 echo "\$(echo \*)" "

126943 the <asterisk> is not quoted since it is inside command substitution; however:

126944 echo "\$(echo " \* ") "

126945 is quoted (and represents the <asterisk> character itself).

## C.2.3 Token Recognition

The "(" and ")" symbols are control operators in the KornShell, used for an alternative syntax of an arithmetic expression command. A conforming application cannot use "(" as a single token (with the exception of the "\$(" form for shell arithmetic).

On some implementations, the symbol "(" is a control operator; its use produces unspecified results. Applications that wish to have nested subshells, such as:

```
((echo Hello);(echo World))
```

must separate the "(" characters into two tokens by including white space between them. Some systems may treat these as invalid arithmetic expressions instead of subshells.

Certain combinations of characters are invalid in portable scripts, as shown in the grammar. Implementations may use these combinations (such as "|&") as valid control operators. Portable scripts cannot rely on receiving errors in all cases where this volume of POSIX.1-2008 indicates that a syntax is invalid.

The (3) rule about combining characters to form operators is not meant to preclude systems from extending the shell language when characters are combined in otherwise invalid ways. Conforming applications cannot use invalid combinations, and test suites should not penalize systems that take advantage of this fact. For example, the unquoted combination "|&" is not valid in a POSIX script, but has a specific KornShell meaning.

The (10) rule about '#' as the current character is the first in the sequence in which a new token is being assembled. The '#' starts a comment only when it is at the beginning of a token. This rule is also written to indicate that the search for the end-of-comment does not consider escaped <newline> specially, so that a comment cannot be continued to the next line.

### C.2.3.1 Alias Substitution

The alias capability was added because it is widely used in historical implementations by interactive users.

The definition of "alias name" precludes an alias name containing a <slash> character. Since the text applies to the command words of simple commands, reserved words (in their proper places) cannot be confused with aliases.

The placement of alias substitution in token recognition makes it clear that it precedes all of the word expansion steps.

An example concerning trailing <blank> characters and reserved words follows. If the user types:

```
$ alias foo="/bin/ls "  
$ alias while="/"
```

The effect of executing:

```
$ while true  
> do  
> echo "Hello, World"  
> done
```

is a never-ending sequence of "Hello, World" strings to the screen. However, if the user types:

```
$ foo while
```



126988 the result is an *ls* listing of */*. Since the alias substitution for **foo** ends in a <space>, the next word  
126989 is checked for alias substitution. The next word, **while**, has also been aliased, so it is substituted  
126990 as well. Since it is not in the proper position as a command word, it is not recognized as a  
126991 reserved word.

126992 If the user types:

126993 **\$** foo; while

126994 **while** retains its normal reserved-word properties.

## 126995 C.2.4 Reserved Words

126996 All reserved words are recognized syntactically as such in the contexts described. However, note  
126997 that **in** is the only meaningful reserved word after a **case** or **for**; similarly, **in** is not meaningful as  
126998 the first word of a simple command.

126999 Reserved words are recognized only when they are delimited (that is, meet the definition of XBD  
127000 Section 3.446, on page 105), whereas operators are themselves delimiters. For instance, ' ( ' and  
127001 ' ) ' are control operators, so that no <space> is needed in (*list*). However, ' { ' and ' } ' are  
127002 reserved words in { *list*; }, so that in this case the leading <space> and <semicolon> are required.

127003 The list of unspecified reserved words is from the KornShell, so conforming applications cannot  
127004 use them in places a reserved word would be recognized. This list contained **time** in early  
127005 proposals, but it was removed when the *time* utility was selected for the Shell and Utilities  
127006 volume of POSIX.1-2008.

127007 There was a strong argument for promoting braces to operators (instead of reserved words), so  
127008 they would be syntactically equivalent to subshell operators. Concerns about compatibility  
127009 outweighed the advantages of this approach. Nevertheless, conforming applications should  
127010 consider quoting ' { ' and ' } ' when they represent themselves.

127011 The restriction on ending a name with a <colon> is to allow future implementations that support  
127012 named labels for flow control; see the RATIONALE for the *break* built-in utility.

127013 It is possible that a future version of the Shell and Utilities volume of POSIX.1-2008 may require  
127014 that ' { ' and ' } ' be treated individually as control operators, although the token " { } " will  
127015 probably be a special-case exemption from this because of the often-used *find*{ } construct.

## 127016 C.2.5 Parameters and Variables

### 127017 C.2.5.1 Positional Parameters

127018 There is no additional rationale provided for this section.

### 127019 C.2.5.2 Special Parameters

127020 Most historical implementations implement subshells by forking; thus, the special parameter  
127021 ' \$ ' does not necessarily represent the process ID of the shell process executing the commands  
127022 since the subshell execution environment preserves the value of ' \$ '.

127023 If a subshell were to execute a background command, the value of " \$ ! " for the parent would  
127024 not change. For example:

```
127025 (
127026 date &
127027 echo $!
127028 )
127029 echo $!
```

127030 would echo two different values for "\$!".

127031 The "\$-" special parameter can be used to save and restore *set* options:

```
127032 Save=$(echo $- | sed 's/[ics]//g')
127033 ...
127034 set +aCefnuvx
127035 if [ -n "$Save" ]; then
127036     set -$Save
127037 fi
```

127038 The three options are removed using *sed* in the example because they may appear in the value of  
127039 "\$-" (from the *sh* command line), but are not valid options to *set*.

127040 The descriptions of parameters '\*' and '@' assume the reader is familiar with the field  
127041 splitting discussion in XCU [Section 2.6.5](#) (on page 2349) and understands that portions of the  
127042 word remain concatenated unless there is some reason to split them into separate fields.

127043 The following examples illustrate some of the ways in which '\*' and '@' can be expanded:

```
127044 set "abc" "def ghi" "jkl"
127045 unset novar
127046 IFS=' ' # a space
127047 printf '%s\n' $*
127048 abc
127049 def
127050 ghi
127051 jkl
127052 printf '%s\n' "$*"
127053 abc def ghi jkl
127054 printf '%s\n' xx$*yy
127055 xxabc
127056 def
127057 ghi
127058 jklyy
127059 printf '%s\n' "xx$*yy"
127060 xxabc def ghi jklyy
127061 printf '%s\n' $@
127062 abc
127063 def
127064 ghi
127065 jkl
127066 printf '%s\n' "$@"
127067 abc
127068 def ghi
127069 jkl
127070 printf '%s\n' ${1+"$@"}
127071 abc
127072 def ghi
127073 jkl
```

```
127074 printf 's\n' ${novar-"$@"}
127075 abc
127076 def ghi
127077 jkl
127078 printf 's\n' xx$@yy
127079 xxabc
127080 def
127081 ghi
127082 jklyy
127083 printf 's\n' "xx$@yy"
127084 xxabc
127085 def ghi
127086 jklyy
127087 printf 's\n' @$@$
127088 abc
127089 def
127090 ghi
127091 jklabc
127092 def
127093 ghi
127094 jkl
127095 printf 's\n' "$@$@"
127096 abc
127097 def ghi
127098 jklabc
127099 def ghi
127100 jkl
127101 IFS=:
127102 printf 's\n' "$*"
127103 abc:def ghi: jkl
127104 var=$*; printf 's\n' "$var"
127105 abc:def ghi: jkl
127106 var="$*"; printf 's\n' "$var"
127107 abc:def ghi: jkl
127108 unset var
127109 printf 's\n' ${var-$*}
127110 abc
127111 def ghi
127112 jkl
127113 printf 's\n' "${var-$*}"
127114 abc:def ghi: jkl
127115 printf 's\n' ${var-"$*"}
127116 abc:def ghi: jkl
127117 printf 's\n' ${var=$*}
127118 abc
127119 def ghi
127120 jkl
127121 printf 'var=%s\n' "$var"
127122 var=abc:def ghi: jkl
127123 unset var
127124 printf 's\n' "${var=$*}"
127125 abc:def ghi: jkl
127126 printf 'var=%s\n' "$var"
```

```

127127      var=abc:def ghi:jkl
127128      IFS=' ' # null
127129      printf '%s\n' "$*"
127130      abcdef ghijkl
127131      var=$*; printf '%s\n' "$var"
127132      abcdef ghijkl
127133      var="$*"; printf '%s\n' "$var"
127134      abcdef ghijkl
127135      unset var
127136      printf '%s\n' "${var-$*}"
127137      abcdef ghijkl
127138      printf '%s\n' "${var-$*}"
127139      abcdef ghijkl
127140      printf '%s\n' "${var-$*}"
127141      abcdef ghijkl
127142      printf '%s\n' "${var=$*}"
127143      abcdef ghijkl
127144      printf 'var=%s\n' "$var"
127145      var=abcdef ghijkl
127146      unset var
127147      printf '%s\n' "${var=$*}"
127148      abcdef ghijkl
127149      printf 'var=%s\n' "$var"
127150      var=abcdef ghijkl
127151      printf '%s\n' "$@"
127152      abc
127153      def ghi
127154      jkl

127155      unset IFS
127156      printf '%s\n' "$*"
127157      abc def ghi jkl
127158      var=$*; printf '%s\n' "$var"
127159      abc def ghi jkl
127160      var="$*"; printf '%s\n' "$var"
127161      abc def ghi jkl
127162      unset var
127163      printf '%s\n' "${var-$*}"
127164      abc
127165      def
127166      ghi
127167      jkl
127168      printf '%s\n' "${var-$*}"
127169      abc def ghi jkl
127170      printf '%s\n' "${var-$*}"
127171      abc def ghi jkl
127172      printf '%s\n' "${var=$*}"
127173      abc
127174      def
127175      ghi
127176      jkl
127177      printf 'var=%s\n' "$var"
127178      var=abc def ghi jkl

```

```

127179      unset var
127180      printf '%s\n' "${var=$*}"
127181      abc def ghi jkl
127182      printf 'var=%s\n' "$var"
127183      var=abc def ghi jkl
127184      printf '%s\n' "$@"
127185      abc
127186      def ghi
127187      jkl

127188      set one "" three
127189      printf ' [%s]\n' $*
127190      [one]
127191      [ ] (this line of output is optional)
127192      [three]
127193      printf ' [%s]\n' $@
127194      [one]
127195      [ ] (this line of output is optional)
127196      [three]

127197      set --
127198      printf ' [%s]\n' foo "$*"
127199      [foo]
127200      [ ]
127201      printf ' [%s]\n' foo "$novar$*(echo)"
127202      [foo]
127203      [ ]
127204      printf ' [%s]\n' foo $@
127205      [foo]
127206      printf ' [%s]\n' foo "$@"
127207      [foo]
127208      printf ' [%s]\n' foo '$@'
127209      [foo]
127210      [ ]
127211      printf ' [%s]\n' foo '"$@"'
127212      [foo]
127213      [ ]
127214      printf ' [%s]\n' foo "$novar$@(echo)"
127215      [foo]
127216      [ ] (this line of output is optional)
127217      printf ' [%s]\n' foo '"$novar$@(echo)"'
127218      [foo]
127219      [ ]

127220      In all of the following commands the results of the expansion of '@' (if performed) are
127221      unspecified:

127222      var=$@
127223      var="$@"
127224      printf '%s\n' ${var=$@}
127225      printf '%s\n' "${var=$@}"
127226      printf '%s\n' ${var="$@"}
127227      printf '%s\n' ${var?@$}
127228      printf '%s\n' "${var?@$}"
127229      printf '%s\n' ${var?"$@"}

```

```

127230 printf '%s\n' "${#@}
127231 printf '%s\n' "${#@}"
127232 printf '%s\n' "${@%foo}"
127233 printf '%s\n' "${@%foo}"
127234 printf '%s\n' "${@#foo}"
127235 printf '%s\n' "${@#foo}"
127236 printf '%s\n' "${var%$@}"
127237 printf '%s\n' "${var%$@}"
127238 printf '%s\n' "${var%"$@"}"
127239 printf '%s\n' "${var%%$@}"
127240 printf '%s\n' "${var%%$@}"
127241 printf '%s\n' "${var%%"$@"}"
127242 printf '%s\n' "${var#$@}"
127243 printf '%s\n' "${var#$@}"
127244 printf '%s\n' "${var#"$@"}"
127245 printf '%s\n' "${var##$@}"
127246 printf '%s\n' "${var##$@}"
127247 printf '%s\n' "${var##"$@"}"
    
```

#### 127248 C.2.5.3 Shell Variables

127249 See the discussion of *IFS* in [Section C.2.6.5](#) (on page 3722) and the RATIONALE for the *sh* utility.

127250 The prohibition on *LC\_CTYPE* changes affecting lexical processing protects the shell  
 127251 implementor (and the shell programmer) from the ill effects of changing the definition of  
 127252 <blank> or the set of alphabetic characters in the current environment. It would probably not be  
 127253 feasible to write a compiled version of a shell script without this rule. The rule applies only to  
 127254 the current invocation of the shell and its subshells—invoking a shell script or performing *exec*  
 127255 *sh* would subject the new shell to the changes in *LC\_CTYPE*.

127256 Other common environment variables used by historical shells are not specified by the Shell and  
 127257 Utilities volume of POSIX.1-2008, but they should be reserved for the historical uses.

127258 Tilde expansion for components of *PATH* in an assignment such as:

127259 `PATH=~hlj/bin:~dwc/bin:$PATH`

127260 is a feature of some historical shells and is allowed by the wording of XCU [Section 2.6.1](#) (on page  
 127261 2344). Note that the <tilde> characters are expanded during the assignment to *PATH*, not when  
 127262 *PATH* is accessed during command search.

127263 The following entries represent additional information about variables included in the Shell and  
 127264 Utilities volume of POSIX.1-2008, or rationale for common variables in use by shells that have  
 127265 been excluded:

127266 — (Underscore.) While <underscore> is historical practice, its overloaded usage  
 127267 in the KornShell is confusing, and it has been omitted from the Shell and  
 127268 Utilities volume of POSIX.1-2008.

127269 **ENV** This variable can be used to set aliases and other items local to the invocation  
 127270 of a shell. The file referred to by *ENV* differs from **\$HOME/.profile** in that  
 127271 **.profile** is typically executed at session start-up, whereas the *ENV* file is  
 127272 executed at the beginning of each shell invocation. The *ENV* value is  
 127273 interpreted in a manner similar to a dot script, in that the commands are  
 127274 executed in the current environment and the file needs to be readable, but not  
 127275 executable. However, unlike dot scripts, no *PATH* searching is performed. This

127276		is used as a guard against Trojan Horse security breaches.
127277	<i>ERRNO</i>	This variable was omitted from the Shell and Utilities volume of POSIX.1-2008
127278		because the values of error numbers are not defined in POSIX.1-2008 in a
127279		portable manner.
127280	<i>FCEDIT</i>	Since this variable affects only the <i>fc</i> utility, it has been omitted from this more
127281		global place. The value of <i>FCEDIT</i> does not affect the command-line editing
127282		mode in the shell; see the description of <i>set -o vi</i> in the <i>set</i> built-in utility.
127283	<i>PS1</i>	This variable is used for interactive prompts. Historically, the “superuser”
127284		has had a prompt of ‘#’. Since privileges are not required to be monolithic, it
127285		is difficult to define which privileges should cause the alternate prompt.
127286		However, a sufficiently powerful user should be reminded of that power by
127287		having an alternate prompt.
127288	<i>PS3</i>	This variable is used by the KornShell for the <i>select</i> command. Since the POSIX
127289		shell does not include <i>select</i> , <i>PS3</i> was omitted.
127290	<i>PS4</i>	This variable is used for shell debugging. For example, the following script:
127291		<pre>PS4=' [ \${LINENO} ]+ '</pre>
127292		<pre>set -x</pre>
127293		<pre>echo Hello</pre>
127294		writes the following to standard error:
127295		<pre>[3]+ echo Hello</pre>
127296	<i>RANDOM</i>	This pseudo-random number generator was not seen as being useful to
127297		interactive users.
127298	<i>SECONDS</i>	Although this variable is sometimes used with <i>PS1</i> to allow the display of the
127299		current time in the prompt of the user, it is not one that would be manipulated
127300		frequently enough by an interactive user to include in the Shell and Utilities
127301		volume of POSIX.1-2008.
127302		POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0002 [152] is applied.

## 127303 C.2.6 Word Expansions

127304	Step (2) refers to the “portions of fields generated by step (1)”. For example, if the word being
127305	expanded were “\$x+\$y” and <i>IFS</i> =+, the word would be split only if “\$x” or “\$y” contained
127306	‘+’; the ‘+’ in the original word was not generated by step (1).
127307	<i>IFS</i> is used for performing field splitting on the results of parameter and command substitution;
127308	it is not used for splitting all fields. Earlier versions of the shell used it for splitting all fields
127309	during field splitting, but this has severe problems because the shell can no longer parse its own
127310	script. There are also important security implications caused by this behavior. All useful
127311	applications of <i>IFS</i> use it for parsing input of the <i>read</i> utility and for splitting the results of
127312	parameter and command substitution.
127313	The rule concerning expansion to a single field requires that if <b>foo=abc</b> and <b>bar=def</b> , that:
127314	<code>"\$foo" "\$bar"</code>

127315 expands to the single field:

127316 abcdef

127317 The rule concerning empty fields can be illustrated by:

```
127318 $ unset foo
127319 $ set $foo bar ' ' xyz "$foo" abc
127320 $ for i
127321 > do
127322 >     echo "-$i-"
127323 > done
127324 -bar-
127325 --
127326 -xyz-
127327 --
127328 -abc-
```

127329 Step (1) indicates that parameter expansion, command substitution, and arithmetic expansion  
127330 are all processed simultaneously as they are scanned. For example, the following is valid  
127331 arithmetic:

```
127332 x=1
127333 echo $(( $(echo 3)+$x ))
```

127334 An early proposal stated that tilde expansion preceded the other steps, but this is not the case in  
127335 known historical implementations; if it were, and if a referenced home directory contained a '\$'  
127336 character, expansions would result within the directory name.

127337 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0003 [49,430] is applied.

#### 127338 C.2.6.1 Tilde Expansion

127339 Tilde expansion generally occurs only at the beginning of words, but an exception based on  
127340 historical practice has been included:

```
127341 PATH=/posix/bin:~dgg/bin
```

127342 This is eligible for tilde expansion because <tilde> follows a <colon> and none of the relevant  
127343 characters is quoted. Consideration was given to prohibiting this behavior because any of the  
127344 following are reasonable substitutes:

```
127345 PATH=$(printf %s ~karels/bin : ~bostic/bin)
127346 for Dir in ~maat/bin ~srb/bin ...
127347 do
127348     PATH=${PATH:+$PATH:}$Dir
127349 done
```

127350 In the first command, explicit <colon> characters are used for each directory. In all cases, the  
127351 shell performs tilde expansion on each directory because all are separate words to the shell.

127352 Note that expressions in operands such as:

```
127353 make -k mumble LIBDIR=~chet/lib
```

127354 do not qualify as shell variable assignments, and tilde expansion is not performed (unless the  
127355 command does so itself, which *make* does not).

127356 Because of the requirement that the word is not quoted, the following are not equivalent; only  
127357 the last causes tilde expansion:



127358        \~hlj/    ~h\lj/    ~"hlj"/    ~hlj\    ~hlj/

127359        In an early proposal, tilde expansion occurred following any unquoted <equals-sign> or  
127360        <colon>, but this was removed because of its complexity and to avoid breaking commands such  
127361        as:

127362        rcp hostname:~marc/.profile .

127363        System administrators on systems where // has an implementation-defined meaning which is  
127364        different to /, should not create users with a home directory of / or //, since this may lead to  
127365        unexpected filename resolution on those systems.

127366        A suggestion was made that the special sequence "\$~" should be allowed to force tilde  
127367        expansion anywhere. Since this is not historical practice, it has been left for future  
127368        implementations to evaluate. (The description in XCU [Section 2.2](#) (on page 2336) requires that a  
127369        <dollar-sign> be quoted to represent itself, so the "\$~" combination is already unspecified.)

127370        The results of giving <tilde> with an unknown login name are undefined because the KornShell  
127371        "~+" and "~-" constructs make use of this condition, but in general it is an error to give an  
127372        incorrect login name with <tilde>. The results of having *HOME* unset are unspecified because  
127373        some historical shells treat this as an error.

127374        Historically, the Korn shell performed field splitting and pathname expansion on the results of  
127375        tilde expansion, and earlier versions of this standard reflected this. However, tilde expansion  
127376        results in a pathname, and performing field splitting and pathname expansion on something  
127377        that is already a pathname is at best redundant and at worst will change the value from the  
127378        correct pathname to one or more incorrect ones. Later versions of the Korn shell do not perform  
127379        these expansions and POSIX.1-2008 has been updated to match. Note that although pathname  
127380        expansion is not performed on the results of tilde expansion, this does not prevent other parts of  
127381        the same word from being expanded. For example, ~/a\* expands to all files in *\$HOME*  
127382        beginning with 'a'.

#### 127383 C.2.6.2 *Parameter Expansion*

127384        The rule for finding the closing '}' in "\${...}" is the one used in the KornShell and is  
127385        upwardly-compatible with the Bourne shell, which does not determine the closing '}' until the  
127386        word is expanded. The advantage of this is that incomplete expansions, such as:

127387        \${foo

127388        can be determined during tokenization, rather than during expansion.

127389        For rationale regarding expansion of "\${...}" within double-quotes, see [Section C.2.2.3](#) (on  
127390        page 3706).

127391        The string length and substring capabilities were included because of the demonstrated need for  
127392        them, based on their usage in other shells, such as C shell and KornShell.

127393        Historical versions of the KornShell have not performed tilde expansion on the word part of  
127394        parameter expansion; however, it is more consistent to do so.

127395        POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0004 [458], XCU/TC1-2008/0005  
127396        [458], XCU/TC1-2008/0006 [457], XCU/TC1-2008/0007 [457], XCU/TC1-2008/0008 [417],  
127397        XCU/TC1-2008/0009 [457], XCU/TC1-2008/0010 [457], XCU/TC1-2008/0011 [457],  
127398        XCU/TC1-2008/0012 [457], XCU/TC1-2008/0013 [457], XCU/TC1-2008/0014 [457],  
127399        XCU/TC1-2008/0015 [457], XCU/TC1-2008/0016 [457], XCU/TC1-2008/0017 [457], and  
127400        XCU/TC1-2008/0018 [458] are applied.

127401 C.2.6.3 Command Substitution

127402 The "\$ ( )" form of command substitution solves a problem of inconsistent behavior when using  
 127403 backquotes. For example:

127404	Command	Output
127405	echo `\\$x`	\\$x
127406	echo `echo `\\$x``	\$x
127407	echo \$(echo `\\$x`)	\\$x

127408 Additionally, the backquoted syntax has historical restrictions on the contents of the embedded  
 127409 command. While the newer "\$ ( )" form can process any kind of valid embedded script, the  
 127410 backquoted form cannot handle some valid scripts that include backquotes. For example, these  
 127411 otherwise valid embedded scripts do not work in the left column, but do work on the right:

127412	echo `	echo \$(
127413	cat <<\eof	cat <<\eof
127414	a here-doc with `	a here-doc with )
127415	eof	eof
127416	`	)
127417	echo `	echo \$(
127418	echo abc # a comment with `	echo abc # a comment with )
127419	`	)
127420	echo `	echo \$(
127421	echo ` `	echo ` )`
127422	`	)

127423 Because of these inconsistent behaviors, the backquoted variety of command substitution is not  
 127424 recommended for new applications that nest command substitutions or attempt to embed  
 127425 complex scripts.

127426 The KornShell feature:

127427 If *command* is of the form <word>, word is expanded to generate a pathname, and the value  
 127428 of the command substitution is the contents of this file with any trailing <newline>  
 127429 characters deleted.

127430 was omitted from the Shell and Utilities volume of POSIX.1-2008 because \$(cat word) is an  
 127431 appropriate substitute. However, to prevent breaking numerous scripts relying on this feature, it  
 127432 is unspecified to have a script within "\$ ( )" that has only redirections.

127433 Arithmetic expansions have precedence over command substitutions. That is, if the shell can  
 127434 parse an expansion beginning with "\$ ( ( )" as an arithmetic expansion then it will do so. It will  
 127435 only parse the expansion as a command substitution (that starts with a subshell) if it determines  
 127436 that it cannot parse the expansion as an arithmetic expansion. If the syntax is valid for neither  
 127437 type of expansion, then it is unspecified what kind of syntax error the shell reports.

127438 How well the shell performs this determination is a quality of implementation issue. Current  
 127439 shell implementations use heuristics. In particular, the shell need not evaluate nested expansions  
 127440 when determining whether it can parse an expansion beginning with "\$ ( ( )" as an arithmetic  
 127441 expansion. For example:

127442 \$( (a \$op b) )

127443 is always an arithmetic expansion if "\$op" expands to, say, '+', but if "\$op" expands to '( '  
 127444 then the shell might still parse the expansion as an arithmetic expansion (resulting in a syntax  
 127445 error due to unbalanced parentheses) or it might perform a command substitution.

127446 This standard requires that conforming applications always separate the "\$(" and '( ' with  
127447 white space when a command substitution starts with a subshell. This is because  
127448 implementations may support extensions in arithmetic expressions which could result in the  
127449 shell parsing the input as an arithmetic expansion even though a minimally conforming shell  
127450 would not. For example, many shells support arrays with the array index (which can be an  
127451 expression) in square brackets. Therefore, the presence of "myfile[0-9]" within an expansion  
127452 beginning "\$(" is no guarantee that it will be parsed as a command substitution.

127453 The ambiguity is not restricted to the simple case of a single subshell. More complicated  
127454 ambiguous cases are possible (even with just the standard shell syntax), such as:

```
127455 $(( cat <<EOH  
127456 + ( (  
127457 EOH  
127458 ) && ( cat <<EOH  
127459 ) ) + 1 +  
127460 EOH  
127461 ) )
```

127462 This can be parsed as an arithmetic expansion, with *cat* and *EOH* as the names of shell variables.  
127463 Ambiguous cases also exist where the end of the expansion is at a different location for the  
127464 arithmetic expansion and the command substitution:

```
127465 $(( cat <<EOF  
127466 + ( ( ( (  
127467 EOF  
127468 ) && (  
127469 cat <<EOF  
127470 +  
127471 EOF  
127472 ) )
```

127473 This is an incomplete arithmetic expansion, but would have been a (complete) command  
127474 substitution if it could not have been parsed as an arithmetic expansion. If this expansion occurs  
127475 at the end of input then the shell reports a syntax error; it does not parse it as a command  
127476 substitution.

127477 IEEE Std 1003.1-2001/Cor 1-2002, item XCU/TC1/D6/4 is applied, changing the text from: "If a  
127478 command substitution occurs inside double-quotes, it shall not be performed on the results of  
127479 the substitution." to: "If a command substitution occurs inside double-quotes, field splitting and  
127480 pathname expansion shall not be performed on the results of the substitution.". The  
127481 replacement text taken from the ISO POSIX-2:1993 standard is clearer about the items that are  
127482 not performed.

127483 SD5-XCU-ERN-84 is applied, clarifying how the search for the matching backquote is satisfied.

127484 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0019 [217] is applied.

#### 127485 C.2.6.4 *Arithmetic Expansion*

127486 The standard developers agreed that there was a strong desire for some kind of arithmetic  
127487 evaluator to provide functionality similar to *expr*, that relating it to '\$ ' makes it work well with  
127488 the standard shell language and provides access to arithmetic evaluation in places where  
127489 accessing a utility would be inconvenient.

127490 The syntax and semantics for arithmetic were revised for the ISO/IEC 9945-2:1993 standard.  
127491 The language represents a simple subset of the previous arithmetic language (which was

derived from the KornShell "`(( ))`" construct). The syntax was changed from that of a command denoted by `((expression))` to an expansion denoted by `$(expression)`. The new form is a dollar expansion (`'$'`) that evaluates the expression and substitutes the resulting value. Objections to the previous style of arithmetic included that it was too complicated, did not fit in well with the use of variables in the shell, and its syntax conflicted with subshells. The justification for the new syntax is that the shell is traditionally a macro language, and if a new feature is to be added, it should be accomplished by extending the capabilities presented by the current model of the shell, rather than by inventing a new one outside the model; adding a new dollar expansion was perceived to be the most intuitive and least destructive way to add such a new capability.

The standard requires assignment operators to be supported (as listed in XCU [Section 1.1.2](#), on page 2321), and since arithmetic expansions are not specified to be evaluated in a subshell environment, changes to variables there have to be in effect after the arithmetic expansion, just as in the parameter expansion `"${x=value}"`.

Note, however, that `"$( ( x=5 ) )"` need not be equivalent to `"$( ( $x=5 ) )"`. If the value of the environment variable `x` is the string `"y="`, the expansion of `"$( ( x=5 ) )"` would set `x` to 5 and output 5, but `"$( ( $x=5 ) )"` would output 0 if the value of the environment variable `y` is not 5 and would output 1 if the environment variable `y` is 5. Similarly, if the value of the environment variable is 4, the expansion of `"$( ( x=5 ) )"` would still set `x` to 5 and output 5, but `"$( ( $x=5 ) )"` (which would be equivalent to `"$( ( 4=5 ) )"`) would yield a syntax error.

In early proposals, a form `$(expression)` was used. It was functionally equivalent to the `"$( )"` of the current text, but objections were lodged that the 1988 KornShell had already implemented `"$( )"` and there was no compelling reason to invent yet another syntax. Furthermore, the `"$[ ]"` syntax had a minor incompatibility involving the patterns in **case** statements.

The portion of the ISO C standard arithmetic operations selected corresponds to the operations historically supported in the KornShell. In addition to the exceptions listed in XCU [Section 2.6.4](#) (on page 2348), the use of the following are explicitly outside the scope of the rules defined in XCU [Section 1.1.2.1](#) (on page 2321):

- The prefix operator `'&'` and the `"[ ]"`, `"->"`, and `'.''` operators.
- Casts

It was concluded that the `test` command (`I`) was sufficient for the majority of relational arithmetic tests, and that tests involving complicated relational expressions within the shell are rare, yet could still be accommodated by testing the value of `"$( )"` itself. For example:

```
# a complicated relational expression
while [ $( ( ($x + $y)/($a * $b)) < ($foo*$bar) ) -ne 0 ]
```

or better yet, the rare script that has many complex relational expressions could define a function like this:

```
val() {
    return $((!$1))
}
```

and complicated tests would be less intimidating:

```
while val $( ( ($x + $y)/($a * $b)) < ($foo*$bar) )
do
    # some calculations
done
```

127538 A suggestion that was not adopted was to modify *true* and *false* to take an optional argument,  
127539 and *true* would exit true only if the argument was non-zero, and *false* would exit false only if the  
127540 argument was non-zero:

```
127541 while true $(( $x > 5 && $y <= 25 ))
```

127542 There is a minor portability concern with the new syntax. The example "`$((2+2))`" could have  
127543 been intended to mean a command substitution of a utility named "2+2" in a subshell. The  
127544 standard developers considered this to be obscure and isolated to some KornShell scripts  
127545 (because "`$()`" command substitution existed previously only in the KornShell). The text on  
127546 command substitution requires that the "`$("`" and "`'(`" be separate tokens if this usage is  
127547 needed.

127548 An example such as:

```
127549 echo $((echo hi);(echo there))
```

127550 should not be misinterpreted by the shell as arithmetic because attempts to balance the  
127551 parentheses pairs would indicate that they are subshells. However, as indicated by XBD [Section](#)  
127552 [3.113](#) (on page 51), a conforming application must separate two adjacent parentheses with white  
127553 space to indicate nested subshells.

127554 The standard is intentionally silent about how a variable's numeric value in an expression is  
127555 determined from its normal "sequence of bytes" value. It could be done as a text substitution, as  
127556 a conversion like that performed by *strtol()*, or even recursive evaluation. Therefore, the only  
127557 cases for which the standard is clear are those for which both conversions produce the same  
127558 result. The cases where they give the same result are those where the sequence of bytes form a  
127559 valid integer constant. Therefore, if a variable does not contain a valid integer constant, the  
127560 behavior is unspecified.

127561 For the commands:

```
127562 x=010; echo $((x += 1))
```

127563 the output must be 9.

127564 For the commands:

```
127565 x=' 1'; echo $((x += 1))
```

127566 the results are unspecified.

127567 For the commands:

```
127568 x=1+1; echo $((x += 1))
```

127569 the results are unspecified.

127570 Although the ISO/IEC 9899:1999 standard now requires support for **long long** and allows  
127571 extended integer types with higher ranks, POSIX.1-2008 only requires arithmetic expansions to  
127572 support **signed long** integer arithmetic. Implementations are encouraged to support signed  
127573 integer values at least as large as the size of the largest file allowed on the implementation.

127574 Implementations are also allowed to perform floating-point evaluations as long as an  
127575 application won't see different results for expressions that would not overflow **signed long**  
127576 integer expression evaluation. (This includes appropriate truncation of results to integer values.)

127577 Changes made in response to IEEE PASC Interpretation 1003.2 #208 removed the requirement  
127578 that the integer constant suffixes `l` and `L` had to be recognized. The ISO POSIX-2: 1993 standard  
127579 did not require the `u`, `uL`, `uL`, `U`, `UL`, `UL`, `Lu`, `LU`, `Lu`, and `LU` suffixes since only signed integer  
127580 arithmetic was required. Since all arithmetic expressions were treated as handling **signed long**  
127581 integer types anyway, the `l` and `L` suffixes were redundant. No known scripts used them and

127582 some historic shells did not support them. When the ISO/IEC 9899:1999 standard was used as  
127583 the basis for the description of arithmetic processing, the `ll` and `LL` suffixes and combinations  
127584 were also not required. Implementations are still free to accept any or all of these suffixes, but  
127585 are not required to do so.

127586 There was also some confusion as to whether the shell was required to recognize character  
127587 constants. Syntactically, character constants were required to be recognized, but the  
127588 requirements for the handling of `<backslash>` and single-quote characters (needed to specify  
127589 character constants) within an arithmetic expansion were ambiguous. Furthermore, no known  
127590 shells supported them. Changes made in response to IEEE PASC Interpretation 1003.2 #208  
127591 removed the requirement to support them (if they were indeed required before). POSIX.1-2008  
127592 clearly does not require support for character constants.

127593 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/3 is applied, clarifying arithmetic  
127594 expressions.

127595 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0020 [50] is applied.

#### 127596 C.2.6.5 *Field Splitting*

127597 The operation of field splitting using *IFS*, as described in early proposals, was based on the way  
127598 the KornShell splits words, but it is incompatible with other common versions of the shell.  
127599 However, each has merit, and so a decision was made to allow both. If the *IFS* variable is unset  
127600 or is `<space><tab><newline>`, the operation is equivalent to the way the System V shell splits  
127601 words. Using characters outside the `<space><tab><newline>` set yields the KornShell behavior,  
127602 where each of the non-`<space><tab><newline>`s is significant. This behavior, which affords the  
127603 most flexibility, was taken from the way the original *awk* handled field splitting.

127604 Rule (3) can be summarized as a pseudo-ERE:

127605  $(s^*ns^* | s^+)$

127606 where *s* is an *IFS* white-space character and *n* is a character in the *IFS* that is not white space.  
127607 Any string matching that ERE delimits a field, except that the *s+* form does not delimit fields at  
127608 the beginning or the end of a line. For example, if *IFS* is `<space>/<comma>/<tab>`, the string:

127609 `<space><space>red<space><space>, <space>white<space>blue`

127610 yields the three colors as the delimited fields.

#### 127611 C.2.6.6 *Pathname Expansion*

127612 There is no additional rationale provided for this section.

#### 127613 C.2.6.7 *Quote Removal*

127614 There is no additional rationale provided for this section.



## 127615 C.2.7 Redirection

127616 In the System Interfaces volume of POSIX.1-2008, file descriptors are integers in the range  
127617 0–(OPEN\_MAX–1). The file descriptors discussed in XCU [Section 2.7](#) (on page 2350) are that  
127618 same set of small integers.

127619 Having multi-digit file descriptor numbers for I/O redirection can cause some obscure  
127620 compatibility problems. Specifically, scripts that depend on an example command:

127621 `echo 22>/dev/null`

127622 echoing "2" to standard error or "22" to standard output are no longer portable. However, the  
127623 file descriptor number must still be delimited from the preceding text. For example:

127624 `cat file2>foo`

127625 writes the contents of **file2**, not the contents of **file**.

127626 The ">|" format of output redirection was adopted from the KornShell. Along with the  
127627 *noclobber* option, set **–C**, it provides a safety feature to prevent inadvertent overwriting of  
127628 existing files. (See the RATIONALE for the *pathchk* utility for why this step was taken.) The  
127629 restriction on regular files is historical practice.

127630 The System V shell and the KornShell have differed historically on pathname expansion of *word*;  
127631 the former never performed it, the latter only when the result was a single field (file). As a  
127632 compromise, it was decided that the KornShell functionality was useful, but only as a shorthand  
127633 device for interactive users. No reasonable shell script would be written with a command such  
127634 as:

127635 `cat foo > a*`

127636 Thus, shell scripts are prohibited from doing it, while interactive users can select the shell with  
127637 which they are most comfortable.

127638 The construct "2>&1" is often used to redirect standard error to the same file as standard  
127639 output. Since the redirections take place beginning to end, the order of redirections is significant.  
127640 For example:

127641 `ls > foo 2>&1`

127642 directs both standard output and standard error to file **foo**. However:

127643 `ls 2>&1 > foo`

127644 only directs standard output to file **foo** because standard error was duplicated as standard  
127645 output before standard output was directed to file **foo**.

127646 Applications should not use the `[n]<&–` or `[n]>&–` operators to execute a utility or application  
127647 with file descriptor 0 not open for reading or with file descriptor 1 or 2 not open for writing, as  
127648 this might cause the executed program (or shell built-in) to misbehave. In order not to pass on  
127649 these file descriptors to an executed utility or application, applications should not just close  
127650 them but should reopen them on, for example, **/dev/null**. Some implementations may reopen  
127651 them automatically, but applications should not rely on this being done.

127652 The "<>" operator could be useful in writing an application that worked with several terminals,  
127653 and occasionally wanted to start up a shell. That shell would in turn be unable to run  
127654 applications that run from an ordinary controlling terminal unless it could make use of "<>"  
127655 redirection. The specific example is a historical version of the pager *more*, which reads from  
127656 standard error to get its commands, so standard input and standard output are both available  
127657 for their usual usage. There is no way of saying the following in the shell without "<>":

127658 `cat food | more – >/dev/tty03 2<>/dev/tty03`

127659 Another example of "<>" is one that opens **/dev/tty** on file descriptor 3 for reading and writing:

127660 `exec 3<> /dev/tty`

127661 An example of creating a lock file for a critical code region:

127662 `set -C`  
127663 `until 2> /dev/null > lockfile`  
127664 `do sleep 30`  
127665 `done`  
127666 `set +C`  
127667 *perform critical function*  
127668 `rm lockfile`

127669 Since **/dev/null** is not a regular file, no error is generated by redirecting to it in *noclobber* mode.

127670 Tilde expansion is not performed on a here-document because the data is treated as if it were  
127671 enclosed in double-quotes.

127672 C.2.7.1 *Redirecting Input*

127673 There is no additional rationale provided for this section.

127674 C.2.7.2 *Redirecting Output*

127675 There is no additional rationale provided for this section.

127676 C.2.7.3 *Appending Redirected Output*

127677 Note that when a file is opened (even with the `O_APPEND` flag set), the initial file offset for that  
127678 file is set to the beginning of the file. Some historic shells set the file offset to the current end-of-  
127679 file when **append** mode shell redirection was used, but this is not allowed by POSIX.1-2008.

127680 C.2.7.4 *Here-Document*

127681 There is no additional rationale provided for this section.

127682 C.2.7.5 *Duplicating an Input File Descriptor*

127683 There is no additional rationale provided for this section.

127684 C.2.7.6 *Duplicating an Output File Descriptor*

127685 There is no additional rationale provided for this section.

127686 C.2.7.7 *Open File Descriptors for Reading and Writing*

127687 There is no additional rationale provided for this section.



127688 **C.2.8 Exit Status and Errors**

127689 There is no additional rationale provided for this section.

127690 C.2.8.1 *Consequences of Shell Errors*

127691 There is no additional rationale provided for this section.

127692 C.2.8.2 *Exit Status for Commands*

127693 There is a historical difference in *sh* and *ksh* non-interactive error behavior. When a command  
127694 named in a script is not found, some implementations of *sh* exit immediately, but *ksh* continues  
127695 with the next command. Thus, the Shell and Utilities volume of POSIX.1-2008 says that the shell  
127696 “may” exit in this case. This puts a small burden on the programmer, who has to test for  
127697 successful completion following a command if it is important that the next command not be  
127698 executed if the previous command was not found. If it is important for the command to have  
127699 been found, it was probably also important for it to complete successfully. The test for successful  
127700 completion would not need to change.

127701 Historically, shells have returned an exit status of  $128+n$ , where  $n$  represents the signal number.  
127702 Since signal numbers are not standardized, there is no portable way to determine which signal  
127703 caused the termination. Also, it is possible for a command to exit with a status in the same range  
127704 of numbers that the shell would use to report that the command was terminated by a signal.  
127705 Implementations are encouraged to choose exit values greater than 256 to indicate programs that  
127706 terminate by a signal so that the exit status cannot be confused with an exit status generated by a  
127707 normal termination.

127708 Historical shells make the distinction between “utility not found” and “utility found but cannot  
127709 execute” in their error messages. By specifying two seldomly used exit status values for these  
127710 cases, 127 and 126 respectively, this gives an application the opportunity to make use of this  
127711 distinction without having to parse an error message that would probably change from locale to  
127712 locale. The *command*, *env*, *nohup*, and *xargs* utilities in the Shell and Utilities volume of  
127713 POSIX.1-2008 have also been specified to use this convention.

127714 When a command fails during word expansion or redirection, most historical implementations  
127715 exit with a status of 1. However, there was some sentiment that this value should probably be  
127716 much higher so that an application could distinguish this case from the more normal exit status  
127717 values. Thus, the language “greater than zero” was selected to allow either method to be  
127718 implemented.

127719 **C.2.9 Shell Commands**

127720 A description of an “empty command” was removed from an early proposal because it is only  
127721 relevant in the cases of *sh -c " "*, *system(" ")*, or an empty shell-script file (such as the  
127722 implementation of *true* on some historical systems). Since it is no longer mentioned in the Shell  
127723 and Utilities volume of POSIX.1-2008, it falls into the silently unspecified category of behavior  
127724 where implementations can continue to operate as they have historically, but conforming  
127725 applications do not construct empty commands. (However, note that *sh* does explicitly state an  
127726 exit status for an empty string or file.) In an interactive session or a script with other commands,  
127727 extra <newline> or <semicolon> characters, such as:

127728 \$ false  
127729 \$  
127730 \$ echo \$?

127731 1  
127732 would not qualify as the empty command described here because they would be consumed by  
127733 other parts of the grammar.

127734 C.2.9.1 *Simple Commands*

127735 The enumerated list is used only when the command is actually going to be executed. For  
127736 example, in:

127737 `true || $foo *`

127738 no expansions are performed.

127739 The following example illustrates both how a variable assignment without a command name  
127740 affects the current execution environment, and how an assignment with a command name only  
127741 affects the execution environment of the command:

```
127742 $ x=red
127743 $ echo $x
127744 red
127745 $ export x
127746 $ sh -c 'echo $x'
127747 red
127748 $ x=blue sh -c 'echo $x'
127749 blue
127750 $ echo $x
127751 red
```

127752 This next example illustrates that redirections without a command name are still performed:

```
127753 $ ls foo
127754 ls: foo: no such file or directory
127755 $ > foo
127756 $ ls foo
127757 foo
```

127758 A command without a command name, but one that includes a command substitution, has an  
127759 exit status of the last command substitution that the shell performed. For example:

```
127760 if      x=$(command)
127761 then    ...
127762 fi
```

127763 An example of redirections without a command name being performed in a subshell shows that  
127764 the here-document does not disrupt the standard input of the **while** loop:

```
127765 IFS=:
127766 while read a b
127767 do     echo $a
127768       <<-eof
127769       Hello
127770       eof
127771 done </etc/passwd
```

127772 Following are examples of commands without command names in AND-OR lists:

```
127773 > foo || {
127774     echo "error: foo cannot be created" >&2
```

```
127775         exit 1
127776     }
127777     # set saved if /vmunix.save exists
127778     test -f /vmunix.save && saved=1
```

127779 Command substitution and redirections without command names both occur in subshells, but  
127780 they are not necessarily the same ones. For example, in:

```
127781     exec 3> file
127782     var=$(echo foo >&3) 3>&1
```

127783 it is unspecified whether **foo** is echoed to the file or to standard output.

127784 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0021 [255] is applied.

## 127785 Command Search and Execution

127786 This description requires that the shell can execute shell scripts directly, even if the underlying  
127787 system does not support the common "#!" interpreter convention. That is, if file **foo** contains  
127788 shell commands and is executable, the following executes **foo**:

```
127789 ./foo
```

127790 The command search shown here does not match all historical implementations. A more typical  
127791 sequence has been:

- 127792 • Any built-in (special or regular)
- 127793 • Functions
- 127794 • Path search for executable files

127795 But there are problems with this sequence. Since the programmer has no idea in advance which  
127796 utilities might have been built into the shell, a function cannot be used to override portably a  
127797 utility of the same name. (For example, a function named *cd* cannot be written for many  
127798 historical systems.) Furthermore, the *PATH* variable is partially ineffective in this case, and only  
127799 a pathname with a <slash> can be used to ensure a specific executable file is invoked.

127800 After the *execve()* failure described, the shell normally executes the file as a shell script. Some  
127801 implementations, however, attempt to detect whether the file is actually a script and not an  
127802 executable from some other architecture. The method used by the KornShell is allowed by the  
127803 text that indicates non-text files may be bypassed.

127804 The sequence selected for the Shell and Utilities volume of POSIX.1-2008 acknowledges that  
127805 special built-ins cannot be overridden, but gives the programmer full control over which  
127806 versions of other utilities are executed. It provides a means of suppressing function lookup (via  
127807 the *command* utility) for the user's own functions and ensures that any regular built-ins or  
127808 functions provided by the implementation are under the control of the path search. The  
127809 mechanisms for associating built-ins or functions with executable files in the path are not  
127810 specified by the Shell and Utilities volume of POSIX.1-2008, but the wording requires that if  
127811 either is implemented, the application is not able to distinguish a function or built-in from an  
127812 executable (other than in terms of performance, presumably). The implementation ensures that  
127813 all effects specified by the Shell and Utilities volume of POSIX.1-2008 resulting from the  
127814 invocation of the regular built-in or function (interaction with the environment, variables, traps,  
127815 and so on) are identical to those resulting from the invocation of an executable file.

127816 Various historical implementations have used the names in item 1.b. as built-ins or reserved +  
127817 words. This standard does not specify their behavior, but their existence means that it is +  
127818 important for portable applications to avoid giving functions (or utilities in *PATH*) those names +

127819 because the function (or utility in *PATH*) might not be executed as expected. +  
 127820 IEEE Std 1003.1-2001/Cor 2-2004, item XCU/TC2/D6/4 is applied, updating the case where  
 127821 *execve()* fails due to an error equivalent to the [ENOEXEC] error.

127822 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0022 [168], XCU/TC1-2008/0023  
 127823 [168], XCU/TC1-2008/0024 [168], XCU/TC1-2008/0025 [168], XCU/TC1-2008/0026 [168,430],  
 127824 XCU/TC1-2008/0027 [168,430], and XCU/TC1-2008/0028 [173] are applied.

## 127825 Examples

127826 Consider three versions of the *ls* utility:

- 127827 1. The application includes a shell function named *ls*.
- 127828 2. The user writes a utility named *ls* and puts it in **/fred/bin**.
- 127829 3. The example implementation provides *ls* as a regular shell built-in that is invoked (either  
 127830 by the shell or directly by *exec*) when the path search reaches the directory **/posix/bin**.

127831 If *PATH*=**/posix/bin**, various invocations yield different versions of *ls*:

Invocation	Version of <i>ls</i>
<i>ls</i> (from within application script)	(1) function
<i>command ls</i> (from within application script)	(3) built-in
<i>ls</i> (from within makefile called by application)	(3) built-in
<i>system("ls")</i>	(3) built-in
<i>PATH="/fred/bin:\$PATH" ls</i>	(2) user's version

## 127838 C.2.9.2 Pipelines

127839 Because pipeline assignment of standard input or standard output or both takes place before  
 127840 redirection, it can be modified by redirection. For example:

127841 **\$ command1 2>&1 | command2**

127842 sends both the standard output and standard error of *command1* to the standard input of  
 127843 *command2*.

127844 The reserved word **!** allows more flexible testing using AND and OR lists. The behavior of **!(** is +  
 127845 unspecified because in the Korn Shell this introduces a negated pathname expansion. Portable +  
 127846 applications need to separate the **!** and **(** to ensure the command is treated as a negated subshell.

127847 It was suggested that it would be better to return a non-zero value if any command in the  
 127848 pipeline terminates with non-zero status (perhaps the bitwise-inclusive OR of all return values).  
 127849 However, the choice of the last-specified command semantics are historical practice and would  
 127850 cause applications to break if changed. An example of historical behavior:

127851 **\$ sleep 5 | (exit 4)**  
 127852 **\$ echo \$?**  
 127853 **4**  
 127854 **\$ (exit 4) | sleep 5**  
 127855 **\$ echo \$?**  
 127856 **0**

127857 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0029 [205] is applied.

## Exit Status

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0030 [52] is applied.

### C.2.9.3 Lists

The equal precedence of "&&" and "||" is historical practice. The standard developers evaluated the model used more frequently in high-level programming languages, such as C, to allow the shell logical operators to be used for complex expressions in an unambiguous way, but they could not allow historical scripts to break in the subtle way unequal precedence might cause. Some arguments were posed concerning the "{" or "(" groupings that are required historically. There are some disadvantages to these groupings:

- The "(" can be expensive, as they spawn other processes on some implementations. This performance concern is primarily an implementation issue.
- The "{" braces are not operators (they are reserved words) and require a trailing <space> after each '{', and a <semicolon> before each '}'. Most programmers (and certainly interactive users) have avoided braces as grouping constructs because of the problematic syntax required. Braces were not changed to operators because that would generate compatibility issues even greater than the precedence question; braces appear outside the context of a keyword in many shell scripts.

IEEE PASC Interpretation 1003.2 #204 is applied, clarifying that the operators "&&" and "||" are evaluated with left associativity.

POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0031 [45] and XCU/TC1-2008/0032 [45] are applied.

## Asynchronous Lists

The grammar treats a construct such as:

```
foo & bar & bam &
```

as one "asynchronous list", but since the status of each element is tracked by the shell, the term "element of an asynchronous list" was introduced to identify just one of the **foo**, **bar**, or **bam** portions of the overall list.

Unless the implementation has an internal limit, such as {CHILD\_MAX}, on the retained process IDs, it would require unbounded memory for the following example:

```
while true
do    foo & echo $!
done
```

The treatment of the signals SIGINT and SIGQUIT with asynchronous lists is described in XCU [Section 2.11](#) (on page 2371).

Since the connection of the input to the equivalent of **/dev/null** is considered to occur before redirections, the following script would produce no output:

```
exec < /etc/passwd
cat <&0 &
wait
```

## 127897 Sequential Lists

127898 There is no additional rationale provided for this section.

## 127899 AND Lists

127900 There is no additional rationale provided for this section.

## 127901 OR Lists

127902 There is no additional rationale provided for this section.

## 127903 C.2.9.4 Compound Commands

### 127904 Grouping Commands

127905 The semicolon shown in *{compound-list;}* is an example of a control operator delimiting the }  
127906 reserved word. Other delimiters are possible, as shown in XCU [Section 2.10](#) (on page 2365);  
127907 <newline> is frequently used.

127908 A proposal was made to use the <do-done> construct in all cases where command grouping in  
127909 the current process environment is performed, identifying it as a construct for the grouping  
127910 commands, as well as for shell functions. This was not included because the shell already has a  
127911 grouping construct for this purpose ("{"}), and changing it would have been counter-  
127912 productive.

127913 The requirement for conforming applications to separate two leading ' ' characters with white  
127914 space if a grouping command would be parsed as an arithmetic expansion if preceded by a '\$'  
127915 is to allow shells which implement the "(( arithmetic expression ))" extension to  
127916 apply the same disambiguation rules consistently to \$(...) and ((...)). See [Section](#)  
127917 [C.2.6.3](#) (on page 3718).

127918 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0033 [217] is applied.

### 127919 For Loop

127920 The format is shown with generous usage of <newline> characters. See the grammar in XCU  
127921 [Section 2.10](#) (on page 2365) for a precise description of where <newline> and <semicolon>  
127922 characters can be interchanged.

127923 Some historical implementations support '{' and '}' as substitutes for **do** and **done**. The  
127924 standard developers chose to omit them, even as an obsolescent feature. (Note that these  
127925 substitutes were only for the **for** command; the **while** and **until** commands could not use them  
127926 historically because they are followed by compound-lists that may contain "{...}" grouping  
127927 commands themselves.)

127928 The reserved word pair **do ... done** was selected rather than **do ... od** (which would have  
127929 matched the spirit of **if ... fi** and **case ... esac**) because *od* is already the name of a standard  
127930 utility.

127931 PASC Interpretation 1003.2 #169 has been applied changing the grammar.



## Case Conditional Construct

An optional <left-parenthesis> before *pattern* was added to allow numerous historical KornShell scripts to conform. At one time, using the leading parenthesis was required if the **case** statement was to be embedded within a "\$ ( )" command substitution; this is no longer the case with the POSIX shell. Nevertheless, many historical scripts use the <left-parenthesis>, if only because it makes matching-parenthesis searching easier in *vi* and other editors. This is a relatively simple implementation change that is upwards-compatible for all scripts.

Consideration was given to requiring *break* inside the *compound-list* to prevent falling through to the next pattern action list. This was rejected as being nonexistent practice. An interesting undocumented feature of the KornShell is that using ";" instead of ";;" as a terminator causes the exact opposite behavior—the flow of control continues with the next *compound-list*.

The pattern '\*', given as the last pattern in a **case** construct, is equivalent to the default case in a C-language **switch** statement.

The grammar shows that reserved words can be used as patterns, even if one is the first word on a line. Obviously, the reserved word **esac** cannot be used in this manner.

## If Conditional Construct

The precise format for the command syntax is described in XCU [Section 2.10](#) (on page 2365).

## While Loop

The precise format for the command syntax is described in XCU [Section 2.10](#) (on page 2365).

## Until Loop

The precise format for the command syntax is described in XCU [Section 2.10](#) (on page 2365).

### C.2.9.5 Function Definition Command

The description of functions in an early proposal was based on the notion that functions should behave like miniature shell scripts; that is, except for sharing variables, most elements of an execution environment should behave as if they were a new execution environment, and changes to these should be local to the function. For example, traps and options should be reset on entry to the function, and any changes to them do not affect the traps or options of the caller. There were numerous objections to this basic idea, and the opponents asserted that functions were intended to be a convenient mechanism for grouping common commands that were to be executed in the current execution environment, similar to the execution of the *dot* special built-in.

It was also pointed out that the functions described in that early proposal did not provide a local scope for everything a new shell script would, such as the current working directory, or *umask*, but instead provided a local scope for only a few select properties. The basic argument was that if a local scope is needed for the execution environment, the mechanism already existed: the application can put the commands in a new shell script and call that script. All historical shells that implemented functions, other than the KornShell, have implemented functions that operate in the current execution environment. Because of this, traps and options have a global scope within a shell script. Local variables within a function were considered and included in another early proposal (controlled by the special built-in *local*), but were removed because they do not fit the simple model developed for functions and because there was some opposition to adding yet another new special built-in that was not part of historical practice. Implementations should reserve the identifier *local* (as well as *typeset*, as used in the KornShell) in case this local variable mechanism is adopted in a future version of this standard.

A separate issue from the execution environment of a function is the availability of that function to child shells. A few objectors maintained that just as a variable can be shared with child shells by exporting it, so should a function. In early proposals, the *export* command therefore had a *-f* flag for exporting functions. Functions that were exported were to be put into the environment as *name( )=value* pairs, and upon invocation, the shell would scan the environment for these and automatically define these functions. This facility was strongly opposed and was omitted. Some of the arguments against exportable functions were as follows:

- There was little historical practice. The Ninth Edition shell provided them, but there was controversy over how well it worked.
- There are numerous security problems associated with functions appearing in the environment of a user and overriding standard utilities or the utilities owned by the application.
- There was controversy over requiring *make* to import functions, where it has historically used an *exec* function for many of its command line executions.
- Functions can be big and the environment is of a limited size. (The counter-argument was that functions are no different from variables in terms of size: there can be big ones, and there can be small ones—and just as one does not export huge variables, one does not export huge functions. However, this might not apply to the average shell-function writer, who typically writes much larger functions than variables.)

As far as can be determined, the functions in the Shell and Utilities volume of POSIX.1-2008 match those in System V. Earlier versions of the KornShell had two methods of defining functions:

```
function fname { compound-list }
```

and:

```
fname() { compound-list }
```

The latter used the same definition as the Shell and Utilities volume of POSIX.1-2008, but differed in semantics, as described previously. The current edition of the KornShell aligns the latter syntax with the Shell and Utilities volume of POSIX.1-2008 and keeps the former as is.

The name space for functions is limited to that of a *name* because of historical practice. Complications in defining the syntactic rules for the function definition command and in dealing with known extensions such as the *"@()"* usage in the KornShell prevented the name space from being widened to a *word*. Using functions to support synonyms such as the *"!!"* and *'%'* usage in the C shell is thus disallowed to conforming applications, but acceptable as an extension. For interactive users, the aliasing facilities in the Shell and Utilities volume of POSIX.1-2008 should be adequate for this purpose. It is recognized that the name space for utilities in the file system is wider than that currently supported for functions, if the portable filename character set guidelines are ignored, but it did not seem useful to mandate extensions in systems for so little benefit to conforming applications.

The *"()"* in the function definition command consists of two operators. Therefore, intermixing <blank> characters with the *fname*, *'( '*, and *')* is allowed, but unnecessary.

An example of how a function definition can be used wherever a simple command is allowed:

```
# If variable i is equal to "yes",
# define function foo to be ls -l
#
[ "$i" = yes ] && foo() {
    ls -l
```



128022            }  
128023            POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0034 [383] and XCU/TC1-2008/0035  
128024            [214] are applied.

## 128025 C.2.10 Shell Grammar

128026            There are several subtle aspects of this grammar where conventional usage implies rules about  
128027            the grammar that in fact are not true.

128028            For *compound\_list*, only the forms that end in a *separator* allow a reserved word to be recognized,  
128029            so usually only a *separator* can be used where a compound list precedes a reserved word (such as  
128030            **Then**, **Else**, **Do**, and **Rbrace**). Explicitly requiring a separator would disallow such valid (if rare)  
128031            statements as:

128032            if (false) then (echo x) else (echo y) fi

128033            See the Note under special grammar rule (1).

128034            Concerning the third sentence of rule (1) (“Also, if the parser ...”):

128035            

- This sentence applies rather narrowly: when a compound list is terminated by some clear  
128036            delimiter (such as the closing **fi** of an inner **if\_clause**) then it would apply; where the  
128037            compound list might continue (as in after a ‘;’), rule (7a) (and consequently the first  
128038            sentence of rule (1)) would apply. In many instances the two conditions are identical, but  
128039            this part of rule (1) does not give license to treating a **WORD** as a reserved word unless it  
128040            is in a place where a reserved word has to appear.

128041            

- The statement is equivalent to requiring that when the LR(1) lookahead set contains  
128042            exactly one reserved word, it must be recognized if it is present. (Here “LR(1)” refers to the  
128043            theoretical concepts, not to any real parser generator.)

128044            For example, in the construct below, and when the parser is at the point marked with ‘^’,  
128045            the only next legal token is **then** (this follows directly from the grammar rules):

128046            if if...fi then ... fi  
128047                                 ^

128048            At that point, the **then** must be recognized as a reserved word.

128049            (Depending on the parser generator actually used, “extra” reserved words may be in some  
128050            lookahead sets. It does not really matter if they are recognized, or even if any possible  
128051            reserved word is recognized in that state, because if it is recognized and is not in the  
128052            (theoretical) LR(1) lookahead set, an error is ultimately detected. In the example above, if  
128053            some other reserved word (for example, **while**) is also recognized, an error occurs later.

128054            This is approximately equivalent to saying that reserved words are recognized after other  
128055            reserved words (because it is after a reserved word that this condition occurs), but avoids  
128056            the “except for ...” list that would be required for **case**, **for**, and so on. (Reserved words  
128057            are of course recognized anywhere a *simple\_command* can appear, as well. Other rules take  
128058            care of the special cases of non-recognition, such as rule (4) for **case** statements.)

128059            Note that the body of here-documents are handled by token recognition (see XCU [Section 2.3](#), on  
128060            page 2337) and do not appear in the grammar directly. (However, the here-document I/O  
128061            redirection operator is handled as part of the grammar.)

128062 C.2.10.1 Shell Grammar Lexical Conventions

128063 There is no additional rationale provided for this section.

128064 C.2.10.2 Shell Grammar Rules

128065 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0036 [44] is applied.

128066 **C.2.11 Signals and Error Handling**

128067 Historically, some shell implementations silently ignored attempts to use *trap* to set SIGINT or +  
128068 SIGQUIT to the default action or to set a trap for them after they have been set to be ignored by +  
128069 the shell when it executes an asynchronous subshell (and job control is disabled). This behavior +  
128070 is not conforming. For example, if a shell script containing the following line is run in the +  
128071 foreground at a terminal: +

128072 `(trap - INT; exec sleep 10) & wait` +

128073 and is then terminated by typing the interrupt character, this standard requires that the *sleep* +  
128074 command is terminated by the SIGINT signal. +

128075 SD5-XCU-ERN-93 is applied, updating the first paragraph of XCU [Section 2.11](#) (on page 2371).

128076 **C.2.12 Shell Execution Environment**

128077 Some implementations have implemented the last stage of a pipeline in the current environment  
128078 so that commands such as:

128079 `command | read foo`

128080 set variable **foo** in the current environment. This extension is allowed, but not required;  
128081 therefore, a shell programmer should consider a pipeline to be in a subshell environment, but  
128082 not depend on it.

128083 In early proposals, the description of execution environment failed to mention that each  
128084 command in a multiple command pipeline could be in a subshell execution environment. For  
128085 compatibility with some historical shells, the wording was phrased to allow an implementation  
128086 to place any or all commands of a pipeline in the current environment. However, this means that  
128087 a POSIX application must assume each command is in a subshell environment, but not depend  
128088 on it.

128089 The wording about shell scripts is meant to convey the fact that describing “trap actions” can  
128090 only be understood in the context of the shell command language. Outside of this context, such  
128091 as in a C-language program, signals are the operative condition, not traps.

128092 POSIX.1-2008, Technical Corrigendum 1, XCU/TC1-2008/0037 [238] is applied.

## 128093 C.2.13 Pattern Matching Notation

128094 Pattern matching is a simpler concept and has a simpler syntax than REs, as the former is  
128095 generally used for the manipulation of filenames, which are relatively simple collections of  
128096 characters, while the latter is generally used to manipulate arbitrary text strings of potentially  
128097 greater complexity. However, some of the basic concepts are the same, so this section points  
128098 liberally to the detailed descriptions in XBD [Chapter 9](#) (on page 181).

### 128099 C.2.13.1 Patterns Matching a Single Character

128100 Both quoting and escaping are described here because pattern matching must work in three  
128101 separate circumstances:

- 128102 1. Calling directly upon the shell, such as in pathname expansion or in a **case** statement. All  
128103 of the following match the string or file **abc**:

128104 `abc "abc" a"b" c a\bc a[b]c a["b"]c a[\b]c a["\b"]c a?c a*c`

128105 The following do not:

128106 `"a?c" a*c a\[b]c`

- 128107 2. Calling a utility or function without going through a shell, as described for *find* and the  
128108 *fnmatch()* function defined in the System Interfaces volume of POSIX.1-2008.

- 128109 3. Calling utilities such as *find*, *cpio*, *tar*, or *pax* through the shell command line. In this case,  
128110 shell quote removal is performed before the utility sees the argument. For example, in:

128111 `find /bin -name "e\c[\h]o" -print`

128112 after quote removal, the <backslash> characters are presented to *find* and it treats them as  
128113 escape characters. Both precede ordinary characters, so the *c* and *h* represent themselves  
128114 and *echo* would be found on many historical systems (that have it in **/bin**). To find a  
128115 filename that contained shell special characters or pattern characters, both quoting and  
128116 escaping are required, such as:

128117 `pax -r ... "a\(\?"`

128118 to extract a filename ending with "a(?".

128119 Conforming applications are required to quote or escape the shell special characters (sometimes  
128120 called metacharacters). If used without this protection, syntax errors can result or  
128121 implementation extensions can be triggered. For example, the KornShell supports a series of  
128122 extensions based on parentheses in patterns.

128123 The restriction on a <circumflex> in a bracket expression is to allow implementations that  
128124 support pattern matching using the <circumflex> as the negation character in addition to the  
128125 <exclamation-mark>. A conforming application must use something like "[\^!]" to match  
128126 either character.

### 128127 C.2.13.2 Patterns Matching Multiple Characters

128128 Since each <asterisk> matches zero or more occurrences, the patterns "a\*b" and "a\*\*b" have  
128129 identical functionality.

128130       **Examples**

- |        |       |  |
|--------|-------|--|
| 128131 | a[bc] | Matches the strings "ab" and "ac".                                     |
| 128132 | a*d   | Matches the strings "ad", "abd", and "abcd", but not the string "abc". |
| 128133 | a*d*  | Matches the strings "ad", "abcd", "abcdef", "aaaad", and "adddd".      |
| 128134 | *a*d  | Matches the strings "ad", "abcd", "efabcd", "aaaad", and "adddd".      |

128135   C.2.13.3   *Patterns Used for Filename Expansion*

128136       The caveat about a <slash> within a bracket expression is derived from historical practice. The  
128137       pattern "a[b/c]d" does not match such pathnames as **abd** or **a/d**. On some implementations  
128138       (including those conforming to the Single UNIX Specification), it matched a pathname of  
128139       literally "a[b/c]d". On other systems, it produced an undefined condition (an unescaped '['  
128140       used outside a bracket expression). In this version, the XSI behavior is now required.

128141       Filenames beginning with a <period> historically have been specially protected from view on  
128142       UNIX systems. A proposal to allow an explicit <period> in a bracket expression to match a  
128143       leading <period> was considered; it is allowed as an implementation extension, but a  
128144       conforming application cannot make use of it. If this extension becomes popular in the future, it  
128145       will be considered for a future version of the Shell and Utilities volume of POSIX.1-2008.

128146       Historical systems have varied in their permissions requirements. To match **f\*/bar** has required  
128147       read permissions on the **f\*** directories in the System V shell, but the Shell and Utilities volume of  
128148       POSIX.1-2008, the C shell, and KornShell require only search permissions.

128149   **C.2.14   Special Built-In Utilities**

128150       See the RATIONALE sections on the individual reference pages.

128151   **C.3       Batch Environment Services and Utilities**

128152       **Scope of the Batch Environment Services and Utilities Option**

128153       This section summarizes the deliberations of the IEEE P1003.15 (Batch Environment) working  
128154       group in the development of the Batch Environment Services and Utilities option, which covers  
128155       a set of services and utilities defining a batch processing system.

128156       This informative section contains historical information concerning the contents of the  
128157       amendment and describes why features were included or discarded by the working group.

128158       **History of Batch Systems**

128159       The supercomputing technical committee began as a "Birds Of a Feather" (BOF) at the January  
128160       1987 Usenix meeting. There was enough general interest to form a supercomputing attachment  
128161       to the /usr/group working groups. Several subgroups rapidly formed. Of those subgroups, the  
128162       batch group was the most ambitious. The first early meetings were spent evaluating user needs  
128163       and existing batch implementations.

128164       To evaluate user needs, individuals from the supercomputing community came and presented  
128165       their needs. Common requests were flexibility, interoperability, control of resources, and ease-of-  
128166       use. Backward-compatibility was not an issue. The working group then evaluated some existing  
128167       systems. The following different systems were evaluated:

- 128168 • PROD
- 128169 • Convex Distributed Batch
- 128170 • NQS
- 128171 • CTSS
- 128172 • MDQS from Ballistics Research Laboratory (BRL)

128173 Finally, NQS was chosen as a model because it satisfied not only the most user requirements, but  
 128174 because it was public domain, already implemented on a variety of hardware platforms, and  
 128175 network-based.

## 128176 **Historical Implementations of Batch Systems**

128177 Deferred processing of work under the control of a scheduler has been a feature of most  
 128178 proprietary operating systems from the earliest days of multi-user systems in order to maximize  
 128179 utilization of the computer.

128180 The arrival of UNIX systems proved to be a dilemma to many hardware providers and users  
 128181 because it did not include the sophisticated batch facilities offered by the proprietary systems.  
 128182 This omission was rectified in 1986 by NASA Ames Research Center who developed the  
 128183 Network Queuing System (NQS) as a portable UNIX application that allowed the routing and  
 128184 processing of batch “jobs” in a network. To encourage its usage, the product was later put into  
 128185 the public domain. It was promptly picked up by UNIX hardware providers, and ported and  
 128186 developed for their respective hardware and UNIX implementations.

128187 Many major vendors, who traditionally offer a batch-dominated environment, ported the  
 128188 public-domain product to their systems, customized it to support the capabilities of their  
 128189 systems, and added many customer-requested features.

128190 Due to the strong hardware provider and customer acceptance of NQS, it was decided to use  
 128191 NQS as the basis for the POSIX Batch Environment amendment in 1987. Other batch systems  
 128192 considered at the time included CTSS, MDQS (a forerunner of NQS from the Ballistics Research  
 128193 Laboratory), and PROD (a Los Alamos Labs development). None were thought to have both the  
 128194 functionality and acceptability of NQS.

## 128195 **NQS Differences from the *at* utility**

128196 The base standard *at* and *batch* utilities are not sufficient to meet the batch processing needs in a  
 128197 supercomputing environment and additional functionality in the areas of resource management,  
 128198 job scheduling, system management, and control of output is required.

## 128199 **Batch Environment Services and Utilities Option Definitions**

128200 The concept of a batch job is closely related to a session with a session leader. The main  
 128201 difference is that a batch job does not have a controlling terminal. There has been much debate  
 128202 over whether to use the term “request” or “job”. Job was the final choice because of the  
 128203 historical use of this term in the batch environment.

128204 The current definition for job identifiers is not sufficient with the model of destinations. The  
 128205 current definition is:

128206 `sequence_number.originating_host`

128207 Using the model of destination, a host may include multiple batch nodes, the location of which  
 128208 is identified uniquely by a name or directory service. If the current definition is used, batch  
 128209 nodes running on the same host would have to coordinate their use of sequence numbers, as  
 128210 sequence numbers are assigned by the originating host. The alternative is to use the originating

- 128211 batch node name instead of the originating host name.
- 128212 The reasons for wishing to run more than one batch system per host could be the following.
- 128213 A test and production batch system are maintained on a single host. This is most likely in a  
128214 development facility, but could also arise when a site is moving from one version to another. The  
128215 new batch system could be installed as a test version that is completely separate from the  
128216 production batch system, so that problems can be isolated to the test system. Requiring the batch  
128217 nodes to coordinate their use of sequence numbers creates a dependency between the two  
128218 nodes, and that defeats the purpose of running two nodes.
- 128219 A site has multiple departments using a single host, with different management policies. An  
128220 example of contention might be in job selection algorithms. One group might want a FIFO type  
128221 of selection, while another group wishes to use a more complex algorithm based on resource  
128222 availability. Again, requiring the batch nodes to coordinate is an unnecessary binding.
- 128223 The proposal eventually accepted was to replace originating host with originating batch node.  
128224 This supplies sufficient granularity to ensure unique job identifiers. If more than one batch node  
128225 is on a particular host, they each have their own unique name.
- 128226 The queue portion of a destination is not part of the job identifier as these are not required to be  
128227 unique between batch nodes. For instance, two batch nodes may both have queues called small,  
128228 medium, and large. It is only the batch node name that is uniquely identifiable throughout the  
128229 batch system. The queue name has no additional function in this context.
- 128230 Assume there are three batch nodes, each of which has its own name server. On batch node one,  
128231 there are no queues. On batch node two, there are fifty queues. On batch node three, there are  
128232 forty queues. The system administrator for batch node one does not have to configure queues,  
128233 because there are none implemented. However, if a user wishes to send a job to either batch  
128234 node two or three, the system administrator for batch node one must configure a destination  
128235 that maps to the appropriate batch node and queue. If every queue is to be made accessible from  
128236 batch node one, the system administrator has to configure ninety destinations.
- 128237 To avoid requiring this, there should be a mechanism to allow a user to separate the destination  
128238 into a batch node name and a queue name. Then, an implementation that is configured to get to  
128239 all the batch nodes does not need any more configuration to allow a user to get to all of the  
128240 queues on all of the batch nodes. The node name is used to locate the batch node, while the  
128241 queue name is sent unchanged to that batch node.
- 128242 The following are requirements that a destination identifier must be capable of providing:
- 128243 • The ability to direct a job to a queue in a particular batch node.
  - 128244 • The ability to direct a job to a particular batch node.
  - 128245 • The ability to group at a higher level than just one queue. This includes grouping similar  
128246 queues across multiple batch nodes (this is a pipe queue).
  - 128247 • The ability to group batch nodes. This allows a user to submit a job to a group name with  
128248 no knowledge of the batch node configuration. This also provides aliasing as a special  
128249 case. Aliasing is a group containing only one batch node name. The group name is the  
128250 alias.
- 128251 In addition, the administrator has the following requirements:
- 128252 • The ability to control access to the queues.
  - 128253 • The ability to control access to the batch nodes.



- The ability to control access to groups of queues (pipe queues).
- The ability to configure retry time intervals and durations.

The requirements of the user are met by destination as explained in the following.

The user has the ability to specify a queue name, which is known only to the batch node specified. There is no configuration of these queues required on the submitting node.

The user has the ability to specify a batch node whose name is network-unique. The configuration required is that the batch node be defined as an application, just as other applications such as FTP are configured.

Once a job reaches a queue, it can again become a user of the batch system. The batch node can choose to send the job to another batch node or queue or both. In other words, the routing is at an application level, and it is up to the batch system to choose where the job will be sent. Configuration is up to the batch node where the queue resides. This provides grouping of queues across batch nodes or within a batch node. The user submits the job to a queue, which by definition routes the job to other queues or nodes or both.

A node name may be given to a naming service, which returns multiple addresses as opposed to just one. This provides grouping at a batch node level. This is a local issue, meaning that the batch node must choose only one of these addresses. The list of addresses is not sent with the job, and once the job is accepted on another node, there is no connection between the list and the job. The requirements of the administrator are met by destination as explained in the following.

The control of queues is a batch system issue, and will be done using the batch administrative utilities.

The control of nodes is a network issue, and will be done through whatever network facilities are available.

The control of access to groups of queues (pipe queues) is covered by the control of any other queue. The fact that the job may then be sent to another destination is not relevant.

The propagation of a job across more than one point-to-point connection was dropped because of its complexity and because all of the issues arising from this capability could not be resolved. It could be provided as additional functionality at some time in the future.

The addition of *network* as a defined term was done to clarify the difference between a network of batch nodes as opposed to a network of hosts. A network of batch nodes is referred to as a batch system. The network refers to the actual host configuration. A single host may have multiple batch nodes.

In the absence of a standard network naming convention, this option establishes its own convention for the sake of consistency and expediency. This is subject to change, should a future working group develop a standard naming convention for network pathnames.

### **C.3.1 Batch General Concepts**

During the development of the Batch Environment Services and Utilities option, a number of topics were discussed at length which influenced the wording of the normative text but could not be included in the final text. The following items are some of the most significant terms and concepts of those discussed:

- Small and Consistent Command Set

Often, conventional utilities from UNIX systems have a very complicated utility syntax and usage. This can often result in confusion and errors when trying to use them. The

- 128297 Batch Environment Services and Utilities option utility set, on the other hand, has been  
128298 paired to a small set of robust utilities with an orthogonal calling sequence.
- 128299 • Checkpoint/Restart
- 128300 This feature permits an already executing process to checkpoint or save its contents. Some  
128301 implementations permit this at both the batch utility level (for example, checkpointing this  
128302 job upon its abnormal termination) or from within the job itself via a system call. Support  
128303 of checkpoint/restart is optional. A conscious, careful effort was made to make the *qsub*  
128304 utility consistently refer to checkpoint/restart as optional functionality.
- 128305 • Rerunability
- 128306 When a user submits a job for batch processing, they can designate it “rerunnable” in that  
128307 it will automatically resume execution from the start of the job if the machine on which it  
128308 was executing crashes for some reason. The decision on whether the job will be rerun or  
128309 not is entirely up to the submitter of the job and no decisions will be made within the batch  
128310 system. A job that is rerunnable and has been submitted with the proper  
128311 checkpoint/restart switch will first be checkpointed and execution begun from that point.  
128312 Furthermore, use of the implementation-defined checkpoint/restart feature will not be  
128313 defined in this context.
- 128314 • Error Codes
- 128315 All utilities exit with error status zero (0) if successful, one (1) if a user error occurred, and  
128316 two (2) for an internal Batch Environment Services and Utilities option error.
- 128317 • Level of Portability
- 128318 Portability is specified at both the user, operator, and administrator levels. A conforming  
128319 batch implementation prevents identical functionality and behavior at all these levels.  
128320 Additionally, portable batch shell scripts with embedded Batch Environment Services and  
128321 Utilities option utilities add an additional level of portability.
- 128322 • Resource Specification
- 128323 A small set of globally understood resources, such as memory and CPU time, is specified.  
128324 All conforming batch implementations are able to process them in a manner consistent  
128325 with the yet-to-be-developed resource management model. Resources not in this  
128326 amendment set are ignored and passed along as part of the argument stream of the utility.
- 128327 • Queue Position
- 128328 Queue position is the place a job occupies in a queue. It is dependent on a variety of factors  
128329 such as submission time and priority. Since priority may be affected by the implementation  
128330 of fair share scheduling, the definition of queue position is implementation-defined.
- 128331 • Queue ID
- 128332 A numerical queue ID is an external requirement for purposes of accounting. The  
128333 identification number was chosen over queue name for processing convenience.
- 128334 • Job ID
- 128335 A common notion of “jobs” is a collection of processes whose process group cannot be  
128336 altered and is used for resource management and accounting. This concept is  
128337 implementation-defined and, as such, has been omitted from the batch amendment.
- 128338 • Bytes *versus* Words
- 128339 Except for one case, bytes are used as the standard unit for memory size. Furthermore, the  
128340 definition of a word varies from machine to machine. Therefore, bytes will be the default



- 128341 unit of memory size.
- 128342 • Regular Expressions
- 128343 The standard definition of regular expressions is much too broad to be used in the batch  
128344 utility syntax. All that is needed is a simple concept of “all”; for example, delete all my jobs  
128345 from the named queue. For this reason, regular expressions have been eliminated from the  
128346 batch amendment.
- 128347 • Display Privacy
- 128348 How much data should be displayed locally through functions? Local policy dictates the  
128349 amount of privacy. Library functions must be used to create and enforce local policy.  
128350 Network and local *qstats* must reflect the policy of the server machine.
- 128351 • Remote Host Naming Convention
- 128352 It was decided that host names would be a maximum of 255 characters in length, with at  
128353 most 15 characters being shown in displays. The 255 character limit was chosen because it  
128354 is consistent with BSD. The 15-character limit was an arbitrary decision.
- 128355 • Network Administration
- 128356 Network administration is important, but is outside the scope of the batch amendment.  
128357 Network administration could be done with *rsh*. However, authentication becomes two-  
128358 sided.
- 128359 • Network Administration Philosophy
- 128360 Keep it simple. Centralized management should be possible. For example, Los Alamos  
128361 needs a dumb set of CPUs to be managed by a central system *versus* several  
128362 independently-managed systems as is the general case for the Batch Environment Services  
128363 and Utilities option.
- 128364 • Operator Utility Defaults (that is, Default Host, User, Account, and so on)
- 128365 It was decided that usability would override orthogonality and syntactic consistency.
- 128366 • The Batch System Manager and Operator Distinction
- 128367 The distinction between manager and operator is that operators can only control the flow  
128368 of jobs. A manager can alter the batch system configuration in addition to job flow. POSIX  
128369 makes a distinction between user and system administrator but goes no further. The  
128370 concepts of manager and operator privileges fall under local policy. The distinction  
128371 between manager and operator is historical in batch environments, and the Batch  
128372 Environment Services and Utilities option has continued that distinction.
- 128373 • The Batch System Administrator
- 128374 An administrator is equivalent to a batch system manager.

### 128375 C.3.2 Batch Services

- 128376 This rationale is provided as informative rather than normative text, to avoid placing  
128377 requirements on implementors regarding the use of symbolic constants, but at the same time to  
128378 give implementors a preferred practice for assigning values to these constants to promote  
128379 interoperability.
- 128380 The *Checkpoint* and *Minimum\_Cpu\_Interval* attributes induce a variety of behavior depending  
128381 upon their values. Some jobs cannot or should not be checkpointed. Other users will simply  
128382 need to ensure job continuation across planned downtimes; for example, scheduled preventive

128383 maintenance. For users consuming expensive resources, or for jobs that run longer than the  
 128384 mean time between failures, however, periodic checkpointing may be essential. However,  
 128385 system administrators must be able to set minimum checkpoint intervals on a queue-by-queue  
 128386 basis to guard against, for example, naive users specifying interval values too small on memory-  
 128387 intensive jobs. Otherwise, system overhead would adversely affect performance.

128388 The use of symbolic constants, such as `NO_CHECKPOINT`, was introduced to lend a degree of  
 128389 formalism and portability to this option.

128390 Support for checkpointing is optional for servers. However, clients must provide for the `-c`  
 128391 option, since in a distributed environment the job may run on a server that does provide such  
 128392 support, even if the host of the client does not support the checkpoint feature.

128393 If the user does not specify the `-c` option, the default action is left unspecified by this option.  
 128394 Some implementations may wish to do checkpointing by default; others may wish to checkpoint  
 128395 only under an explicit request from the user.

128396 The *Priority* attribute has been made non-optional. All clients already had been required to  
 128397 support the `-p` option. The concept of prioritization is common in historical implementations.  
 128398 The default priority is left to the server to establish.

128399 The *Hold\_Types* attribute has been modified to allow for implementation-defined hold types to  
 128400 be passed to a batch server.

128401 It was the intent of the IEEE P1003.15 working group to mandate the support for the  
 128402 *Resource\_List* attribute in this option by referring to another amendment, specifically the  
 128403 IEEE P1003.1a draft standard. However, during the development of the IEEE P1003.1a draft  
 128404 standard this was excluded. As such this requirement has been removed from the normative  
 128405 text.

128406 The *Shell\_Path* attribute has been modified to accept a list of shell paths that are associated with  
 128407 a host. The name of the attribute has been changed to *Shell\_Path\_List*.

### 128408 **C.3.3 Common Behavior for Batch Environment Utilities**

128409 This section was defined to meet the goal of a “Small and Consistent Command Set” for this  
 128410 option.

## 128411 **C.4 Utilities**

128412 For the utilities included in POSIX.1-2008, see the RATIONALE sections on the individual  
 128413 reference pages.

### 128414 **C.4.1 Utilities Removed in this Version**

128415 None.

128416 **C.4.2 Utilities Removed in the Previous Version**

128417 The following utilities were removed in the previous version of this standard:

128418	<i>calendar</i>	<i>cu</i>	<i>line</i>	<i>pcat</i>	<i>unpack</i>
128419	<i>cancel</i>	<i>dircmp</i>	<i>lint</i>	<i>pg</i>	<i>uulog</i>
128420	<i>cc</i>	<i>dis</i>	<i>lpstat</i>	<i>spell</i>	<i>uuname</i>
128421	<i>col</i>	<i>egrep</i>	<i>mail</i>	<i>sum</i>	<i>uupick</i>
128422	<i>cpio</i>	<i>fgrep</i>	<i>pack</i>	<i>tar</i>	<i>uuto</i>

128423 **C.4.3 Exclusion of Utilities**

128424 The set of utilities contained in POSIX.1-2008 is drawn from the base documents for IEEE Std  
 128425 1003.2-1992, with one addition: the *c99* utility. This section contains rationale for some of the  
 128426 deliberations that led to this set of utilities, and why certain utilities were excluded.

128427 Many utilities were evaluated by the standard developers; more historical utilities were  
 128428 excluded from the base documents for IEEE Std 1003.2-1992 than included. The following list  
 128429 contains many common UNIX system utilities that were not included as mandatory utilities, in  
 128430 the User Portability Utilities option, in the XSI option, or in one of the software development  
 128431 groups. It is logistically difficult for this rationale to distribute correctly the reasons for not  
 128432 including a utility among the various utility options. Therefore, this section covers the reasons  
 128433 for all utilities not included in POSIX.1-2008.

128434 This rationale is limited to a discussion of only those utilities actively or indirectly evaluated by  
 128435 the IEEE Std 1003.2-1992 standard developers, rather than the list of all known UNIX utilities  
 128436 from all its variants.

128437 *adb* The intent of the various software development utilities was to assist in the  
 128438 installation (rather than the actual development and debugging) of applications.  
 128439 This utility is primarily a debugging tool. Furthermore, many useful aspects of *adb*  
 128440 are very hardware-specific.

128441 *as* Assemblers are hardware-specific and are included implicitly as part of the  
 128442 compilers in POSIX.1-2008.

128443 *banner* The only known use of this command is as part of the *lp* printer header pages. It  
 128444 was decided that the format of the header is implementation-defined, so this utility  
 128445 is superfluous to application portability.

128446 *calendar* This reminder service program is not useful to conforming applications.

128447 *cancel* The *lp* (line printer spooling) system specified is the most basic possible and did  
 128448 not need this level of application control.

128449 *chroot* This is primarily of administrative use, requiring superuser privileges.

128450 *col* No utilities defined in POSIX.1-2008 produce output requiring such a filter. The  
 128451 *nroff* text formatter is present on many historical systems and will continue to  
 128452 remain as an extension; *col* is expected to be shipped by all the systems that ship  
 128453 *nroff*.

128454 *cpio* This has been replaced by *pax*, for reasons explained in the rationale for that utility.

128455 *cpp* This is subsumed by *c99*.

128456	<i>cu</i>	This utility is terminal-oriented and is not useful from shell scripts or typical application programs.
128457		
128458	<i>dc</i>	The functionality of this utility can be provided by the <i>bc</i> utility; <i>bc</i> was selected because it was easier to use and had superior functionality. Although the historical versions of <i>bc</i> are implemented using <i>dc</i> as a base, POSIX.1-2008 prescribes the interface and not the underlying mechanism used to implement it.
128459		
128460		
128461		
128462	<i>dircmp</i>	Although a useful concept, the historical output of this directory comparison program is not suitable for processing in application programs. Also, the <i>diff -r</i> command gives equivalent functionality.
128463		
128464		
128465	<i>dis</i>	Disassemblers are hardware-specific.
128466	<i>emacs</i>	The community of <i>emacs</i> editing enthusiasts was adamant that the full <i>emacs</i> editor not be included in IEEE Std 1003.2-1992 because they were concerned that an attempt to standardize this very powerful environment would encourage vendors to ship versions conforming strictly to the standard, but lacking the extensibility required by the community. The author of the original <i>emacs</i> program also expressed his desire to omit the program. Furthermore, there were a number of historical UNIX systems that did not include <i>emacs</i> , or included it without supporting it, but there were very few that did not include and support <i>vi</i> .
128467		
128468		
128469		
128470		
128471		
128472		
128473		
128474	<i>ld</i>	This is subsumed by <i>c99</i> .
128475	<i>line</i>	The functionality of <i>line</i> can be provided with <i>read</i> .
128476	<i>lint</i>	This technology is partially subsumed by <i>c99</i> . It is also hard to specify the degree of checking for possible error conditions in programs in any compiler, and specifying what <i>lint</i> would do in these cases is equally difficult.
128477		
128478		
128479		It is fairly easy to specify what a compiler does. It requires specifying the language, what it does with that language, and stating that the interpretation of any incorrect program is unspecified. Unfortunately, any description of <i>lint</i> is required to specify what to do with erroneous programs. Since the number of possible errors and questionable programming practices is infinite, one cannot require <i>lint</i> to detect all errors of any given class.
128480		
128481		
128482		
128483		
128484		
128485		Additionally, some vendors complained that since many compilers are distributed in a binary form without a <i>lint</i> facility (because the ISO C standard does not require one), implementing the standard as a stand-alone product will be much harder. Rather than being able to build upon a standard compiler component (simply by providing <i>c99</i> as an interface), source to that compiler would most likely need to be modified to provide the <i>lint</i> functionality. This was considered a major burden on system providers for a very small gain to developers (users).
128486		
128487		
128488		
128489		
128490		
128491		
128492	<i>login</i>	This utility is terminal-oriented and is not useful from shell scripts or typical application programs.
128493		
128494	<i>lorder</i>	This utility is an aid in creating an implementation-defined detail of object libraries that the standard developers did not feel required standardization.
128495		
128496	<i>lpstat</i>	The <i>lp</i> system specified is the most basic possible and did not need this level of application control.
128497		
128498	<i>mail</i>	This utility was omitted in favor of <i>mailx</i> because there was a considerable functionality overlap between the two.
128499		

128500	<i>mknod</i>	This was omitted in favor of <i>mkfifo</i> , as <i>mknod</i> has too many implementation-defined functions.
128501		
128502	<i>news</i>	This utility is terminal-oriented and is not useful from shell scripts or typical application programs.
128503		
128504	<i>pack</i>	This compression program was considered inferior to <i>compress</i> .
128505	<i>passwd</i>	This utility was proposed in an early draft of the IEEE Std 1003.2-1992 UPE but
128506		met with too many objections to be included. There were various reasons:
128507		<ul style="list-style-type: none"> <li>• Changing a password should not be viewed as a command, but as part of the login sequence. Changing a password should only be done while a trusted path is in effect.</li> </ul>
128508		
128509		
128510		<ul style="list-style-type: none"> <li>• Even though the text in early drafts was intended to allow a variety of implementations to conform, the security policy for one site may differ from another site running with identical hardware and software. One site might use password authentication while the other did not. Vendors could not supply a <i>passwd</i> utility that would conform to POSIX.1-2008 for all sites using their system.</li> </ul>
128511		
128512		
128513		
128514		
128515		
128516		<ul style="list-style-type: none"> <li>• This is really a subject for a system administration working group or a security working group.</li> </ul>
128517		
128518	<i>pcat</i>	This compression program was considered inferior to <i>zcat</i> .
128519	<i>pg</i>	This duplicated many of the features of the <i>more</i> pager, which was preferred by the standard developers.
128520		
128521	<i>prof</i>	The intent of the various software development utilities was to assist in the installation (rather than the actual development and debugging) of applications.
128522		This utility is primarily a debugging tool.
128523		
128524	RCS	RCS was originally considered as part of a version control utilities portion of the scope. However, this aspect was abandoned by the standard developers. SCCS is now included as an optional part of the XSI option.
128525		
128526		
128527	<i>red</i>	Restricted editor. This was not considered by the standard developers because it never provided the level of security restriction required.
128528		
128529	<i>rsh</i>	Restricted shell. This was not considered by the standard developers because it does not provide the level of security restriction that is implied by historical documentation.
128530		
128531		
128532	<i>sdb</i>	The intent of the various software development utilities was to assist in the installation (rather than the actual development and debugging) of applications.
128533		This utility is primarily a debugging tool. Furthermore, some useful aspects of <i>sdb</i> are very hardware-specific.
128534		
128535		
128536	<i>sdiff</i>	The “side-by-side <i>diff</i> ” utility from System V was omitted because it is used infrequently, and even less so by conforming applications. Despite being in System V, it is not in the SVID or XPG.
128537		
128538		
128539	<i>shar</i>	Any of the numerous “shell archivers” were excluded because they did not meet the requirement of existing practice.
128540		
128541	<i>shl</i>	This utility is terminal-oriented and is not useful from shell scripts or typical application programs. The job control aspects of the shell command language are generally more useful.
128542		
128543		

128544	<i>size</i>	The intent of the various software development utilities was to assist in the installation (rather than the actual development and debugging) of applications. This utility is primarily a debugging tool.
128545		
128546		
128547	<i>spell</i>	This utility is not useful from shell scripts or typical application programs. The <i>spell</i> utility was considered, but was omitted because there is no known technology that can be used to make it recognize general language for user-specified input without providing a complete dictionary along with the input file.
128548		
128549		
128550		
128551	<i>su</i>	This utility is not useful from shell scripts or typical application programs. (There was also sentiment to avoid security-related utilities.)
128552		
128553	<i>sum</i>	This utility was renamed <i>cksum</i> .
128554	<i>tar</i>	This has been replaced by <i>pax</i> , for reasons explained in the rationale for that utility.
128555	<i>unpack</i>	This compression program was considered inferior to <i>uncompress</i> .
128556	<i>wall</i>	This utility is terminal-oriented and is not useful in shell scripts or typical applications. It is generally used only by system administrators.
128557		

128558

# *Rationale (Informative)*

128559

## **Part D:**

128560

## **Portability Considerations**

128561

*The Open Group*

128562

*The Institute of Electrical and Electronics Engineers, Inc.*





## Portability Considerations (Informative)

This section contains information to satisfy various international requirements:

- [Section D.1](#) describes perceived user requirements.
- [Section D.2](#) (on page 3753) indicates how the facilities of POSIX.1-2008 satisfy those requirements.
- [Section D.3](#) (on page 3761) offers guidance to writers of profiles on how the configurable options, limits, and optional behavior of POSIX.1-2008 should be cited in profiles.

### D.1 User Requirements

This section describes the user requirements that were perceived by the standard developers. The primary source for these requirements was an analysis of historical practice in widespread use, as typified by the base documents for the ISO POSIX-1: 1996 standard.

POSIX.1-2008 addresses the needs of users requiring open systems solutions for source code portability of applications. It currently addresses users requiring open systems solutions for source-code portability of applications involving multi-programming and process management (creating processes, signaling, and so on); access to files and directories in a hierarchy of file systems (opening, reading, writing, deleting files, and so on); access to asynchronous communications ports and other special devices; access to information about other users of the system; facilities supporting applications requiring bounded (realtime) response.

The following users are identified for POSIX.1-2008:

- Those employing applications written in high-level languages, such as C, Ada, or FORTRAN.
- Users who desire conforming applications that do not necessarily require the characteristics of high-level languages (for example, the speed of execution of compiled languages or the relative security of source code intellectual property inherent in the compilation process).
- Users who desire conforming applications that can be developed quickly and can be modified readily without the use of compilers and other system components that may be unavailable on small systems or those without special application development capabilities.
- Users who interact with a system to achieve general-purpose time-sharing capabilities common to most business or government offices or academic environments: editing, filing, inter-user communications, printing, and so on.
- Users who develop applications for POSIX-conformant systems.
- Users who develop applications for UNIX systems.

An acknowledged restriction on applicable users is that they are limited to the group of individuals who are familiar with the style of interaction characteristic of historically-derived systems based on one of the UNIX operating systems (as opposed to other historical systems

128601 with different models, such as MS/DOS, Macintosh, VMS, MVS, and so on). Typical users  
 128602 would include program developers, engineers, or general-purpose time-sharing users.

128603 The requirements of users of POSIX.1-2008 can be summarized as a single goal: *application source*  
 128604 *portability*. The requirements of the user are stated in terms of the requirements of portability of  
 128605 applications. This in turn becomes a requirement for a standardized set of syntax and semantics  
 128606 for operations commonly found on many operating systems.

128607 The following sections list the perceived requirements for application portability.

#### 128608 **D.1.1 Configuration Interrogation**

128609 An application must be able to determine whether and how certain optional features are  
 128610 provided and to identify the system upon which it is running, so that it may appropriately adapt  
 128611 to its environment.

128612 Applications must have sufficient information to adapt to varying behaviors of the system.

#### 128613 **D.1.2 Process Management**

128614 An application must be able to manage itself, either as a single process or as multiple processes.  
 128615 Applications must be able to manage other processes when appropriate.

128616 Applications must be able to identify, control, create, and delete processes, and there must be  
 128617 communication of information between processes and to and from the system.

128618 Applications must be able to use multiple flows of control with a process (threads) and  
 128619 synchronize operations between these flows of control.

#### 128620 **D.1.3 Access to Data**

128621 Applications must be able to operate on the data stored on the system, access it, and transmit it  
 128622 to other applications. Information must have protection from unauthorized or accidental access  
 128623 or modification.

#### 128624 **D.1.4 Access to the Environment**

128625 Applications must be able to access the external environment to communicate their input and  
 128626 results.

#### 128627 **D.1.5 Access to Determinism and Performance Enhancements**

128628 Applications must have sufficient control of resource allocation to ensure the timeliness of  
 128629 interactions with external objects.

128630 **D.1.6 Operating System-Dependent Profile**

128631 The capabilities of the operating system may make certain optional characteristics of the base  
128632 language in effect no longer optional, and this should be specified.

128633 **D.1.7 I/O Interaction**

128634 The interaction between the C language I/O subsystem (*stdio*) and the I/O subsystem of  
128635 POSIX.1-2008 must be specified.

128636 **D.1.8 Internationalization Interaction**

128637 The effects of the environment of POSIX.1-2008 on the internationalization facilities of the C  
128638 language must be specified.

128639 **D.1.9 C-Language Extensions**

128640 Certain functions in the C language must be extended to support the additional capabilities  
128641 provided by POSIX.1-2008.

128642 **D.1.10 Command Language**

128643 Users should be able to define procedures that combine simple tools and/or applications into  
128644 higher-level components that perform to the specific needs of the user. The user should be able  
128645 to store, recall, use, and modify these procedures. These procedures should employ a powerful  
128646 command language that is used for recurring tasks in conforming applications (scripts) in the  
128647 same way that it is used interactively to accomplish one-time tasks. The language and the  
128648 utilities that it uses must be consistent between systems to reduce errors and retraining.

128649 **D.1.11 Interactive Facilities**

128650 Use the system to accomplish individual tasks at an interactive terminal. The interface should be  
128651 consistent, intuitive, and offer usability enhancements to increase the productivity of terminal  
128652 users, reduce errors, and minimize retraining costs. Online documentation or usage assistance  
128653 should be available.

128654 **D.1.12 Accomplish Multiple Tasks Simultaneously**

128655 Access applications and interactive facilities from a single terminal without requiring serial  
128656 execution: switch between multiple interactive tasks; schedule one-time or periodic background  
128657 work; display the status of all work in progress or scheduled; influence the priority scheduling  
128658 of work, when authorized.

**128659 D.1.13 Complex Data Manipulation**

128660 Manipulate data in files in complex ways: sort, merge, compare, translate, edit, format, pattern  
 128661 match, select subsets (strings, columns, fields, rows, and so on). These facilities should be  
 128662 available to both conforming applications and interactive users.

**128663 D.1.14 File Hierarchy Manipulation**

128664 Create, delete, move/rename, copy, backup/archive, and display files and directories. These  
 128665 facilities should be available to both conforming applications and interactive users.

**128666 D.1.15 Locale Configuration**

128667 Customize applications and interactive sessions for the cultural and language conventions of the  
 128668 user. Employ a wide variety of standard character encodings. These facilities should be available  
 128669 to both conforming applications and interactive users.

**128670 D.1.16 Inter-User Communication**

128671 Send messages or transfer files to other users on the same system or other systems on a network.  
 128672 These facilities should be available to both conforming applications and interactive users.

**128673 D.1.17 System Environment**

128674 Display information about the status of the system (activities of users and their interactive and  
 128675 background work, file system utilization, system time, configuration, and presence of optional  
 128676 facilities) and the environment of the user (terminal characteristics, and so on). Inform the  
 128677 system operator/administrator of problems. Control access to user files and other resources.

**128678 D.1.18 Printing**

128679 Output files on a variety of output device classes, accessing devices on local or network-  
 128680 connected systems. Control (or influence) the formatting, priority scheduling, and output  
 128681 distribution of work. These facilities should be available to both conforming applications and  
 128682 interactive users.

**128683 D.1.19 Software Development**

128684 Develop (create and manage source files, compile/interpret, debug) portable open systems  
 128685 applications and package them for distribution to, and updating of, other systems.

## D.2 Portability Capabilities

This section describes the significant portability capabilities of POSIX.1-2008 and indicates how the user requirements listed in [Section D.1](#) (on page 3749) are addressed. The capabilities are listed in the same format as the preceding user requirements; they are summarized below:

- Configuration Interrogation
- Process Management
- Access to Data
- Access to the Environment
- Access to Determinism and Performance Enhancements
- Operating System-Dependent Profile
- I/O Interaction
- Internationalization Interaction
- C-Language Extensions
- Command Language
- Interactive Facilities
- Accomplish Multiple Tasks Simultaneously
- Complex Data Manipulation
- File Hierarchy Manipulation
- Locale Configuration
- Inter-User Communication
- System Environment
- Printing
- Software Development

### D.2.1 Configuration Interrogation

The *uname()* operation provides basic identification of the system. The *sysconf()*, *pathconf()*, and *fpathconf()* functions and the *getconf* utility provide means to interrogate the implementation to determine how to adapt to the environment in which it is running. These values can be either static (indicating that all instances of the implementation have the same value) or dynamic (indicating that different instances of the implementation have the different values, or that the value may vary for other reasons, such as reconfiguration).

#### Unsatisfied Requirements

None directly. However, as new areas are added, there will be a need for additional capability in this area.

## 128719 D.2.2 Process Management

128720 The *fork()*, *exec* family, *posix\_spawn()*, and *posix\_spawnp()* functions provide for the creation of  
 128721 new processes or the insertion of new applications into existing processes. The *\_Exit()*, *\_exit()*,  
 128722 *exit()*, and *abort()* functions allow for the termination of a process by itself. The *wait()*, *waitid()*,  
 128723 and *waitpid()* functions allow one process to deal with the termination of another.

128724 The *times()* function allows for basic measurement of times used by a process. Various  
 128725 functions, including *fstat()*, *getegid()*, *geteuid()*, *getgid()*, *getgrgid()*, *getgrnam()*, *getlogin()*,  
 128726 *getpid()*, *getppid()*, *getpwnam()*, *getpwuid()*, *getuid()*, *lstat()*, and *stat()*, provide for access to the  
 128727 identifiers of processes and the identifiers and names of owners of processes (and files).

128728 The various functions operating on environment variables provide for communication of  
 128729 information (primarily user-configurable defaults) from a parent to child processes.

128730 The operations on the current working directory control and interrogate the directory from  
 128731 which relative pathname searches start. The *umask()* function controls the default protections  
 128732 applied to files created by the process.

128733 The *alarm()*, *pause()*, *sleep()*, *ualarm()*, and *usleep()* operations allow the process to suspend until  
 128734 a timer has expired or to be notified when a period of time has elapsed. The *time()* operation  
 128735 interrogates the current time and date.

128736 The signal mechanism provides for communication of events either from other processes or  
 128737 from the environment to the application, and the means for the application to control the effect  
 128738 of these events. The mechanism provides for external termination of a process and for a process  
 128739 to suspend until an event occurs. The mechanism also provides for a value to be associated with  
 128740 an event.

128741 Job control provides a means to group processes and control them as groups, and to control their  
 128742 access to the function between the user and the system (the “controlling terminal”). It also  
 128743 provides the means to suspend and resume processes.

128744 The Process Scheduling option provides control of the scheduling and priority of a process.

128745 The Message Passing option provides a means for interprocess communication involving small  
 128746 amounts of data.

128747 The Memory Management facilities provide control of memory resources and for the sharing of  
 128748 memory. This functionality is mandatory on POSIX-conforming systems.

128749 The Threads facilities provide multiple flows of control with a process (threads),  
 128750 synchronization between threads (including mutexes, barriers, and spin locks), association of  
 128751 data with threads, and controlled cancellation of threads.

128752 The XSI interprocess communications functionality provide an alternate set of facilities to  
 128753 manipulate semaphores, message queues, and shared memory. These are provided on XSI-  
 128754 conformant systems to support conforming applications developed to run on UNIX systems.

## 128755 D.2.3 Access to Data

128756 The *open()*, *close()*, *fclose()*, *fopen()*, and *pipe()* functions provide for access to files and data.  
 128757 Such files may be regular files, interprocess data channels (pipes), or devices. Additional types  
 128758 of objects in the file system are permitted and are being contemplated for standardization.

128759 The *access()*, *chmod()*, *chown()*, *dup()*, *dup2()*, *fchmod()*, *fcntl()*, *fstat()*, *ftruncate()*, *lstat()*,  
 128760 *readlink()*, *realpath()*, *stat()*, and *utime()* functions allow for control and interrogation of file and  
 128761 file-related objects (including symbolic links), and their ownership, protections, and timestamps.

128762 The *fgetc()*, *fputc()*, *fread()*, *fseek()*, *fsetpos()*, *fwrite()*, *getc()*, *getchar()*, *lseek()*, *putchar()*, *putc()*,  
 128763 *read()*, and *write()* functions provide for data transfer from the application to files (in all their  
 128764 forms).

128765 The *closedir()*, *link()*, *mkdir()*, *opendir()*, *readdir()*, *rename()*, *rmdir()*, *rewinddir()*, and *unlink()*  
 128766 functions provide for a complete set of operations on directories. Directories can arbitrarily  
 128767 contain other directories, and a single file can be mentioned in more than one directory.

128768 The *faccessat()*, *openat()*, *fchmodat()*, *fchownat()*, *fstatat()*, *linkat()*, *renameat()*, *readlinkat()*,  
 128769 *symlinkat()*, and *unlinkat()* functions allow for race-free and thread-safe file access. The  
 128770 motivation for the introduction of these functions was as follows:

- 128771 • Interfaces taking a pathname may be limited by the maximum length of a pathname  
 128772 ({PATH\_MAX}). The absolute path of files can far exceed this length. The alternative  
 128773 solution of changing the working directory and using relative pathnames is not thread-  
 128774 safe.
- 128775 • A second motivation is that files accessed outside the current working directory are subject  
 128776 to attacks caused by the race condition created by changing any of the elements of the  
 128777 pathnames used.
- 128778 • A third motivation is to allow application code which makes use of a virtual current  
 128779 working directory for each individual thread. In the alternative model there is only one  
 128780 current working directory for all threads.

128781 The file-locking mechanism provides for advisory locking (protection during transactions) of  
 128782 ranges of bytes (in effect, records) in a file.

128783 The *confstr()*, *fpathconf()*, *pathconf()*, and *sysconf()* functions provide for enquiry as to the  
 128784 behavior of the system where variability is permitted.

128785 The asynchronous input and output functions *aio\_cancel()*, *aio\_error()*, *aio\_fsync()*, *aio\_read()*,  
 128786 *aio\_return()*, *aio\_suspend()*, *aio\_write()*, and *lio\_listio()* provide for initiation and control of  
 128787 asynchronous data transfers.

128788 The Synchronized Input and Output option provides for assured commitment of data to media.

#### 128789 D.2.4 Access to the Environment

128790 The operations and types in XBD are provided for access to asynchronous serial devices. The  
 128791 primary intended use for these is the controlling terminal for the application (the interaction  
 128792 point between the user and the system). They are general enough to be used to control any  
 128793 asynchronous serial device. The functions are also general enough to be used with many other  
 128794 device types as a user interface when some emulation is provided.

128795 Less detailed access is provided for other device types, but in many instances an application  
 128796 need not know whether an object in the file system is a device or a regular file to operate  
 128797 correctly.



128798 **Unsatisfied Requirements**

128799 Detailed control of common device classes, specifically magnetic tape, is not provided.

128800 **D.2.5 Bounded (Realtime) Response**128801 The realtime signal functions *sigqueue()*, *sigtimedwait()*, and *sigwaitinfo()* provide queued signals  
128802 and the prioritization of the handling of signals.128803 The SCHED\_FIFO, SCHED\_SPORADIC, and SCHED\_RR scheduling policies provide control  
128804 over processor allocation.128805 The semaphore functions *sem\_close()*, *sem\_destroy()*, *sem\_getvalue()*, *sem\_init()*, *sem\_open()*,  
128806 *sem\_post()*, *sem\_timedwait()*, *sem\_trywait()*, *sem\_unlink()*, and *sem\_wait()* provide high-  
128807 performance synchronization.128808 The memory management functions provide memory locking for control of memory allocation,  
128809 file mapping for high performance, and shared memory for high-performance interprocess  
128810 communication. The Message Passing option provides for interprocess communication without  
128811 being dependent on shared memory.128812 The timers functions *clock\_getres()*, *clock\_gettime()*, *clock\_settime()*, *nanosleep()*, *timer\_create()*,  
128813 *timer\_delete()*, *timer\_getoverrun()*, *timer\_gettime()*, and *timer\_settime()* provide functionality to  
128814 manipulate clocks and timers and include a high resolution function called *nanosleep()* with a  
128815 finer resolution than the *sleep()* function.128816 The timeout functions — *pthread\_mutex\_timedlock()*, *pthread\_rwlock\_timedrdlock()*,  
128817 *pthread\_rwlock\_timedwrlock()*, and *sem\_timedwait()* — the Typed Memory Objects option and the  
128818 Monotonic Clock option provide further facilities for applications to use to obtain predictable  
128819 bounded response.128820 **D.2.6 Operating System-Dependent Profile**128821 POSIX.1-2008 makes no distinction between text and binary files. The values of EXIT\_SUCCESS  
128822 and EXIT\_FAILURE are further defined.128823 **Unsatisfied Requirements**128824 None known, but the ISO C standard may contain some additional options that could be  
128825 specified.128826 **D.2.7 I/O Interaction**128827 POSIX.1-2008 defines how each of the ISO C standard *stdio* functions interact with the POSIX.1  
128828 operations, typically specifying the behavior in terms of POSIX.1 operations.



128829 **Unsatisfied Requirements**

128830 None.

128831 **D.2.8 Internationalization Interaction**

128832 The POSIX.1-2008 environment operations provide a means to define the environment for  
 128833 *setlocale()* and time functions such as *ctime()*. The *tzset()* function is provided to set time  
 128834 conversion information.

128835 The *nl\_langinfo()* function is provided to query locale-specific cultural settings.

128836 The multiple concurrent locale functions *duplocale()*, *freelocale()*, *is\*\_l()*, *newlocale()*,  
 128837 *strcasemp\_l()*, *strcoll\_l()*, *strfmon\_l()*, *strncasemp\_l()*, *strxfrm\_l()*, *tolower\_l()*, *toupper\_l()*,  
 128838 *towctrans\_l()*, *towlower\_l()*, *towupper\_l()*, *uselocale\_l()*, *wscasemp\_l()*, *wscoll\_l()*, *wscncasemp\_l()*,  
 128839 *wcsxfrm\_l()*, *wctrans\_l()*, and *wctype\_l()* are provide to support per-thread locale information.

128840 **Unsatisfied Requirements**

128841 None.

128842 **D.2.9 C-Language Extensions**

128843 The *setjmp()* and *longjmp()* functions are not defined to be cognizant of the signal masks defined  
 128844 for POSIX.1. The *sigsetjmp()* and *siglongjmp()* functions are provided to fill this gap.

128845 The *\_setjmp()* and *\_longjmp()* functions are provided as XSI options to support historic practice.

128846 **Unsatisfied Requirements**

128847 None.

128848 **D.2.10 Command Language**

128849 The shell command language, as described in XCU [Chapter 2](#) (on page 2335), is a common  
 128850 language useful in batch scripts, through an API to high-level languages (for the C-Language  
 128851 Binding option, *system()* and *popen()*) and through an interactive terminal (see the *sh* utility).  
 128852 The shell language has many of the characteristics of a high-level language, but it has been  
 128853 designed to be more suitable for user terminal entry and includes interactive debugging  
 128854 facilities. Through the use of pipelining, many complex commands can be constructed from  
 128855 combinations of data filters and other common components. Shell scripts can be created, stored,  
 128856 recalled, and modified by the user with simple editors.

128857 In addition to the basic shell language, the following utilities offer features that simplify and  
 128858 enhance programmatic access to the utilities and provide features normally found only in high-  
 128859 level languages: *basename*, *bc*, *command*, *dirname*, *echo*, *env*, *expr*, *false*, *printf*, *read*, *sleep*, *tee*, *test*,  
 128860 *time\**,<sup>9</sup> *true*, *wait*, *xargs*, and all of the special built-in utilities in XCU [Section 2.14](#) (on page 2374).

---

128861 9. The utilities listed with an asterisk here and later in this section are present only on systems which support the User Portability Utilities  
 128862 option. There may be further restrictions on the utilities offered with various configuration option combinations; see the individual utility  
 128863 descriptions.

128864      **Unsatisfied Requirements**

128865      None.

128866      **D.2.11 Interactive Facilities**

128867      The utilities offer a common style of command-line interface through conformance to the Utility  
128868      Syntax Guidelines (see XBD [Section 12.2](#), on page 216) and the common utility defaults (see XCU  
128869      [Section 1.4](#), on page 2326). The *sh* utility offers an interactive command-line history and editing  
128870      facility.

128871      The following utilities can be used interactively as well as by scripts; *alias*, *fc*, *mailx*, *unalias*, and  
128872      *write*.

128873      The following utilities in the User Portability Utilities option provide for interactive use: *ex*, *more*,  
128874      and *vi*; the *man* utility offers online access to system documentation.

128875      **Unsatisfied Requirements**

128876      The command line interface to individual utilities is as intuitive and consistent as historical  
128877      practice allows. Work underway based on graphical user interfaces may be more suitable for  
128878      novice or occasional users of the system.

128879      **D.2.12 Accomplish Multiple Tasks Simultaneously**

128880      The shell command language offers background processing through the asynchronous list  
128881      command form; see XCU [Section 2.9](#) (on page 2355).

128882      The *nohup* utility makes background processing more robust and usable.

128883      The *kill* utility can terminate background jobs.

128884      The following utilities support periodic job scheduling, control, and display: *at*, *batch*, *crontab*,  
128885      *nice*, *ps*, and *renice*.

128886      When the User Portability Utilities option is supported, the following utilities allow  
128887      manipulation of jobs: *bg*, *fg*, and *jobs*.

128888      **Unsatisfied Requirements**

128889      Terminals with multiple windows may be more suitable for some multi-tasking interactive uses  
128890      than the job control approach in POSIX.1-2008. See the comments on graphical user interfaces in  
128891      [Section D.2.11](#). The *nice* and *renice* utilities do not necessarily take advantage of complex system  
128892      scheduling algorithms that are supported by the realtime options within POSIX.1-2008.

128893      **D.2.13 Complex Data Manipulation**

128894      The following utilities address user requirements in this area: *asa*, *awk*, *bc*, *cmp*, *comm*, *csplit*, *cut*,  
128895      *dd*, *diff*, *ed*, *ex\**, *expand*, *expr*, *find*, *fold*, *grep*, *head*, *join*, *od*, *paste*, *pr*, *printf*, *sed*, *sort*, *split*, *tabs*, *tail*, *tr*,  
128896      *unexpand*, *uniq*, *uudecode*, *uuencode*, and *wc*.

128897 **Unsatisfied Requirements**

128898 Sophisticated text formatting utilities, such as *troff* or *TeX*, are not included. Standards work in  
 128899 the area of SGML may satisfy this.

128900 **D.2.14 File Hierarchy Manipulation**

128901 The following utilities address user requirements in this area: *basename*, *cd*, *chgrp*, *chmod*, *chown*,  
 128902 *cksum*, *cp*, *dd*, *df*, *diff*, *dirname*, *du*, *find*, *ls*, *ln*, *mkdir*, *mkfifo*, *mv*, *patch*, *pathchk*, *pax*, *pwd*, *rm*, *rmdir*,  
 128903 *test*, and *touch*.

128904 **Unsatisfied Requirements**

128905 Some graphical user interfaces offer more intuitive file manager components that allow file  
 128906 manipulation through the use of icons for novice users.

128907 **D.2.15 Locale Configuration**

128908 The standard utilities are affected by the various *LC\_* variables to achieve locale-dependent  
 128909 operation: character classification, collation sequences, regular expressions and shell pattern  
 128910 matching, date and time formats, numeric formatting, and monetary formatting. When the  
 128911 POSIX2\_LOCALEDEF option is supported, applications can provide their own locale definition  
 128912 files.

128913 The following utilities address user requirements in this area: *date*, *ed*, *ex\**, *find*, *grep*, *locale*,  
 128914 *localedef*, *more\**, *sed*, *sh*, *sort*, *tr*, *uniq*, and *vi\**.

128915 The *iconv()*, *iconv\_close()*, and *iconv\_open()* functions are available to allow an application to  
 128916 convert character data between supported character sets.

128917 The *gencat* utility and the *catopen()*, *catclose()*, and *catgets()* functions provide for message  
 128918 catalog manipulation.

128919 **Unsatisfied Requirements**

128920 Some aspects of multi-byte character and state-encoded character encodings have not yet been  
 128921 addressed. The C-language functions, such as *getopt()*, are generally limited to single-byte  
 128922 characters. The effect of the *LC\_MESSAGES* variable on message formats is only suggested at  
 128923 this time.

128924 **D.2.16 Inter-User Communication**

128925 The following utilities address user requirements in this area: *cksum*, *mailx*, *mesg*, *patch*, *pax*, *talk*,  
 128926 *uudecode*, *uuencode*, *who*, and *write*.

128927 The historical UUCP utilities are included as a separate UUCP Utilities option.

128928 **Unsatisfied Requirements**

128929 None.

128930 **D.2.17 System Environment**128931 The following utilities address user requirements in this area: *chgrp*, *chmod*, *chown*, *df*, *du*, *env*,  
128932 *getconf*, *id*, *logger*, *logname*, *mesg*, *newgrp*, *ps*, *stty*, *tput*, *tty*, *umask*, *uname*, and *who*.128933 The *closelog()*, *openlog()*, *setlogmask()*, and *syslog()* functions provide system logging facilities on  
128934 XSI-conformant systems; these are analogous to the *logger* utility.128935 **Unsatisfied Requirements**

128936 None.

128937 **D.2.18 Printing**128938 The following utilities address user requirements in this area: *pr* and *lp*.128939 **Unsatisfied Requirements**

128940 There are no features to control the formatting or scheduling of the print jobs.

128941 **D.2.19 Software Development**128942 The following utilities address user requirements in this area: *ar*, *asa*, *awk*, *c99*, *ctags*, *fort77*,  
128943 *getconf*, *getopts*, *lex*, *localedef*, *make*, *nm*, *od*, *patch*, *pax*, *strings*, *strip*, *time*, and *yacc*.128944 The *system()*, *popen()*, *pclose()*, *regcomp()*, *regex()*, *regerror()*, *regfree()*, *fnmatch()*, *getopt()*,  
128945 *glob()*, *globfree()*, *wordexp()*, and *wordfree()* functions allow C-language programmers to access  
128946 some of the interfaces used by the utilities, such as argument processing, regular expressions,  
128947 and pattern matching.128948 The SCCS source-code control system utilities are available on systems supporting the XSI  
128949 Development option.128950 **Unsatisfied Requirements**128951 There are no language-specific development tools related to languages other than C and  
128952 FORTRAN. The C tools are more complete and varied than the FORTRAN tools. There is no  
128953 data dictionary or other CASE-like development tools.128954 **D.2.20 Future Growth**128955 It is arguable whether or not all functionality to support applications is potentially within the  
128956 scope of POSIX.1-2008. As a simple matter of practicality, it cannot be. Areas such as graphics,  
128957 application domain-specific functionality, windowing, and so on, should be in unique standards.  
128958 As such, they are properly “Unsatisfied Requirements” in terms of providing fully conforming  
128959 applications, but ones which are outside the scope of POSIX.1-2008.128960 However, as the standards evolve, certain functionality once considered “exotic” enough to be  
128961 part of a separate standard become common enough to be included in a core standard such as  
128962 this. Realtime and networking, for example, have both moved from separate standards (with

128963 much difficult cross-referencing) into this standard over time, and although no specific areas  
128964 have been identified for inclusion in a future version, such inclusions seem likely.

## 128965 **D.3 Profiling Considerations**

128966 This section offers guidance to writers of profiles on how the configurable options, limits, and  
128967 optional behavior of POSIX.1-2008 should be cited in profiles. Profile writers should consult the  
128968 general guidance in POSIX.0 when writing POSIX Standardized Profiles.

128969 The information in this section is an inclusive list of features that should be considered by profile  
128970 writers. Subsetting of POSIX.1-2008 should follow XBD [Section 2.1.5.1](#) (on page 21). A set of  
128971 profiling options is described in [Appendix E](#) (on page 3775).

### 128972 **D.3.1 Configuration Options**

128973 There are two set of options suggested by POSIX.1-2008: those for POSIX-conforming systems  
128974 and those for X/Open System Interface (XSI) conformance. The requirements for XSI  
128975 conformance are documented in the Base Definitions volume of POSIX.1-2008 and not discussed  
128976 further here, as they superset the POSIX conformance requirements.

### 128977 **D.3.2 Configuration Options (Shell and Utilities)**

128978 There are three broad optional configurations for the Shell and Utilities volume of POSIX.1-2008:  
128979 basic execution system, development system, and user portability interactive system. The  
128980 options to support these, and other minor configuration options, are listed in XBD [Chapter 2](#) (on  
128981 page 15). Profile writers should consult the following list and the comments concerning user  
128982 requirements addressed by various components in [Section D.2](#) (on page 3753).

128983 POSIX2\_UPE

128984 The system supports the User Portability Utilities option.

128985 This option is a requirement for a user portability interactive system. It is required  
128986 frequently except for those systems, such as embedded realtime or dedicated application  
128987 systems, that support little or no interactive time-sharing work by users or operators. XSI-  
128988 conformant systems support this option.

128989 POSIX2\_SW\_DEV

128990 The system supports the Software Development Utilities option.

128991 This option is required by many systems, even those in which actual software development  
128992 does not occur. The *make* utility, in particular, is required by many application software  
128993 packages as they are installed onto the system. If POSIX2\_C\_DEV is supported,  
128994 POSIX2\_SW\_DEV is almost a mandatory requirement because of *ar* and *make*.

128995 POSIX2\_C\_BIND

128996 The system supports the C-Language Bindings option.

128997 This option is required on some implementations developing complex C applications or on  
128998 any system installing C applications in source form that require the functions in this option.  
128999 The *system()* and *popen()* functions, in particular, are widely used by applications; the  
129000 others are rather more specialized.

- 129001 POSIX2\_C\_DEV  
 129002 The system supports the C-Language Development Utilities option.
- 129003 This option is required by many systems, even those in which actual C-language software  
 129004 development does not occur. The *c99* utility, in particular, is required by many application  
 129005 software packages as they are installed onto the system. The *lex* and *yacc* utilities are used  
 129006 less frequently.
- 129007 POSIX2\_FORT\_DEV  
 129008 The system supports the FORTRAN Development Utilities option
- 129009 As with C, this option is needed on any system developing or installing FORTRAN  
 129010 applications in source form.
- 129011 POSIX2\_FORT\_RUN  
 129012 The system supports the FORTRAN Runtime Utilities option.
- 129013 This option is required for some FORTRAN applications that need the *asa* utility to convert  
 129014 Hollerith printing statement output. It is unknown how frequently this occurs.
- 129015 POSIX2\_LOCALEDEF  
 129016 The system supports the creation of locales.
- 129017 This option is needed if applications require their own customized locale definitions to  
 129018 operate. It is presently unknown whether many applications are dependent on this.  
 129019 However, the option is virtually mandatory for systems in which internationalized  
 129020 applications are developed.
- 129021 XSI-conformant systems support this option.
- 129022 POSIX2\_PBS  
 129023 The system supports the Batch Environment Services and Utilities option.
- 129024 POSIX2\_PBS\_ACCOUNTING  
 129025 The system supports the optional feature of accounting within the Batch Environment  
 129026 Services and Utilities option. It will be required in servers that implement the optional  
 129027 feature of accounting.
- 129028 POSIX2\_PBS\_CHECKPOINT  
 129029 The system supports the optional feature of checkpoint/restart within the Batch  
 129030 Environment Services and Utilities option.
- 129031 POSIX2\_PBS\_LOCATE  
 129032 The system supports the optional feature of locating batch jobs within the Batch  
 129033 Environment Services and Utilities option.
- 129034 POSIX2\_PBS\_MESSAGE  
 129035 The system supports the optional feature of sending messages to batch jobs within the Batch  
 129036 Environment Services and Utilities option.
- 129037 POSIX2\_PBS\_TRACK  
 129038 The system supports the optional feature of tracking batch jobs within the Batch  
 129039 Environment Services and Utilities option.
- 129040 POSIX2\_CHAR\_TERM  
 129041 The system supports at least one terminal type capable of all operations described in  
 129042 POSIX.1-2008.
- 129043 On systems with POSIX2\_UPE, this option is almost always required. It was developed  
 129044 solely to allow certain specialized vendors and user applications to bypass the requirement  
 129045 for general-purpose asynchronous terminal support. For example, an application and



129046 system that was suitable for block-mode terminals, such as IBM 3270s, would not need this  
 129047 option.  
 129048 XSI-conformant systems support this option.

### 129049 D.3.3 Configurable Limits

129050 Very few of the limits need to be increased for profiles. No profile can cite lower values.  
 129051 {POSIX2\_BC\_BASE\_MAX}  
 129052 {POSIX2\_BC\_DIM\_MAX}  
 129053 {POSIX2\_BC\_SCALE\_MAX}  
 129054 {POSIX2\_BC\_STRING\_MAX}  
 129055 No increase is anticipated for any of these *bc* values, except for very specialized applications  
 129056 involving huge numbers.  
 129057 {POSIX2\_COLL\_WEIGHTS\_MAX}  
 129058 Some natural languages with complex collation requirements require an increase from the  
 129059 default 2 to 4; no higher numbers are anticipated.  
 129060 {POSIX2\_EXPR\_NEST\_MAX}  
 129061 No increase is anticipated.  
 129062 {POSIX2\_LINE\_MAX}  
 129063 This number is much larger than most historical applications have been able to use. At some  
 129064 future time, applications may be rewritten to take advantage of even larger values.  
 129065 {POSIX2\_RE\_DUP\_MAX}  
 129066 No increase is anticipated.  
 129067 {POSIX2\_VERSION}  
 129068 This is actually not a limit, but a standard version stamp. Generally, a profile should specify  
 129069 XCU [Chapter 2](#) (on page 2335) by name in the normative references section, not this value.

### 129070 D.3.4 Configuration Options (System Interfaces)

129071 {NGROUPS\_MAX}  
 129072 A non-zero value indicates that the implementation supports supplementary groups.  
 129073 This option is needed where there is a large amount of shared use of files, but where a  
 129074 certain amount of protection is needed. Many profiles<sup>10</sup> are known to require this option; it  
 129075 should only be required if needed, but it should never be prohibited.  
 129076 \_POSIX\_ADVISORY\_INFO  
 129077 The system provides advisory information for file management.  
 129078 This option allows the application to specify advisory information that can be used to  
 129079 achieve better or even deterministic response time in file manager or input and output  
 129080 operations.  
 129081 \_POSIX\_ASYNCHRONOUS\_IO  
 129082 Support for asynchronous input and output is mandatory in POSIX.1-2008.

---

129083 10. There are no formally approved profiles of POSIX.1-2008 at the time of publication; the reference here is to various profiles generated by  
 129084 private bodies or governments.

## 129085        \_posix\_barriers

129086        Support for barrier synchronization is mandatory in POSIX.1-2008.

129087        This facility allows efficient synchronization of multiple parallel threads in multi-processor  
129088        systems in which the operation is supported in part by the hardware architecture.

## 129089        \_posix\_chown\_restricted

129090        The system restricts the right to “give away” files to other users. It is mandatory that an  
129091        implementation be able to support this facility in POSIX.1-2008; however, it is recognized  
129092        that implementations need not enable the functionality by default.129093        Some applications expect that they can change the ownership of files in this way. It is  
129094        provided where either security or system account requirements cause this ability to be a  
129095        problem. It is also known to be specified in many profiles.

## 129096        \_posix\_clock\_selection

129097        Support for clock selection is mandatory in POSIX.1-2008.

129098        This facility allows applications to request a high resolution sleep in order to suspend a  
129099        thread during a relative time interval, or until an absolute time value, using the desired  
129100        clock. It also allows the application to select the clock used in a *pthread\_cond\_timedwait()*  
129101        function call.

## 129102        \_posix\_cputime

129103        The system supports the Process CPU-Time Clocks option.

129104        This option allows applications to use a new clock that measures the execution times of  
129105        processes or threads, and the possibility to create timers based upon these clocks, for  
129106        runtime detection (and treatment) of execution time overruns.

## 129107        \_posix\_fsync

129108        The system supports file synchronization requests.

129109        This option was created to support historical systems that did not provide the feature.  
129110        Applications that are expecting guaranteed completion of their input and output operations  
129111        should require the *\_POSIX\_SYNC\_IO* option. This option should never be prohibited.

129112        XSI-conformant systems support this option.

## 129113        \_posix\_ipv6

129114        The system supports facilities related to Internet Protocol Version 6 (IPv6).

129115        This option was created to allow systems to transition to IPv6.

## 129116        \_posix\_job\_control

129117        Support for job control is mandatory in POSIX.1-2008.

129118        Most applications that use it can run when it is not present, although with a degraded level  
129119        of user convenience.

## 129120        \_posix\_mapped\_files

129121        Support for memory mapped files is mandatory in POSIX.1-2008.

129122        This facility provides for the mapping of regular files into the process address space.

129123        Both this facility and the Shared Memory Objects option provide shared access to memory  
129124        objects in the process address space. The *mmap()* and *munmap()* functions provide the  
129125        functionality of existing practice for mapping regular files. This functionality was deemed  
129126        unnecessary, if not inappropriate, for embedded systems applications and is expected to be  
129127        optional in subprofiles.



129128 `_POSIX_MEMLOCK`

129129 The system supports the locking of the address space.

129130 This option was created to support historical systems that did not provide the feature. It  
129131 should only be required if needed, but it should never be prohibited.129132 `_POSIX_MEMLOCK_RANGE`

129133 The system supports the locking of specific ranges of the address space.

129134 For applications that have well-defined sections that need to be locked and others that do  
129135 not, POSIX.1-2008 supports an optional set of functions to lock or unlock a range of process  
129136 addresses. The following are two reasons for having a means to lock down a specific range:

- 129137 1. An asynchronous event handler function that must respond to external events in a  
129138 deterministic manner such that page faults cannot be tolerated
- 129139 2. An input/output “buffer” area that is the target for direct-to-process I/O, and the  
129140 overhead of implicit locking and unlocking for each I/O call cannot be tolerated

129141 It should only be required if needed, but it should never be prohibited.

129142 `_POSIX_MEMORY_PROTECTION`

129143 Support for memory protection is mandatory in POSIX.1-2008.

129144 The provision of this facility typically imposes additional hardware requirements.

129145 `_POSIX_PRIORITIZED_IO`

129146 The system provides prioritization for input and output operations.

129147 The use of this option may interfere with the ability of the system to optimize input and  
129148 output throughput. It should only be required if needed, but it should never be prohibited.129149 `_POSIX_MESSAGE_PASSING`

129150 The system supports the passing of messages between processes.

129151 This option was created to support historical systems that did not provide the feature. The  
129152 functionality adds a high-performance XSI interprocess communication facility for local  
129153 communication. It should only be required if needed, but it should never be prohibited.129154 `_POSIX_MONOTONIC_CLOCK`

129155 The system supports the Monotonic Clock option.

129156 This option allows realtime applications to rely on a monotonically increasing clock that  
129157 does not jump backwards, and whose value does not change except for the regular ticking  
129158 of the clock.129159 `_POSIX_PRIORITY_SCHEDULING`

129160 The system provides priority-based process scheduling.

129161 Support of this option provides predictable scheduling behavior, allowing applications to  
129162 determine the order in which processes that are ready to run are granted access to a  
129163 processor. It should only be required if needed, but it should never be prohibited.129164 `_POSIX_REALTIME_SIGNALS`

129165 Support for realtime signals is mandatory in POSIX.1-2008.

129166 This facility provides prioritized, queued signals with associated data values.

129167 `_POSIX_REGEX`

129168 Support for regular expression facilities is mandatory in POSIX.1-2008.

- 129169 `_POSIX_SAVED_IDS`  
 129170 Support for this feature is mandatory in POSIX.1-2008.
- 129171 Certain classes of applications rely on it for proper operation, and there is no alternative  
 129172 short of giving the application root privileges on most implementations that did not provide  
 129173 `_POSIX_SAVED_IDS`.
- 129174 `_POSIX_SEMAPHORES`  
 129175 Support for counting semaphores is mandatory in POSIX.1-2008.
- 129176 `_POSIX_SHARED_MEMORY_OBJECTS`  
 129177 The system supports the mapping of shared memory objects into the process address space.
- 129178 Both this option and the Memory Mapped Files option provide shared access to memory  
 129179 objects in the process address space. The functions defined under this option provide the  
 129180 functionality of existing practice for shared memory objects. This functionality was deemed  
 129181 appropriate for embedded systems applications and, hence, is provided under this option.  
 129182 It should only be required if needed, but it should never be prohibited.
- 129183 `_POSIX_SHELL`  
 129184 Support for the *sh* utility command line interpreter is mandatory in POSIX.1-2008.
- 129185 `_POSIX_SPAWN`  
 129186 The system supports the spawn option.
- 129187 This option provides applications with an efficient mechanism to spawn execution of a new  
 129188 process.
- 129189 `_POSIX_SPINLOCKS`  
 129190 Support for spin locks is mandatory in POSIX.1-2008.
- 129191 This facility provides a simple and efficient synchronization mechanism for threads  
 129192 executing in multi-processor systems.
- 129193 `_POSIX_SPORADIC_SERVER`  
 129194 The system supports the sporadic server scheduling policy.
- 129195 This option provides applications with a new scheduling policy for scheduling aperiodic  
 129196 processes or threads in hard realtime applications.
- 129197 `_POSIX_SYNCHRONIZED_IO`  
 129198 The system supports guaranteed file synchronization.
- 129199 This option was created to support historical systems that did not provide the feature.  
 129200 Applications that are expecting guaranteed completion of their input and output operations  
 129201 should require this option, rather than the File Synchronization option. It should only be  
 129202 required if needed, but it should never be prohibited.
- 129203 `_POSIX_THREADS`  
 129204 Support for multiple threads of control within a single process is mandatory in  
 129205 POSIX.1-2008.
- 129206 `_POSIX_THREAD_ATTR_STACKADDR`  
 129207 The system supports specification of the stack address for a created thread.
- 129208 Applications may take advantage of support of this option for performance benefits, but  
 129209 dependence on this feature should be minimized. This option should never be prohibited.
- 129210 XSI-conformant systems support this option.

129211 `_POSIX_THREAD_ATTR_STACKSIZE`

129212 The system supports specification of the stack size for a created thread.

129213 Applications may require this option in order to ensure proper execution, but such usage  
129214 limits portability and dependence on this feature should be minimized. It should only be  
129215 required if needed, but it should never be prohibited.

129216 XSI-conformant systems support this option.

129217 `_POSIX_THREAD_PRIORITY_SCHEDULING`

129218 The system provides priority-based thread scheduling.

129219 Support of this option provides predictable scheduling behavior, allowing applications to  
129220 determine the order in which threads that are ready to run are granted access to a processor.  
129221 It should only be required if needed, but it should never be prohibited.

129222 `_POSIX_THREAD_PRIO_INHERIT`

129223 The system provides mutual-exclusion operations with priority inheritance.

129224 Support of this option provides predictable scheduling behavior, allowing applications to  
129225 determine the order in which threads that are ready to run are granted access to a processor.  
129226 It should only be required if needed, but it should never be prohibited.

129227 `_POSIX_THREAD_PRIO_PROTECT`

129228 The system supports a priority ceiling emulation protocol for mutual-exclusion operations.

129229 Support of this option provides predictable scheduling behavior, allowing applications to  
129230 determine the order in which threads that are ready to run are granted access to a processor.  
129231 It should only be required if needed, but it should never be prohibited.

129232 `_POSIX_THREAD_PROCESS_SHARED`

129233 The system provides shared access among multiple processes to synchronization objects.

129234 This option was created to support historical systems that did not provide the feature. It  
129235 should only be required if needed, but it should never be prohibited.

129236 XSI-conformant systems support this option.

129237 `_POSIX_THREAD_SAFE_FUNCTIONS`

129238 Support for thread-safe functions is mandatory in POSIX.1-2008.

129239 `_POSIX_THREAD_SPORADIC_SERVER`

129240 The system supports the thread sporadic server scheduling policy.

129241 Support for this option provides applications with a new scheduling policy for scheduling  
129242 aperiodic threads in hard realtime applications.

129243 `_POSIX_TIMEOUTS`

129244 Support for timeouts for some blocking services is mandatory in POSIX.1-2008.

129245 `_POSIX_TIMERS`

129246 Support for higher resolution clocks with multiple timers per process is mandatory in  
129247 POSIX.1-2008.

129248 This facility is appropriate for applications requiring higher resolution timestamps or  
129249 needing to control the timing of multiple activities.

129250 `_POSIX_TRACE`

129251 The system supports the Trace option.

129252 This option was created to allow applications to perform tracing.

- 129253 `_POSIX_TRACE_EVENT_FILTER`  
 129254 The system supports the Trace Event Filter option.  
 129255 This option is dependent on support of the Trace option.
- 129256 `_POSIX_TRACE_INHERIT`  
 129257 The system supports the Trace Inherit option.  
 129258 This option is dependent on support of the Trace option.
- 129259 `_POSIX_TRACE_LOG`  
 129260 The system supports the Trace Log option.  
 129261 This option is dependent on support of the Trace option.
- 129262 `_POSIX_TYPED_MEMORY_OBJECTS`  
 129263 The system supports the Typed Memory Objects option.  
 129264 This option was created to allow realtime applications to access different kinds of physical  
 129265 memory, and allow processes in these applications to share portions of this memory.

### 129266 D.3.5 Configurable Limits

- 129267 In general, the configurable limits in the `<limits.h>` header defined in the Base Definitions  
 129268 volume of POSIX.1-2008 have been set to minimal values; many applications or implementations  
 129269 may require larger values. No profile can cite lower values.
- 129270 `{AIO_LISTIO_MAX}`  
 129271 The current minimum is likely to be inadequate for most applications. It is expected that  
 129272 this value will be increased by profiles requiring support for list input and output  
 129273 operations.
- 129274 `{AIO_MAX}`  
 129275 The current minimum is likely to be inadequate for most applications. It is expected that  
 129276 this value will be increased by profiles requiring support for asynchronous input and  
 129277 output operations.
- 129278 `{AIO_PRIO_DELTA_MAX}`  
 129279 The functionality associated with this limit is needed only by sophisticated applications. It  
 129280 is not expected that this limit would need to be increased under a general-purpose profile.
- 129281 `{ARG_MAX}`  
 129282 The current minimum is likely to need to be increased for profiles, particularly as larger  
 129283 amounts of information are passed through the environment. Many implementations are  
 129284 believed to support larger values.
- 129285 `{CHILD_MAX}`  
 129286 The current minimum is suitable only for systems where a single user is not running  
 129287 applications in parallel. It is significantly too low for any system also requiring windows,  
 129288 and if `_POSIX_JOB_CONTROL` is specified, it should be raised.
- 129289 `{CLOCKRES_MIN}`  
 129290 It is expected that profiles will require a finer granularity clock, perhaps as fine as 1  $\mu$ s,  
 129291 represented by a value of 1 000 for this limit.
- 129292 `{DELAYTIMER_MAX}`  
 129293 It is believed that most implementations will provide larger values.

129294	{LINK_MAX}
129295	For most applications and usage, the current minimum is adequate. Many implementations
129296	have a much larger value, but this should not be used as a basis for raising the value unless
129297	the applications to be used require it.
129298	{LOGIN_NAME_MAX}
129299	This is not actually a limit, but an implementation parameter. No profile should impose a
129300	requirement on this value.
129301	{MAX_CANON}
129302	For most purposes, the current minimum is adequate. Unless high-speed burst serial
129303	devices are used, it should be left as is.
129304	{MAX_INPUT}
129305	See {MAX_CANON}.
129306	{MQ_OPEN_MAX}
129307	The current minimum should be adequate for most profiles.
129308	{MQ_PRIO_MAX}
129309	The current minimum corresponds to the required number of process scheduling priorities.
129310	Many realtime practitioners believe that the number of message priority levels ought to be
129311	the same as the number of execution scheduling priorities.
129312	{NAME_MAX}
129313	Many implementations now support larger values, and many applications and users
129314	assume that larger names can be used. Many existing profiles also specify a larger value.
129315	Specifying this value will reduce the number of conforming implementations, although this
129316	might not be a significant consideration over time. Values greater than 255 should not be
129317	required.
129318	{NGROUPS_MAX}
129319	The value selected will typically be 8 or larger.
129320	{OPEN_MAX}
129321	The historically common value for this has been 20. Many implementations support larger
129322	values. If applications that use larger values are anticipated, an appropriate value should be
129323	specified.
129324	{PAGESIZE}
129325	This is not actually a limit, but an implementation parameter. No profile should impose a
129326	requirement on this value.
129327	{PATH_MAX}
129328	Historically, the minimum has been either 1024 or indefinite, depending on the
129329	implementation. Few applications actually require values larger than 256, but some users
129330	may create file hierarchies that must be accessed with longer paths. This value should only
129331	be changed if there is a clear requirement.
129332	{PIPE_BUF}
129333	The current minimum is adequate for most applications. Historically, it has been larger. If
129334	applications that write single transactions larger than this are anticipated, it should be
129335	increased. Applications that write lines of text larger than this probably do not need it
129336	increased, as the text line is delimited by a <newline>.
129337	{POSIX_VERSION}
129338	This is actually not a limit, but a standard version stamp. Generally, a profile should specify
129339	POSIX.1-2008 by a name in the normative references section, not this value.

129340 {PTHREAD\_DESTRUCTOR\_ITERATIONS}  
 129341 It is unlikely that applications will need larger values to avoid loss of memory resources.

129342 {PTHREAD\_KEYS\_MAX}  
 129343 The current value should be adequate for most profiles.

129344 {PTHREAD\_STACK\_MIN}  
 129345 This should not be treated as an actual limit, but as an implementation parameter. No  
 129346 profile should impose a requirement on this value.

129347 {PTHREAD\_THREADS\_MAX}  
 129348 It is believed that most implementations will provide larger values.

129349 {RTSIG\_MAX}  
 129350 The current limit was chosen so that the set of POSIX.1 signal numbers can fit within a  
 129351 32-bit field. It is recognized that most existing implementations define many more signals  
 129352 than are specified in POSIX.1 and, in fact, many implementations have already exceeded 32  
 129353 signals (including the “null signal”). Support of {\_POSIX\_RTSIG\_MAX} additional signals  
 129354 may push some implementations over the single 32-bit word line, but is unlikely to push  
 129355 any implementations that are already over that line beyond the 64 signal line.

129356 {SEM\_NSEMS\_MAX}  
 129357 The current value should be adequate for most profiles.

129358 {SEM\_VALUE\_MAX}  
 129359 The current value should be adequate for most profiles.

129360 {SSIZE\_MAX}  
 129361 This limit reflects fundamental hardware characteristics (the size of an integer), and should  
 129362 not be specified unless it is clearly required. Extreme care should be taken to assure that  
 129363 any value that might be specified does not unnecessarily eliminate implementations  
 129364 because of accidents of hardware design.

129365 {STREAM\_MAX}  
 129366 This limit is very closely related to {OPEN\_MAX}. It should never be larger than  
 129367 {OPEN\_MAX}, but could reasonably be smaller for application areas where most files are  
 129368 not accessed through *stdio*. Some implementations may limit {STREAM\_MAX} to 20 but  
 129369 allow {OPEN\_MAX} to be considerably larger. Such implementations should be allowed for  
 129370 if the applications permit.

129371 {TIMER\_MAX}  
 129372 The current limit should be adequate for most profiles, but it may need to be larger for  
 129373 applications with a large number of asynchronous operations.

129374 {TTY\_NAME\_MAX}  
 129375 This is not actually a limit, but an implementation parameter. No profile should impose a  
 129376 requirement on this value.

129377 {TZNAME\_MAX}  
 129378 The minimum has been historically adequate, but if longer timezone names are anticipated  
 129379 (particularly such values as UTC-1), this should be increased.

129380 **D.3.6 Optional Behavior**

129381 In POSIX.1-2008, there are no instances of the terms unspecified, undefined, implementation-  
129382 defined, or with the verbs “may” or “need not”, that the standard developers anticipate or  
129383 sanction as suitable for profile or test method citation. All of these are merely warnings to  
129384 conforming applications to avoid certain areas that can vary from system to system, and even  
129385 over time on the same system. In many cases, these terms are used explicitly to support  
129386 extensions, but profiles should not anticipate and require such extensions; future versions of this  
129387 standard may do so.





129388

# *Rationale (Informative)*

129389

## **Part E:**

129390

## **Subprofiling Considerations**

129391

*The Open Group*

129392

*The Institute of Electrical and Electronics Engineers, Inc.*



129393

Appendix E

129394

## Subprofiling Considerations (Informative)

129395  
129396  
129397  
129398

This section contains further information to satisfy the requirement that the project scope enable subprofiling of POSIX.1-2008. The approach taken is to include a general requirement in normative text regarding subprofiling and to include an informative section (here) containing a proposed set of subprofiling options.

### E.1 Subprofiling Option Groups

129400  
129401  
129402  
129403

The following Option Groups<sup>11</sup> are defined to support profiling. Systems claiming support to POSIX.1-2008 need not implement these options apart from the requirements stated in XBD Section 2.1.3 (on page 17). These Option Groups allow profiles to subset the System Interfaces volume of POSIX.1-2008 by collecting sets of related functions.

129404  
129405  
129406

POSIX\_ASYNCHRONOUS\_IO: Asynchronous Input and Output Functions

*aio\_cancel()*, *aio\_error()*, *aio\_fsync()*, *aio\_read()*, *aio\_return()*, *aio\_suspend()*, *aio\_write()*,  
*lio\_listio()*

129407  
129408

POSIX\_BARRIERS: Barriers

*pthread\_barrier\_destroy()*, *pthread\_barrier\_init()*, *pthread\_barrier\_wait()*, *pthread\_barrierattr()*

129409  
129410

POSIX\_C\_LANG\_JUMP: Jump Functions

*longjmp()*, *setjmp()*

129411

POSIX\_C\_LANG\_MATH: Maths Library

129412 *acos()*, *acosf()*, *acosh()*, *acoshf()*, *acoshl()*, *acosl()*, *asin()*, *asinf()*, *asinh()*, *asinhf()*, *asinhf()*, *asinhf()*,  
129413 *asinl()*, *atan()*, *atan2()*, *atan2f()*, *atan2l()*, *atanf()*, *atanh()*, *atanhf()*, *atanhl()*, *atanl()*, *cabs()*,  
129414 *cabsf()*, *cabsl()*, *cacos()*, *cacosf()*, *cacosh()*, *cacoshf()*, *cacoshl()*, *cacosl()*, *carg()*, *cargf()*, *cargl()*,  
129415 *casin()*, *casinf()*, *casinh()*, *casinhf()*, *casinhf()*, *casinl()*, *catan()*, *catanf()*, *catanh()*, *catanhf()*,  
129416 *catanhf()*, *catanhf()*, *catanhf()*, *catanhf()*, *ccos()*, *ccosf()*, *ccosh()*, *ccoshf()*, *ccoshl()*, *ccosl()*,  
129417 *ceil()*, *ceilf()*, *ceilf()*, *cexp()*, *cexpf()*, *cexpl()*, *cimag()*, *cimagf()*, *cimagf()*, *clog()*, *clogf()*, *clogl()*,  
129418 *conj()*, *conjf()*, *conjl()*, *copysign()*, *copysignf()*, *copysignf()*, *cos()*, *cosf()*, *cosh()*, *coshf()*,  
129419 *coshf()*, *cosl()*, *cpow()*, *cpowf()*, *cpowl()*, *cproj()*, *cprojf()*, *cprojf()*, *creal()*, *crealf()*, *creall()*,  
129420 *csin()*, *csinf()*, *csinh()*, *csinhf()*, *csinhf()*, *csinl()*, *csqrt()*, *csqrtf()*, *csqrtf()*, *ctan()*, *ctanf()*,  
129421 *ctanh()*, *ctanhf()*, *ctanhf()*, *ctanhf()*, *erf()*, *erfc()*, *erfcf()*, *erfcf()*, *erff()*, *erfl()*, *exp()*, *exp2()*,  
129422 *exp2f()*, *exp2l()*, *expf()*, *expl()*, *expm1()*, *expm1f()*, *expm1f()*, *fabs()*, *fabsf()*, *fabsf()*, *fdim()*,  
129423 *fdimf()*, *fdimf()*, *floor()*, *floorf()*, *floorf()*, *fma()*, *fmaf()*, *fmaf()*, *fmax()*, *fmaxf()*, *fmaxf()*, *fmin()*,  
129424 *fminf()*, *fminf()*, *fmod()*, *fmodf()*, *fmodf()*, *fpclassify()*, *frexp()*, *frexp()*, *frexp()*, *hypot()*,  
129425 *hypotf()*, *hypotf()*, *ilogb()*, *ilogbf()*, *ilogbf()*, *isfinite()*, *isgreater()*, *isgreater()*, *isinf()*,  
129426 *isless()*, *islessequal()*, *islessgreater()*, *isnan()*, *isnormal()*, *isunordered()*, *ldexp()*, *ldexpf()*,  
129427 *ldexpl()*, *lgamma()*, *lgammaf()*, *lgammaf()*, *llrint()*, *llrintf()*, *llrintf()*, *llround()*, *llroundf()*,  
129428 *llroundf()*, *log()*, *log10()*, *log10f()*, *log10l()*, *log1p()*, *log1pf()*, *log1pf()*, *log2()*, *log2f()*, *log2l()*,  
129429 *logb()*, *logbf()*, *logbf()*, *logf()*, *logl()*, *lrint()*, *lrintf()*, *lrintf()*, *lround()*, *lroundf()*, *lroundf()*,  
129430 *modf()*, *modff()*, *modff()*, *nan()*, *nanf()*, *nanf()*, *nearbyint()*, *nearbyintf()*, *nearbyintf()*,  
129431 *nextafter()*, *nextafterf()*, *nextafterf()*, *nexttoward()*, *nexttowardf()*, *nexttowardf()*, *pow()*, *powf()*,

129432 11. These are modeled on the Units of Functionality from IEEE Std 1003.13-1998.

129433 *powl(), remainder(), remainderf(), remainderl(), remquo(), remquoof(), remquol(), rint(), rintf(),*  
 129434 *rintl(), round(), roundf(), roundl(), scalbln(), scalblnf(), scalblnl(), scalbn(), scalbnf(),*  
 129435 *scalbnl(), signbit(), sin(), sinf(), sinh(), sinhf(), sinhl(), sinl(), sqrt(), sqrtf(), sqrtl(), tan(),*  
 129436 *tanf(), tanh(), tanhf(), tanhl(), tanl(), tgamma(), tgammaf(), tgammaL(), trunc(), truncf(),*  
 129437 *truncl()*

#### 129438 POSIX\_C\_LANG\_SUPPORT: General ISO C Library

129439 *abs(), asctime(), atof(), atoi(), atol(), atoll(), bsearch(), calloc(), ctime(), difftime(), div(),*  
 129440 *feclearexcept(), fegetenv(), fegetexceptflag(), fegetround(), feholdexcept(), feraiseexcept(),*  
 129441 *fesetenv(), fesetexceptflag(), fesetround(), fetestexcept(), feupdateenv(), free(), gmtime(),*  
 129442 *imaxabs(), imaxdiv(), isalnum(), isalpha(), isblank(), iscntrl(), isdigit(), isgraph(), islower(),*  
 129443 *isprint(), ispunct(), isspace(), isupper(), isxdigit(), labs(), ldiv(), llabs(), lldiv(), localeconv(),*  
 129444 *localtime(), malloc(), memchr(), memcmp(), memcpy(), memmove(), memset(), mktime(),*  
 129445 *qsort(), rand(), realloc(), setlocale(), snprintf(), sprintf(), srand(), sscanf(), strcat(), strchr(),*  
 129446 *strcmp(), strcoll(), strcpy(), strcspn(), strerror(), strtime(), strlen(), strncat(), strncmp(),*  
 129447 *strncpy(), strpbrk(), strrchr(), strspn(), strstr(), strtod(), strtodf(), strtointmax(), strtok(), strtol(),*  
 129448 *strtold(), strtoll(), strtoul(), strtoull(), strtoumax(), strxfrm(), time(), tolower(), toupper(),*  
 129449 *tzname, tzset(), va\_arg(), va\_copy(), va\_end(), va\_start(), vsnprintf(), vsprintf(), vsscanf()*

#### 129450 POSIX\_C\_LANG\_SUPPORT\_R: Thread-Safe General ISO C Library

129451 *asctime\_r(), ctime\_r(), gmtime\_r(), localtime\_r(), rand\_r(), strerror\_r(), strtok\_r()*

#### 129452 POSIX\_C\_LANG\_WIDE\_CHAR: Wide-Character ISO C Library

129453 *btowc(), iswalnum(), iswalpalpha(), iswblank(), iswcntrl(), iswctype(), iswdigit(), iswgraph(),*  
 129454 *iswlower(), iswprint(), iswpunct(), iswspace(), iswupper(), iswxdigit(), mblen(), mbrlen(),*  
 129455 *mbrtowc(), mbsinit(), mbsrtowcs(), mbstowcs(), mbtowc(), swprintf(), swscanf(), towctrans(),*  
 129456 *towlower(), towupper(), vswprintf(), vswscanf(), wctomb(), wcscat(), wcschr(), wcscmp(),*  
 129457 *wscoll(), wcscpy(), wcscspn(), wcsftime(), wcslen(), wcsncat(), wcsncmp(), wcsncpy(),*  
 129458 *wcspbrk(), wcsrchr(), wcstombs(), wcspn(), wcsstr(), wcstod(), wcstof(), wcstointmax(),*  
 129459 *wcstok(), wcstol(), wcstold(), wcstoll(), wcstombs(), wcstoul(), wcstoull(), wcstoumax(),*  
 129460 *wcsxfrm(), wctob(), wctomb(), wctrans(), wctype(), wmemchr(), wmemcmp(), wmemcpy(),*  
 129461 *wmemmove(), wmemset()*

#### 129462 POSIX\_C\_LANG\_WIDE\_CHAR\_EXT: Extended Wide-Character ISO C Library

129463 *mbsnrtowcs(), wcpncpy(), wcpncpy(), wcscasecmp(), wcsdup(), wcsncasecmp(), wcsnlen(),*  
 129464 *wcsnrtombs()*

#### 129465 POSIX\_C\_LIB\_EXT: General C Library Extension

129466 *fnmatch(), getopt(), getsubopt(), optarg, opterr, optind, optopt, stpcpy(), stpncpy(), strcasecmp(),*  
 129467 *strdup(), strfmon(), strncasecmp(), strndup(), strnlen()*

#### 129468 POSIX\_CLOCK\_SELECTION: Clock Selection

129469 *clock\_nanosleep(), pthread\_condattr\_getclock(), pthread\_condattr\_setclock()*

#### 129470 POSIX\_DEVICE\_IO: Device Input and Output

129471 *FD\_CLR(), FD\_ISSET(), FD\_SET(), FD\_ZERO(), clearerr(), close(), fclose(), fdopen(), feof(),*  
 129472 *ferror(), fflush(), fgetc(), fgets(), fileno(), fopen(), fprintf(), fputc(), fputs(), fread(), freopen(),*  
 129473 *fscanf(), fwrite(),getc(), getchar(), gets(), open(), perror(), poll(), printf(), pread(), pselect(),*  
 129474 *putc(), putchar(), puts(), pwrite(), read(), scanf(), select(), setbuf(), setvbuf(), stderr, stdin,*  
 129475 *stdout, ungetc(), vfprintf(), vfscanf(), vprintf(), vscanf(), write()*

#### 129476 POSIX\_DEVICE\_IO\_EXT: Extended Device Input and Output

129477 *dprintf(), fmemopen(), open\_memstream(), vdprintf()*

#### 129478 POSIX\_DEVICE\_SPECIFIC: General Terminal

129479 *cfgetispeed(), cfgetospeed(), cfsetispeed(), cfsetospeed(), ctermid(), isatty(), tcdrain(), tcflow(),*  
 129480 *tcflush(), tcgetattr(), tcsetattr(), tcsetpgrp(), ttyname()*

129481	POSIX_DEVICE_SPECIFIC_R: Thread-Safe General Terminal
129482	<i>ttyname_r()</i>
129483	POSIX_DYNAMIC_LINKING: Dynamic Linking
129484	<i>dlclose(), dlderror(), dlopen(), dlsym()</i>
129485	POSIX_FD_MGMT: File Descriptor Management
129486	<i>dup(), dup2(), fcntl(), fgetpos(), fseek(), fseeko(), fsetpos(), ftell(), ftello(), ftruncate(), lseek(),</i>
129487	<i>rewind()</i>
129488	POSIX_FIFO: FIFO
129489	<i>mkfifo()</i>
129490	POSIX_FIFO_FD: FIFO File Descriptor Routines
129491	<i>mkfifoat(), mknodat()</i>
129492	POSIX_FILE_ATTRIBUTES: File Attributes
129493	<i>chmod(), chown(), fchmod(), fchown(), umask()</i>
129494	POSIX_FILE_ATTRIBUTES_FD: File Attributes File Descriptor Routines
129495	<i>fchmodat(), fchownat()</i>
129496	POSIX_FILE_LOCKING: Thread-Safe Stdio Locking
129497	<i>flockfile(), frylockfile(), funlockfile(), getc_unlocked(), getchar_unlocked(), putc_unlocked(),</i>
129498	<i>putchar_unlocked()</i>
129499	POSIX_FILE_SYSTEM: File System
129500	<i>access(), chdir(), closedir(), creat(), fchdir(), fpathconf(), fstat(), fstatvfs(), getcwd(), link(),</i>
129501	<i>mkdir(), mkstemp(), opendir(), pathconf(), readdir(), remove(), rename(), rewinddir(), rmdir(),</i>
129502	<i>stat(), statvfs(), tmpfile(), tmpnam(), truncate(), unlink(), utime()</i>
129503	POSIX_FILE_SYSTEM_EXT: File System Extensions
129504	<i>alphasort(), dirfd(), getdelim(), getline(), mkdtemp(), scandir()</i>
129505	POSIX_FILE_SYSTEM_FD: File System File Descriptor Routines
129506	<i>faccessat(), fdopendir(), fstatat(), linkat(), mkdirat(), openat(), renameat(), unlinkat(),</i>
129507	<i>utimensat()</i>
129508	POSIX_FILE_SYSTEM_GLOB: File System Glob Expansion
129509	<i>glob(), globfree()</i>
129510	POSIX_FILE_SYSTEM_R: Thread-Safe File System
129511	<i>readdir_r()</i>
129512	POSIX_I18N: Internationalization
129513	<i>catclose(), catgets(), catopen(), iconv(), iconv_close(), iconv_open(), nl_langinfo()</i>
129514	POSIX_JOB_CONTROL: Job Control
129515	<i>setpgid(), tcgetpgrp(), tcsetpgrp(), tcgetsid()</i>
129516	POSIX_MAPPED_FILES: Memory Mapped Files
129517	<i>mmap(), munmap()</i>
129518	POSIX_MEMORY_PROTECTION: Memory Protection
129519	<i>mprotect()</i>
129520	POSIX_MULTI_CONCURRENT_LOCALES: Multiple Concurrent Locales
129521	<i>duplocale(), freelocale(), isalnum_l(), isalpha_l(), isblank_l(), iscntrl_l(), isdigit_l(), isgraph_l(),</i>
129522	<i>islower_l(), isprint_l(), ispunct_l(), isspace_l(), isupper_l(), iswalnum_l(), iswalalpha_l(),</i>
129523	<i>iswblank_l(), iswcntrl_l(), iswctype_l(), iswdigit_l(), iswgraph_l(), iswlower_l(), iswprint_l(),</i>
129524	<i>iswpunct_l(), iswspace_l(), iswupper_l(), iswxdigit_l(), isxdigit_l(), newlocale(), strcasecmp_l(),</i>

129525 *strcoll\_l()*, *strfmon\_l()*, *strncasecmp\_l()*, *strxfrm\_l()*, *tolower\_l()*, *toupper\_l()*, *towctrans\_l()*,  
129526 *towlower()*, *towupper()*, *uselocale()*, *wscasecmp\_l()*, *wscoll\_l()*, *wcsncasecmp\_l()*, *wcsxfrm\_l()*,  
129527 *wctrans\_l()*, *wctype\_l()*

129528 POSIX\_MULTI\_PROCESS: Multiple Processes  
129529 *\_Exit()*, *\_exit()*, *assert()*, *atexit()*, *clock()*, *execl()*, *execle()*, *execlp()*, *execv()*, *execve()*, *execvp()*,  
129530 *exit()*, *fork()*, *getpgrp()*, *getpgid()*, *getpid()*, *getppid()*, *getsid()*, *setsid()*, *sleep()*, *times()*, *wait()*,  
129531 *waitid()*, *waitpid()*

129532 POSIX\_MULTI\_PROCESS\_FD: Multiple Processes File Descriptor Routines  
129533 *fxexecve()*

129534 POSIX\_NETWORKING: Networking  
129535 *accept()*, *bind()*, *connect()*, *endhostent()*, *endnetent()*, *endprotoent()*, *endservent()*,  
129536 *freeaddrinfo()*, *gai\_strerror()*, *getaddrinfo()*, *gethostent()*, *gethostname()*, *getnameinfo()*, -  
129537 *getnetbyaddr()*, *getnetbyname()*, *getnetent()*, *getpeername()*, *getprotobyname()*,  
129538 *getprotobynumber()*, *getprotoent()*, *getservbyname()*, *getservbyport()*, *getservent()*,  
129539 *getsockname()*, *getsockopt()*, *htonl()*, *htons()*, *if\_freenameindex()*, *if\_indextoname()*, -  
129540 *if\_nameindex()*, *if\_nametoindex()*, *inet\_addr()*, *inet\_ntoa()*, *inet\_ntop()*, *inet\_pton()*, *listen()*,  
129541 *ntohl()*, *ntohs()*, *recv()*, *recvfrom()*, *recvmsg()*, *send()*, *sendmsg()*, *sendto()*, *sethostent()*,  
129542 *setnetent()*, *setprotoent()*, *setservent()*, *setsockopt()*, *shutdown()*, *socket()*, *socketatmark()*,  
129543 *socketpair()*

129544 POSIX\_PIPE: Pipe  
129545 *pipe()*

129546 POSIX\_ROBUST\_MUTEXES: Robust Mutexes  
129547 *pthread\_mutex\_consistent()*, *pthread\_mutexattr\_getrobust()*, *pthread\_mutexattr\_setrobust()*

129548 POSIX\_REALTIME\_SIGNALS: Realtime Signals  
129549 *sigqueue()*, *sigtimedwait()*, *sigwaitinfo()*

129550 POSIX\_REGEX: Regular Expressions  
129551 *regcomp()*, *regerror()*, *regexec()*, *regfree()*

129552 POSIX\_RW\_LOCKS: Reader Writer Locks  
129553 *pthread\_rwlock\_destroy()*, *pthread\_rwlock\_init()*, *pthread\_rwlock\_rdlock()*,  
129554 *pthread\_rwlock\_timedrdlock()*, *pthread\_rwlock\_timedwrlock()*, *pthread\_rwlock\_tryrdlock()*,  
129555 *pthread\_rwlock\_trywrlock()*, *pthread\_rwlock\_unlock()*, *pthread\_rwlock\_wrlock()*,  
129556 *pthread\_rwlockattr\_destroy()*, *pthread\_rwlockattr\_init()*, *pthread\_rwlockattr\_getpshared()*,  
129557 *pthread\_rwlockattr\_setpshared()*

129558 POSIX\_SEMAPHORES: Semaphores  
129559 *sem\_close()*, *sem\_destroy()*, *sem\_getvalue()*, *sem\_init()*, *sem\_open()*, *sem\_post()*,  
129560 *sem\_timedwait()*, *sem\_trywait()*, *sem\_unlink()*, *sem\_wait()*

129561 POSIX\_SHELL\_FUNC: Shell and Utilities  
129562 *pclose()*, *popen()*, *system()*, *wordexp()*, *wordfree()*

129563 POSIX\_SIGNAL\_JUMP: Signal Jump Functions  
129564 *siglongjmp()*, *sigsetjmp()*

129565 POSIX\_SIGNALS: Signals  
129566 *abort()*, *alarm()*, *kill()*, *pause()*, *raise()*, *sigaction()*, *sigaddset()*, *sigdelset()*, *sigemptyset()*,  
129567 *sigfillset()*, *sigismember()*, *signal()*, *sigpending()*, *sigprocmask()*, *sigsuspend()*, *sigwait()*

129568 POSIX\_SIGNALS\_EXT: Extended Signals  
129569 *psignal()*, *psiginfo()*, *strsignal()*

129570	POSIX_SINGLE_PROCESS: Single Process	
129571	<i>confstr()</i> , <i>environ</i> , <i>errno</i> , <i>getenv()</i> , <i>setenv()</i> , <i>sysconf()</i> , <i>uname()</i> , <i>unsetenv()</i>	
129572	POSIX_SPIN_LOCKS: Spin Locks	
129573	<i>pthread_spin_destroy()</i> , <i>pthread_spin_init()</i> , <i>pthread_spin_lock()</i> , <i>pthread_spin_trylock()</i> ,	
129574	<i>pthread_spin_unlock()</i>	
129575	POSIX_SYMBOLIC_LINKS: Symbolic Links	
129576	<i>lchown()</i> , <sup>12</sup> <i>lstat()</i> , <i>readlink()</i> , <i>symlink()</i>	
129577	POSIX_SYMBOLIC_LINKS_FD: Symbolic Links File Descriptor Routines	
129578	<i>readlinkat()</i> , <i>symlinkat()</i>	
129579	POSIX_SYSTEM_DATABASE: System Database	
129580	<i>getgrgid()</i> , <i>getgrnam()</i> , <i>getpwnam()</i> , <i>getpwuid()</i>	
129581	POSIX_SYSTEM_DATABASE_R: Thread-Safe System Database	
129582	<i>getgrgid_r()</i> , <i>getgrnam_r()</i> , <i>getpwnam_r()</i> , <i>getpwuid_r()</i>	
129583	POSIX_THREADS_BASE: Base Threads	
129584	<i>pthread_atfork()</i> , <i>pthread_attr_destroy()</i> , <i>pthread_attr_getdetachstate()</i> ,	
129585	<i>pthread_attr_getschedparam()</i> , <i>pthread_attr_init()</i> , <i>pthread_attr_setdetachstate()</i> ,	
129586	<i>pthread_attr_setschedparam()</i> , <i>pthread_cancel()</i> , <i>pthread_cleanup_pop()</i> , <i>pthread_cleanup_push()</i> ,	
129587	<i>pthread_cond_broadcast()</i> , <i>pthread_cond_destroy()</i> , <i>pthread_cond_init()</i> , <i>pthread_cond_signal()</i> ,	
129588	<i>pthread_cond_timedwait()</i> , <i>pthread_cond_wait()</i> , <i>pthread_condattr_destroy()</i> ,	
129589	<i>pthread_condattr_init()</i> , <i>pthread_create()</i> , <i>pthread_detach()</i> , <i>pthread_equal()</i> , <i>pthread_exit()</i> ,	
129590	<i>pthread_getspecific()</i> , <i>pthread_join()</i> , <i>pthread_key_create()</i> , <i>pthread_key_delete()</i> , <i>pthread_kill()</i> ,	
129591	<i>pthread_mutex_destroy()</i> , <i>pthread_mutex_init()</i> , <i>pthread_mutex_lock()</i> ,	
129592	<i>pthread_mutex_timedlock()</i> , <i>pthread_mutex_trylock()</i> , <i>pthread_mutex_unlock()</i> ,	
129593	<i>pthread_mutexattr_destroy()</i> , <i>pthread_mutexattr_init()</i> , <i>pthread_once()</i> , <i>pthread_self()</i> ,	
129594	<i>pthread_setcancelstate()</i> , <i>pthread_setcanceltype()</i> , <i>pthread_setspecific()</i> , <i>pthread_sigmask()</i> ,	
129595	<i>pthread_testcancel()</i>	
129596	POSIX_THREADS_EXT: Extended Threads	
129597	<i>pthread_attr_getguardsize()</i> , <i>pthread_attr_setguardsize()</i> , <i>pthread_mutexattr_gettype()</i> ,	
129598	<i>pthread_mutexattr_settype()</i>	
129599	POSIX_TIMERS: Timers	
129600	<i>clock_getres()</i> , <i>clock_gettime()</i> , <i>clock_settime()</i> , <i>nanosleep()</i> , <i>timer_create()</i> , <i>timer_delete()</i> ,	
129601	<i>timer_getoverrun()</i> , <i>timer_gettime()</i> , <i>timer_settime()</i>	
129602	POSIX_USER_GROUPS: User and Group	
129603	<i>getegid()</i> , <i>geteuid()</i> , <i>getgid()</i> , <i>getgroups()</i> , <i>getlogin()</i> , <i>getuid()</i> , <i>setegid()</i> , <i>seteuid()</i> , <i>setgid()</i> ,	
129604	<i>setuid()</i>	
129605	POSIX_USER_GROUPS_R: Thread-Safe User and Group	
129606	<i>getlogin_r()</i>	
129607	POSIX_WIDE_CHAR_DEVICE_IO: Device Input and Output	
129608	<i>fgetwc()</i> , <i>fgetws()</i> , <i>fputwc()</i> , <i>fputws()</i> , <i>fwide()</i> , <i>fwprintf()</i> , <i>fwscanf()</i> , <i>getwc()</i> , <i>getwchar()</i> ,	
129609	<i>putwc()</i> , <i>putwchar()</i> , <i>ungetwc()</i> , <i>vfwprintf()</i> , <i>vfwscanf()</i> , <i>vwprintf()</i> , <i>vwscanf()</i> , <i>wprintf()</i> ,	
129610	<i>wscanf()</i>	
129611	XSI_C_LANG_SUPPORT: XSI General C Library	
129612	<i>_tolower()</i> , <i>_toupper()</i> , <i>a64l()</i> , <i>daylight()</i> , <i>drand48()</i> , <i>erand48()</i> , <i>ffs()</i> , <i>getdate()</i> , <i>hcreate()</i> ,	-
129613	<i>hdestroy()</i> , <i>hsearch()</i> , <i>initstate()</i> , <i>insque()</i> , <i>isascii()</i> , <i>jrand48()</i> , <i>l64a()</i> , <i>lcong48()</i> , <i>lfind()</i> ,	
129614	<i>lrand48()</i> , <i>lsearch()</i> , <i>memccpy()</i> , <i>mrand48()</i> , <i>rand48()</i> , <i>random()</i> , <i>remque()</i> , <i>seed48()</i> ,	-

---

129615 12. The *lchown()* function also depends on POSIX\_FILE\_ATTRIBUTES.



129616	<i>setstate()</i> , <i>signgam</i> , <i>srand48()</i> , <i>srandom()</i> , <i>strptime()</i> , <i>swab()</i> , <i>tdelete()</i> , <i>tfind()</i> , <i>timezone()</i> ,	-
129617	<i>toascii()</i> , <i>tsearch()</i> , <i>twalk()</i>	
129618	XSI_DBM: XSI Database Management	
129619	<i>dbm_clearerr()</i> , <i>dbm_close()</i> , <i>dbm_delete()</i> , <i>dbm_error()</i> , <i>dbm_fetch()</i> , <i>dbm_firstkey()</i> ,	
129620	<i>dbm_nextkey()</i> , <i>dbm_open()</i> , <i>dbm_store()</i>	
129621	XSI_DEVICE_IO: XSI Device Input and Output	
129622	<i>fntmsg()</i> , <i>readv()</i> , <i>writv()</i>	
129623	XSI_DEVICE_SPECIFIC: XSI General Terminal	
129624	<i>grantpt()</i> , <i>posix_openpt()</i> , <i>ptsname()</i> , <i>unlockpt()</i>	
129625	XSI_FILE_SYSTEM: XSI File System	
129626	<i>basename()</i> , <i>dirname()</i> , <i>ftw()</i> , <i>lockf()</i> , <i>mknod()</i> , <i>nftw()</i> , <i>realpath()</i> , <i>seekdir()</i> , <i>sync()</i> , <i>telldir()</i> ,	
129627	<i>tempnam()</i>	
129628	XSI_IPC: XSI Interprocess Communication	
129629	<i>flok()</i> , <i>msgctl()</i> , <i>msgget()</i> , <i>msgrcv()</i> , <i>msgsnd()</i> , <i>semctl()</i> , <i>semget()</i> , <i>semop()</i> , <i>shmat()</i> , <i>shmctl()</i> ,	
129630	<i>shmdt()</i> , <i>shmget()</i>	
129631	XSI_JUMP: XSI Jump Functions	
129632	<i>_longjmp()</i> , <i>_setjmp()</i>	
129633	XSI_MATH: XSI Maths Library	
129634	<i>j0()</i> , <i>j1()</i> , <i>jn()</i> , <i>y0()</i> , <i>y1()</i> , <i>yn()</i>	-
129635	XSI_MULTI_PROCESS: XSI Multiple Process	
129636	<i>getpriority()</i> , <i>getrlimit()</i> , <i>getrusage()</i> , <i>nice()</i> , <i>setpgrp()</i> , <i>setpriority()</i> , <i>setrlimit()</i> , <i>ulimit()</i>	
129637	XSI_SIGNALS: XSI Signal	
129638	<i>killpg()</i> , <i>sigaltstack()</i> , <i>sighold()</i> , <i>sigignore()</i> , <i>siginterrupt()</i> , <i>sigpause()</i> , <i>sigrelse()</i> , <i>sigset()</i>	-
129639	XSI_SINGLE_PROCESS: XSI Single Process	
129640	<i>gethostid()</i> , <i>gettimeofday()</i> , <i>putenv()</i>	
129641	XSI_SYSTEM_DATABASE: XSI System Database	
129642	<i>endpwent()</i> , <i>getpwent()</i> , <i>setpwent()</i>	
129643	XSI_SYSTEM_LOGGING: XSI System Logging	
129644	<i>closelog()</i> , <i>openlog()</i> , <i>setlogmask()</i> , <i>syslog()</i>	
129645	XSI_THREADS_EXT: XSI Threads Extensions	
129646	<i>pthread_attr_getstack()</i> , <i>pthread_attr_setstack()</i> , <i>pthread_getconcurrency()</i> ,	
129647	<i>pthread_setconcurrency()</i>	
129648	XSI_TIMERS: XSI Timers	
129649	<i>getitimer()</i> , <i>setitimer()</i>	
129650	XSI_USER_GROUPS: XSI User and Group	
129651	<i>endgrent()</i> , <i>endutxent()</i> , <i>getgrent()</i> , <i>getutxent()</i> , <i>getutxid()</i> , <i>getutxline()</i> , <i>pututxline()</i> ,	
129652	<i>setgrent()</i> , <i>setregid()</i> , <i>setreuid()</i> , <i>setutxent()</i>	
129653	XSI_WIDE_CHAR: XSI Wide-Character Library	
129654	<i>wcswidth()</i> , <i>wcwidth()</i>	



# Index

(time) resolution .....	86
/ .....	197
/dev .....	197
/dev/console .....	197
/dev/null .....	197
/dev/tty .....	197, 3487
/etc/passwd .....	3502
/tmp .....	197
<aio.h> .....	220
<alert> .....	34
<apostrophe> .....	35
<arpa/inet.h> .....	222
<assert.h> .....	223
<backspace> .....	39
<blank> .....	44
<carriage-return> .....	46
<circumflex> .....	48
<complex.h> .....	224
<control>-V .....	2699
<control>-W .....	2699
<cpio.h> .....	227
<ctype.h> .....	229
<dirent.h> .....	231
<dlfcn.h> .....	233
<dollar-sign> .....	54
<errno.h> .....	234
<fcntl.h> .....	238
<fenv.h> .....	243
<float.h> .....	247
<fmtmsg.h> .....	252
<fnmatch.h> .....	254
<form-feed> .....	63
<ftw.h> .....	255
<glob.h> .....	257
<grp.h> .....	259
<iconv.h> .....	261
<inttypes.h> .....	262
<iso646.h> .....	265
<langinfo.h> .....	266
<libgen.h> .....	269
<limits.h> .....	270
<locale.h> .....	285
<math.h> .....	288
<monetary.h> .....	295
<mqueue.h> .....	296
<ndbm.h> .....	298
<net/if.h> .....	300

<netdb.h> .....	301
<netinet/in.h> .....	305
<netinet/tcp.h> .....	310
<newline> .....	72
<nl_types.h> .....	311
<number-sign> .....	73
<period> .....	77
<poll.h> .....	312
<pthread.h> .....	314, 3633
<pwd.h> .....	320
<regex.h> .....	322
<sched.h> .....	324
<search.h> .....	326
<semaphore.h> .....	328
<setjmp.h> .....	330
<signal.h> .....	331
<slash> .....	91
<space> .....	92
<spawn.h> .....	340
<stdarg.h> .....	342
<stdbool.h> .....	344
<stddef.h> .....	345
<stdint.h> .....	347
<stdio.h> .....	354
<stdlib.h> .....	358
<string.h> .....	362
<strings.h> .....	364
<stropts.h> .....	365
<sys/dir.h> .....	231
<sys/ipc.h> .....	370
<sys/mman.h> .....	372
<sys/msg.h> .....	375
<sys/resource.h> .....	377
<sys/select.h> .....	379
<sys/sem.h> .....	381
<sys/shm.h> .....	383
<sys/socket.h> .....	385
<sys/stat.h> .....	391
<sys/statvfs.h> .....	396
<sys/time.h> .....	398
<sys/times.h> .....	400
<sys/types.h> .....	401
<sys/uio.h> .....	405
<sys/un.h> .....	406
<sys/utsname.h> .....	407
<sys/wait.h> .....	408
<syslog.h> .....	410
<tab> .....	98
<tar.h> .....	412
<termios.h> .....	414
<tgmath.h> .....	420
<tilde> .....	99

<time.h> .....	424
<trace.h> .....	428
<ulimit.h> .....	432
<unistd.h> .....	433
<utime.h> .....	454
<utmpx.h> .....	455
<vertical-tab> .....	104
<wchar.h> .....	457
<wctype.h> .....	462
<wordexp.h> .....	464
±0 .....	106
_asm_builtin_atoi() .....	3556
_BSD .....	3677
_CFLAGS .....	2537
_Complex_I .....	224
_CS_PATH .....	440
_CS_POSIX_V6_ILP32_OFF32_CFLAGS .....	442
_CS_POSIX_V6_ILP32_OFF32_LDFLAGS .....	442
_CS_POSIX_V6_ILP32_OFF32_LIBS .....	442
_CS_POSIX_V6_ILP32_OFFBIG_CFLAGS .....	442
_CS_POSIX_V6_ILP32_OFFBIG_LDFLAGS .....	442
_CS_POSIX_V6_ILP32_OFFBIG_LIBS .....	442
_CS_POSIX_V6_LP64_OFF64_CFLAGS .....	442
_CS_POSIX_V6_LP64_OFF64_LDFLAGS .....	442
_CS_POSIX_V6_LP64_OFF64_LIBS .....	442
_CS_POSIX_V6_LPBIG_OFFBIG_CFLAGS .....	442
_CS_POSIX_V6_LPBIG_OFFBIG_LDFLAGS .....	442
_CS_POSIX_V6_LPBIG_OFFBIG_LIBS .....	442
_CS_POSIX_V6_WIDTH_RESTRICTED_ENVS .....	442
_CS_POSIX_V7_ILP32_OFF32_CFLAGS .....	440
_CS_POSIX_V7_ILP32_OFF32_LDFLAGS .....	441
_CS_POSIX_V7_ILP32_OFF32_LIBS .....	441
_CS_POSIX_V7_ILP32_OFFBIG_CFLAGS .....	441
_CS_POSIX_V7_ILP32_OFFBIG_LDFLAGS .....	441
_CS_POSIX_V7_ILP32_OFFBIG_LIBS .....	441
_CS_POSIX_V7_LP64_OFF64_CFLAGS .....	441
_CS_POSIX_V7_LP64_OFF64_LDFLAGS .....	441
_CS_POSIX_V7_LP64_OFF64_LIBS .....	441
_CS_POSIX_V7_LPBIG_OFFBIG_CFLAGS .....	441
_CS_POSIX_V7_LPBIG_OFFBIG_LDFLAGS .....	441
_CS_POSIX_V7_LPBIG_OFFBIG_LIBS .....	442
_CS_POSIX_V7_THREADS_CFLAGS .....	442
_CS_POSIX_V7_WIDTH_RESTRICTED_ENVS .....	442
_CS_V6_ENV .....	442
_CS_V7_ENV .....	442
_Exit() .....	553
_exit() .....	553, 2223, 3572, 3591
_Exit() .....	3754
_exit() .....	3754
_FILE .....	607
_Imaginary_I .....	224
_IOFBF .....	354, 1880, 1922

_IOLBF .....	354, 872, 1922
_IONBF .....	354, 1880, 1922
_LDFLAGS .....	2537
_LIBS .....	2537
_LINE_ .....	607
_longjmp() .....	558, 3757
_LVL .....	476
_MAX .....	475
_MIN .....	270, 475
_PC constants	
defined in <unistd.h> .....	443
used in pathconf .....	899
_PC_2_SYMLINKS .....	899
_PC_ALLOC_SIZE_MIN .....	899
_PC_ASYNC_IO .....	899
_PC_CHOWN_RESTRICTED .....	899
_PC_FILESIZEBITS .....	899
_PC_LINK_MAX .....	899
_PC_MAX_CANON .....	899
_PC_MAX_INPUT .....	899
_PC_NAME_MAX .....	899
_PC_NO_TRUNC .....	899
_PC_PATH_MAX .....	899
_PC_PIPE_BUF .....	899
_PC_PRIO_IO .....	899
_PC_REC_INCR_XFER_SIZE .....	899
_PC_REC_MAX_XFER_SIZE .....	899
_PC_REC_MIN_XFER_SIZE .....	899
_PC_REC_XFER_ALIGN .....	899
_PC_SYMLINK_MAX .....	899
_PC_SYNC_IO .....	899
_PC_TIMESTAMP_RESOLUTION .....	899
_PC_VDISABLE .....	899
_POSIX .....	270
_POSIX maximum values	
in <limits.h> .....	275
_POSIX minimum values	
in <limits.h> .....	276
_POSIX2 constants	
in sysconf .....	2089
_POSIX2_BC_BASE_MAX .....	274, 279
_POSIX2_BC_DIM_MAX .....	275, 279
_POSIX2_BC_SCALE_MAX .....	275, 279
_POSIX2_BC_STRING_MAX .....	275, 279
_POSIX2_CHARCLASS_NAME_MAX .....	275, 279
_POSIX2_CHAR_TERM .....	438, 2092
_POSIX2_COLL_WEIGHTS_MAX .....	275, 279
_POSIX2_C_BIND .....	17, 438, 2092
_POSIX2_C_DEV .....	438, 2092
_POSIX2_EXPR_NEST_MAX .....	275, 279
_POSIX2_FORT_DEV .....	438, 2092
_POSIX2_FORT_RUN .....	438, 2092

_POSIX2_LINE_MAX.....	275, 279, 282
_POSIX2_LOCALEDEF.....	438, 2092
_POSIX2_PBS.....	438, 2092
_POSIX2_PBS_ACCOUNTING.....	438, 2092
_POSIX2_PBS_CHECKPOINT.....	439, 2092
_POSIX2_PBS_LOCATE.....	439, 2092
_POSIX2_PBS_MESSAGE.....	439, 2092
_POSIX2_PBS_TRACK.....	439, 2092
_POSIX2_RE_DUP_MAX.....	279
_POSIX2_SW_DEV.....	439, 2092
_POSIX2_SYMLINKS.....	440
_POSIX2_UPE.....	439, 2092
_POSIX2_VERSION.....	433, 2092
_POSIX.....	474
_POSIX_ADVISORY_INFO.....	18, 23, 433, 901, 2090, 3763
_POSIX_AIO_LISTIO_MAX.....	270, 276
_POSIX_AIO_MAX.....	271, 276
_POSIX_ARG_MAX.....	271, 276
_POSIX_ASYNCIO.....	17, 434, 2090, 3475, 3763
_POSIX_ASYNC_IO.....	440, 899
_POSIX_BARRIERS.....	17, 434, 2090, 3475, 3764
_POSIX_CHILD_MAX.....	271, 276
_POSIX_CHOWN_RESTRICTED.....	17, 434, 669, 899, 902, 3469, 3764
_POSIX_CLOCKRES_MIN.....	275
_POSIX_CLOCK_SELECTION.....	17, 434, 2090, 3475, 3764
_POSIX_CPUTIME.....	18, 23, 434, 2090, 3764
_POSIX_C_SOURCE.....	472-473, 3557, 3560
_POSIX_DELAYTIMER_MAX.....	271, 276
_POSIX_FSYNC.....	18, 20, 23, 434, 2090, 3764
_POSIX_HOST_NAME_MAX.....	271, 276
_POSIX_IPV6.....	18, 434, 2090, 3764
_POSIX_JOB_CONTROL.....	17, 434, 2090, 3469, 3764, 3768
_POSIX_LINK_MAX.....	273, 276
_POSIX_LOGIN_NAME_MAX.....	271, 276
_POSIX_MAPPED_FILES.....	17, 434, 2090, 3475, 3764
_POSIX_MAX_CANON.....	273, 276
_POSIX_MAX_INPUT.....	274, 276
_POSIX_MEMLOCK.....	18, 23, 434, 2090, 3765
_POSIX_MEMLOCK_RANGE.....	18, 23, 434, 2090, 3765
_POSIX_MEMORY_PROTECTION.....	17, 434, 2090, 3475, 3765
_POSIX_MESSAGE_PASSING.....	18, 23, 434, 2090, 3765
_POSIX_MONOTONIC_CLOCK.....	18, 23, 435, 2090, 3765
_POSIX_MQ_OPEN_MAX.....	271, 276
_POSIX_MQ_PRIO_MAX.....	271, 276
_POSIX_NAME_MAX.....	274, 277, 1351, 1361, 1538, 1846, 1856, 1929
_POSIX_NGROUPS_MAX.....	275, 277
_POSIX_NO_TRUNC.....	17, 112, 435, 899, 3469
_POSIX_OPEN_MAX.....	271, 277, 1082
_POSIX_PATH_MAX.....	274, 277, 406, 1351, 1361, 1846, 1856, 1929
_POSIX_PIPE_BUF.....	274, 277
_POSIX_PRIORITYIO.....	18, 23, 435, 504, 2090, 3765
_POSIX_PRIORITY_SCHEDULING.....	18, 23-24, 435, 504, 2090, 3765

_POSIX_PRIO_IO.....	440, 899
_POSIX_RAW_SOCKETS .....	18, 435, 2090
_POSIX_READER_WRITER_LOCKS .....	17, 435, 2090, 3475
_POSIX_REALTIME_SIGNALS.....	17, 435, 2090, 3475, 3765
_POSIX_REGEX.....	17, 435, 2090, 3765
_POSIX_RE_DUP_MAX.....	275, 277
_POSIX_RTSIG_MAX.....	272, 277, 3567, 3770
_POSIX_SAVED_IDS .....	17, 435, 2090, 3469, 3766
_POSIX_SEMAPHORES .....	17, 435, 2090, 3475, 3766
_POSIX_SEM_NSEMS_MAX .....	272, 277
_POSIX_SEM_VALUE_MAX .....	272, 277
_POSIX_SHARED_MEMORY_OBJECTS .....	18, 23, 435, 2090, 3766
_POSIX_SHELL .....	17, 435, 2090, 3766
_POSIX_SIGQUEUE_MAX.....	272, 277
_POSIX_SOURCE.....	473, 3557
_POSIX_SPAWN.....	18, 23, 435, 2090, 3766
_POSIX_SPINLOCKS .....	3766
_POSIX_SPIN_LOCKS .....	17, 436, 2090, 3475
_POSIX_SPORADIC_SERVER.....	18, 23-24, 436, 2090, 3766
_POSIX_SSIZE_MAX.....	277, 281
_POSIX_SS_REPL_MAX .....	272, 277, 2090, 3606
_POSIX_STREAM_MAX.....	272, 277
_POSIX_SYMLINK_MAX.....	274, 278
_POSIX_SYMLOOP_MAX.....	272, 278
_POSIX_SYNCHRONIZED_IO .....	18, 23, 436, 2090, 3766
_POSIX_SYNC_IO .....	440, 899, 3764
_POSIX_THREADS.....	17, 437, 2090, 3475, 3766
_POSIX_THREAD_ATTR_STACKADDR .....	18, 20, 436, 2090, 3766
_POSIX_THREAD_ATTR_STACKSIZE.....	18, 20, 436, 2090, 3767
_POSIX_THREAD_CPUTIME .....	18, 24-25, 436, 2090
_POSIX_THREAD_DESTRUCTOR_ITERATIONS.....	272, 278
_POSIX_THREAD_KEYS_MAX .....	272, 278
_POSIX_THREAD_PRIORITY_SCHEDULING.....	18, 24-25, 436, 2090, 3767
_POSIX_THREAD_PRIO_INHERIT .....	18, 24, 436, 2090, 3767
_POSIX_THREAD_PRIO_PROTECT.....	18, 24, 436, 2090, 3767
_POSIX_THREAD_PROCESS_SHARED .....	18, 20, 436, 1673, 2090, 3767
_POSIX_THREAD_ROBUST_PRIO_INHERIT .....	24, 436, 2090
_POSIX_THREAD_ROBUST_PRIO_PROTECT .....	24, 436, 2090
_POSIX_THREAD_SAFE_FUNCTIONS .....	17, 437, 2090, 3475, 3767
_POSIX_THREAD_SPORADIC_SERVER.....	18, 24-25, 437, 2090, 3767
_POSIX_THREAD_THREADS_MAX .....	272, 278
_POSIX_TIMEOUTS.....	17, 437, 2090, 3475, 3767
_POSIX_TIMERS .....	17, 437, 2091, 3475, 3767
_POSIX_TIMER_MAX.....	272, 278
_POSIX_TIMESTAMP_RESOLUTION .....	440, 899
_POSIX_TRACE .....	18, 25, 437, 2091, 3767
_POSIX_TRACE_EVENT_FILTER.....	18, 25, 437, 2091, 3768
_POSIX_TRACE_EVENT_NAME_MAX.....	273, 278, 1509, 1511, 2091
_POSIX_TRACE_INHERIT .....	18, 25, 437, 2091, 3768
_POSIX_TRACE_LOG.....	18, 25, 437, 2091, 3768
_POSIX_TRACE_NAME_MAX .....	273, 278, 2091
_POSIX_TRACE_SYS_MAX .....	273, 278, 1506, 2091

_POSIX_TRACE_USER_EVENT_MAX .....	273, 278, 1511, 2091
_POSIX_TTY_NAME_MAX .....	273, 278
_POSIX_TYPED_MEMORY_OBJECTS .....	18, 23, 437, 2091, 3768
_POSIX_TZNAME_MAX .....	273, 278, 3528
_POSIX_V6_ILP32_OFF32 .....	437, 2092
_POSIX_V6_ILP32_OFFBIG .....	437, 2092
_POSIX_V6_LP64_OFF64 .....	437, 2092
_POSIX_V6_LPBIG_OFFBIG .....	438, 2092
_POSIX_V7_ILP32_OFF32 .....	438, 2091
_POSIX_V7_ILP32_OFFBIG .....	438, 2091
_POSIX_V7_LP64_OFF64 .....	438, 2091
_POSIX_V7_LPBIG_OFFBIG .....	438, 2091
_POSIX_VDISABLE .....	18, 446, 899, 3260, 3469
_POSIX_VERSION .....	17, 433, 2091, 2182
_PROCESS .....	476
_PTHREAD_THREADS_MAX .....	1641
_SC constants	
defined in <unistd.h> .....	<b>443</b>
in sysconf .....	2089
_SC_2_CHAR_TERM .....	2092
_SC_2_C_BIND .....	2092
_SC_2_C_DEV .....	2092
_SC_2_FORT_DEV .....	2092
_SC_2_FORT_RUN .....	2092
_SC_2_LOCALEDEF .....	2092
_SC_2_PBS_ACCOUNTING .....	2092
_SC_2_PBS_CHECKPOINT .....	2092
_SC_2_PBS_LOCATE .....	2092
_SC_2_PBS_MESSAGE .....	2092
_SC_2_PBS_TRACK .....	2092
_SC_2_SW_DEV .....	2092
_SC_2_UPE .....	2092
_SC_2_VERSION .....	1430, 2092
_SC_ADVISORY_INFO .....	2090
_SC_AIO_LISTIO_MAX .....	2089
_SC_AIO_MAX .....	2089
_SC_AIO_PRIO_DELTA_MAX .....	2089
_SC_ARG_MAX .....	2089
_SC_ASYNCHRONOUS_IO .....	2090
_SC_ATEXIT_MAX .....	2089
_SC_BARRIERS .....	2090
_SC_BC_BASE_MAX .....	2089
_SC_BC_DIM_MAX .....	2089
_SC_BC_SCALE_MAX .....	2089
_SC_BC_STRING_MAX .....	2089
_SC_CHILD_MAX .....	2089
_SC_CLK_TCK .....	2089, 2150
_SC_CLOCK_SELECTION .....	2090
_SC_COLL_WEIGHTS_MAX .....	2089
_SC_CPUTIME .....	2090
_SC_DELAYTIMER_MAX .....	2089
_SC_EXPR_NEST_MAX .....	2089

_SC_FSYNC .....	2090
_SC_GETGR_R_SIZE_MAX .....	1030, 2089
_SC_GETPW_R_SIZE_MAX .....	2089
_SC_IOV_MAX .....	2089
_SC_IPV6 .....	2090
_SC_JOB_CONTROL .....	2090
_SC_LINE_MAX .....	2089
_SC_LOGIN_NAME_MAX .....	2089
_SC_MEMLOCK .....	2090
_SC_MEMLOCK_RANGE .....	2090
_SC_MEMORY_PROTECTION .....	2090
_SC_MESSAGE_PASSING .....	2090
_SC_MONOTONIC_CLOCK .....	2090
_SC_MQ_OPEN_MAX .....	2089
_SC_MQ_PRIO_MAX .....	2089
_SC_NGROUPS_MAX .....	2089
_SC_OPEN_MAX .....	2089
_SC_PAGESIZE .....	1436, 2089, 3590, 3592
_SC_PAGE_SIZE .....	2089
_SC_PRIORITIZED_IO .....	2090
_SC_PRIORITY_SCHEDULING .....	2090
_SC_RAW_SOCKETS .....	2090
_SC_READER_WRITER_LOCKS .....	2090
_SC_REALTIME_SIGNALS .....	2090
_SC_REGEX .....	2090
_SC_RE_DUP_MAX .....	2089
_SC_RTSIG_MAX .....	2089
_SC_SAVED_IDS .....	2090
_SC_SEMAPHORES .....	2090
_SC_SEM_NSEMS_MAX .....	2090
_SC_SEM_VALUE_MAX .....	2090
_SC_SHARED_MEMORY_OBJECTS .....	2090
_SC_SHELL .....	2090
_SC_SIGQUEUE_MAX .....	2090
_SC_SPAWN .....	2090
_SC_SPIN_LOCKS .....	2090
_SC_SPORADIC_SERVER .....	2090
_SC_SS_REPL_MAX .....	2090
_SC_STREAM_MAX .....	2090
_SC_SYMLOOP_MAX .....	2090
_SC_SYNCHRONIZED_IO .....	2090
_SC_THREADS .....	2090
_SC_THREAD_ATTR_STACKADDR .....	2090
_SC_THREAD_ATTR_STACKSIZE .....	2090
_SC_THREAD_CPUTIME .....	2090
_SC_THREAD_DESTRUCTOR_ITERATIONS .....	2089
_SC_THREAD_KEYS_MAX .....	2089
_SC_THREAD_PRIORITY_SCHEDULING .....	2090
_SC_THREAD_PRIO_INHERIT .....	2090
_SC_THREAD_PRIO_PROTECT .....	2090
_SC_THREAD_PROCESS_SHARED .....	2090
_SC_THREAD_ROBUST_PRIO_INHERIT .....	2090



_SC_THREAD_ROBUST_PRIO_PROTECT.....	2090
_SC_THREAD_SAFE_FUNCTIONS.....	2090
_SC_THREAD_SPORADIC_SERVER.....	2090
_SC_THREAD_STACK_MIN.....	2089
_SC_THREAD_THREADS_MAX.....	2089
_SC_TIMEOUTS.....	2090
_SC_TIMERS.....	2091
_SC_TIMER_MAX.....	2090
_SC_TRACE.....	2091
_SC_TRACE_EVENT_FILTER.....	2091
_SC_TRACE_EVENT_NAME_MAX.....	2091
_SC_TRACE_INHERIT.....	2091
_SC_TRACE_LOG.....	2091
_SC_TRACE_NAME_MAX.....	2091
_SC_TRACE_SYS_MAX.....	2091
_SC_TRACE_USER_EVENT_MAX.....	2091
_SC_TTY_NAME_MAX.....	2090
_SC_TYPED_MEMORY_OBJECTS.....	2091
_SC_TZNAME_MAX.....	2090
_SC_V6_ILP32_OFF32.....	2092
_SC_V6_ILP32_OFFBIG.....	2092
_SC_V6_LP64_OFF64.....	2092
_SC_V6_LPBIG_OFFBIG.....	2092
_SC_V7_ILP32_OFF32.....	2091
_SC_V7_ILP32_OFFBIG.....	2091
_SC_V7_LP64_OFF64.....	2091
_SC_V7_LPBIG_OFFBIG.....	2091
_SC_VERSION.....	2091
_SC_XOPEN_CRYPT.....	2092
_SC_XOPEN_ENH_I18N.....	2092
_SC_XOPEN_REALTIME.....	2092
_SC_XOPEN_REALTIME_THREADS.....	2092
_SC_XOPEN_SHM.....	2092
_SC_XOPEN_STREAMS.....	2092
_SC_XOPEN_UNIX.....	2092
_SC_XOPEN_UUCP.....	2092
_SC_XOPEN_VERSION.....	2092
_setjmp().....	558, 3757
_t.....	476
_TIME.....	476
_tolower().....	560
_toupper().....	561
_XOPEN_CRYPT.....	18, 22, 439, 2092
_XOPEN_ENH_I18N.....	439, 2092
_XOPEN_IOV_MAX.....	271, 279
_XOPEN_NAME_MAX.....	274, 279, 1351, 1361, 1539, 1846, 1856, 1929
_XOPEN_PATH_MAX.....	274, 279, 1351, 1361, 1538, 1846, 1856, 1929
_XOPEN_REALTIME.....	18, 22-23, 439, 827, 2092
_XOPEN_REALTIME_THREADS.....	18, 24, 439, 2092
_XOPEN_SHM.....	439, 2092
_XOPEN_SOURCE.....	473, 3557
_XOPEN_STREAMS.....	18, 25-26, 439, 2092

_XOPEN_UNIX.....	18-19, 439, 2092
_XOPEN_UUCP .....	439, 2092
_XOPEN_VERSION.....	19, 433, 2092
__errno().....	3564
a64l() .....	<b>562</b>
ABDAY_.....	267
ABDAY_1.....	1396
ABMON_.....	267
abort() .....	<b>564</b> , 3754
abortive release .....	33
abs() .....	<b>566</b>
absolute pathname .....	33, 111
accept() .....	<b>567</b>
access mode.....	33
access() .....	<b>569</b> , 3501, 3754
Account_Name .....	2425
acos().....	<b>573</b>
acosf() .....	573
acosh() .....	<b>575</b>
acoshf() .....	575
acoshl() .....	575
acosl() .....	573, 577
ACTION.....	1110
actions equivalent to functions .....	2321
active trace stream.....	3676
adb	
rationale for omission.....	3743
additional file access control mechanism .....	33
address families.....	3650
address information.....	930
address space .....	33
address string.....	930
addressing .....	3650
addrinfo .....	<b>302</b> , 930
admin .....	<b>2444</b>
ADV .....	<b>7</b>
advanced realtime .....	23
ADVANCED REALTIME.....	340, 676, 1432, 1434, 1436, 1438, 1440, 1444, 1452, 1455, 1458, 1460
.....	1462, 1464, 1466, 1468, 1470, 1472, 1535, 1537
advanced realtime threads.....	24
ADVANCED REALTIME THREADS.....	1635
advisory information.....	34, 3577
affirmative response .....	34
AF_.....	476
AF_INET.....	387
AF_INET6.....	387
AF_UNIX.....	387
AF_UNSPEC .....	388, 700
AIO_.....	475
aio_.....	475
AIO_ALLDONE .....	220, 578
aio_cancel().....	<b>578</b> , 3585-3586

AIO_CANCELED .....	220, 578
aio_error() .....	<b>580</b>
aio_fsync() .....	<b>582</b> , 3569, 3584
AIO_LISTIO_MAX .....	270, 1241, 2089, 3768
AIO_MAX .....	271, 1241, 2089, 3768
AIO_NOTCANCELED .....	220, 578
AIO_PRIO_DELTA_MAX .....	271, 504, 2089, 3768
aio_read() .....	<b>584</b> , 3586
aio_return() .....	<b>587</b>
aio_suspend() .....	<b>589</b> , 3585, 3611
aio_write() .....	<b>591</b> , 3586
ai_ .....	475
AI_ADDRCONFIG .....	302, 931
AI_ALL .....	302, 931
AI_CANONNAME .....	302, 931
AI_INET6 .....	931
AI_NUMERICHOST .....	302, 931
AI_NUMERICSERV .....	302, 931
AI_PASSIVE .....	302, 931
AI_V4MAPPED .....	302, 931
alarm() .....	<b>594</b> , 3574, 3610, 3754
alert .....	34
alert character .....	34
alias .....	2334, 2357, 2449, 3705, 3758
alias name .....	34
alias substitution .....	2338, 3708
alignment .....	35
alphasort() .....	<b>596</b>
alternate file access control mechanism .....	35
alternate signal stack .....	35
ALT_DIGITS .....	267
AM_STR .....	267
anchoring .....	188
ancillary data .....	35
AND list .....	2360, 3730
AND-OR list .....	2359
angle brackets .....	35
anycast .....	533
ANYMARK .....	368, 1146
API .....	36
apostrophe character .....	35
appending redirected output .....	2351
application .....	35
application address .....	36
application conformance .....	29
application instrumentation .....	3664
application program interface .....	36
application-managed thread stack .....	522, 3650
appropriate privileges .....	36, 571, 900, 3478
ar. ....	<b>2452</b> , 3760-3761
arbitrary file size .....	3704

archives	
ar command .....	2452
AREGTYPE .....	412
argc .....	789
argument .....	36
ARG_MAX .....	271, 481, 782, 786, 791, 2089, 3441, 3488, 3698, 3768
arithmetic expansion .....	2348, 3719
arithmetic language	
bc.....	2514
arithmetic precision and operations.....	2321
arm (a timer) .....	36
array identifiers .....	2519
as	
rationale for omission.....	3743
asa .....	<b>2460</b> , 3758, 3760, 3762
ASCII .....	3490
asctime() .....	<b>599</b>
asctime_r() .....	599
asin() .....	<b>602</b>
asinf() .....	602
asinh() .....	<b>604</b>
asinhf() .....	604
asinhf() .....	604
asinhf() .....	604
asinhf() .....	602, 606
assert() .....	<b>607</b>
asterisk .....	36
async-cancel safety.....	3648
async-cancel-safe function .....	37
async-signal-safe .....	1552, 3572
async-signal-safe function.....	37
asynchronous error .....	3652
asynchronous events .....	37
asynchronous I/O .....	3584, 3755
completion.....	37
operation.....	37
asynchronous input and output.....	37
asynchronous lists.....	2360, 3729
asynchronously-generated signal .....	37
at.....	<b>2463</b> , 3758
at-job.....	2463
atan() .....	<b>608</b>
atan2() .....	<b>610</b>
atan2f() .....	610
atan2l() .....	610
atanf() .....	608, 613
atanh() .....	<b>614</b>
atanhf() .....	614
atanhl() .....	614
atanl() .....	608, 616
atexit() .....	<b>617</b> , 3647
ATEXIT_MAX.....	271, 617, 2089
atof() .....	<b>619</b>

atoi()	620, 3555-3556
atol()	622
atoll()	622
attributes	
clock-resolution	542, 1484
creation-time	542, 1484
generation-version	542, 1484
inheritance	542, 1486
log-full-policy	540, 542, 1486, 1489
log-max-size	542, 1487, 1489
max-data-size	542, 1489
stream-full-policy	539-540, 542, 1487
stream-min-size	542, 1490
trace-name	542, 1484
truncation-status	1509
AT_EACCESS	239
AT_FDCWD	239, 569, 664, 670, 962, 983, 1235, 1308, 1315, 1320, 1403, 1775, 1809, 2085
AT_REMOVEDIR	3290
AT_SYMLINK_FOLLOW	240, 2188
AT_SYMLINK_NOFOLLOW	240, 1235
authentication	240, 664, 670, 962, 984
authorization	38
automatic storage class	38
awk	2523
actions	2472, 3758, 3760
arithmetic functions	2484
escape sequences	2486
expression patterns	2482
expressions	2484
functions	2475
grammar	2486
input/output and general functions	2490
lexical conventions	2488
output statements	2496
overall program structure	2485
pattern ranges	2475
patterns	2484
regular expressions	2483
special patterns	2481
string functions	2483
user-defined functions	2487
variables and special variables	2489
background	2479
background job	1900, 3484-3487, 3539-3540
background process	38
background process group	38, 2123
background work	38
at	2463
batch	2511
bg	2529
crontab	2602

fg .....	2775
jobs .....	2859
nice .....	3016
nohup .....	3029
renice .....	3183
backquote .....	38
BACKREF .....	192
backslash .....	38, 3706
backspace character .....	39
bandinfo .....	<b>365</b>
banner	
rationale for omission .....	3743
barrier .....	39, 3626
basename .....	39, 2508, 3757, 3759
basename() .....	<b>623</b>
basic regular expression .....	39, 183, 3530
batch .....	<b>2511</b> , 3758
general concepts .....	3739
batch access list .....	39
batch administration .....	2420
batch administrator .....	39
batch authorization .....	2420
batch client .....	40
batch client-server interaction .....	2417
batch destination .....	40
batch destination identifier .....	40
batch directive .....	40
batch environment .....	3736
option definitions .....	3737
services .....	2417
utilities .....	2417
utilities, common behavior .....	3742
batch job .....	40
batch job abort .....	2420, 2431
batch job attribute .....	41
batch job creation .....	2418
batch job execution .....	2419, 2423
batch job exit .....	2420, 2430
batch job identifier .....	41, 2439
batch job message request .....	2433
batch job name .....	41
batch job owner .....	41
batch job priority .....	41
batch job routing .....	2419, 2430
batch job state .....	41
batch job states .....	2422
batch job status request .....	2434
batch job tracking .....	2418
batch name service .....	41
batch name space .....	42
batch node .....	42
batch notification .....	2421

batch operator .....	42
batch queue .....	42, 2418
batch queue attribute .....	42
batch queue position .....	42
batch queue priority .....	43
batch queue status request .....	2436
batch rerunability .....	43
batch restart .....	43
batch server .....	43
batch server name .....	43
batch server restart .....	2431
batch service .....	43
batch service request .....	43
batch services .....	2421, 3741
batch submission .....	44
batch system .....	44
historical implementations .....	3737
history .....	3736
batch target user .....	44
batch user .....	44
baud rate functions .....	657
bc .....	2514, 3757-3758, 3763
grammar .....	2515
lexical conventions .....	2517
operations .....	2519
operators .....	2519
bcc (mailer blind carbon copy) .....	2955
bcmp() .....	3679
bcopy() .....	3679
BC_ constants .....	
in sysconf .....	2089
BC_BASE_MAX .....	274, 2089, 2324, 3698
BC_DIM_MAX .....	275, 2089, 2324, 3698
BC_SCALE_MAX .....	275, 2089, 2324, 3698
BC_STRING_MAX .....	275, 2089, 2324, 2517
BE .....	7
bg .....	2334, 2357, 2529, 3705, 3758
binary primaries .....	3279
bind .....	44
bind() .....	625
bi_ .....	475
blank character .....	44
blank line .....	44
blkcnt_t .....	401
blksize_t .....	401
BLKTYPE .....	412
block special file .....	45
block-mode terminal .....	45
blocked process (or thread) .....	45
blocking .....	45
BOOT_TIME .....	455, 769-770
bounded response .....	3756

braces .....	45
bracket expression	
grammar .....	3535
brackets .....	46
BRE	
expression anchoring .....	3533
grammar lexical conventions .....	3535
matching a collating element .....	3530
matching a single character .....	3530
matching multiple characters .....	3532
ordinary character .....	3530
periods .....	3530
precedence .....	3533
special character .....	3530
BRE (ERE) matching a single character .....	182
BRE (ERE) matching multiple characters .....	182
break .....	<b>2376</b>
BRKINT .....	415
broadcast .....	46
BSD .....	3481, 3538, 3565
BSDLY .....	416
bsd_signal() .....	3680
bsearch() .....	<b>628</b>
BSn .....	416
btowc() .....	<b>631</b>
buffer cache .....	970
BUFSIZ .....	354, 1880
built-in .....	46
built-in utilities .....	46, 2334, 3704
builtin .....	2589
BUS_ .....	475
BUS_ADRALN .....	336
BUS_ADRERR .....	336
BUS_OBJERR .....	336
byte .....	46
byte input/output functions .....	46
byte-oriented stream .....	498
byte-stream mode .....	1764
bzero() .....	3680
C Shell .....	3484, 3486
C-language extensions .....	3751, 3757
c99 .....	<b>2532</b> , 3760
external symbols .....	2536
standard libraries .....	2535
cabs() .....	<b>632</b>
cabsf() .....	632
cabsl() .....	632
cacos() .....	<b>633</b>
cacosf() .....	633
cacosh() .....	<b>634</b>
cacoshf() .....	634
cacoshl() .....	634



cacosl()	633, 635
cal	<b>2543</b>
calendar	
rationale for omission	3743
calloc()	<b>636</b>
can	5
cancel	
rationale for omission	3743
cancel-safe	1720
cancelability state	517, 1642, 1720
cancelability type	1642, 1720
canceled execution of a thread	1595
cancellation cleanup handler	1600, 1612, 1631, 1645, 3646-3647
cancellation cleanup stack	3646
cancellation points	518
canonical mode input processing	202, 3540
canonical name	931
carg()	<b>638</b>
cargf()	638
cargl()	638
carriage-control characters	2460
carriage-return character	46
case	3731
case conditional construct	2362
case folding	3502-3503
casin()	<b>639</b>
casinf()	639
casinh()	<b>640</b>
casinhf()	640
casinhl()	640
casinl()	639, 641
cat	<b>2546</b> , 3704
catan()	<b>642</b>
catanf()	642
catanh()	<b>643</b>
catanhf()	643
catanhl()	643
catanl()	642, 644
catclose()	<b>645</b> , 3759
catgets()	<b>646</b> , 3759
catopen()	<b>648</b> , 3759
CBAUD	477
cbrt()	<b>650</b>
cbrtf()	650
cbrtl()	650
cc (mailer carbon copy)	2955
ccos()	<b>651</b>
ccosf()	651
ccosh()	<b>652</b>
ccoshf()	652
ccoshl()	652
ccosl()	651, 653

CD.....	7
cd .....	2334, 2357, 2550, 3705, 3759
ceil() .....	<b>654</b>
ceilf() .....	654
ceill() .....	654
CEO .....	3531
cexp() .....	<b>656</b>
cexpf() .....	656
cexpl() .....	656
cfgetispeed() .....	<b>657</b>
cfgetospeed() .....	<b>659</b>
cflow .....	<b>2555</b>
cfsetispeed() .....	<b>660</b>
cfsetospeed() .....	<b>661</b>
change current working directory .....	663, 2321
change file modes .....	667
change history .....	3470, 3551, 3695
change owner and group of file .....	671
char .....	548, 3679
character .....	47, 3479
rationale .....	3479
character array .....	47
character class .....	47
character counting .....	3423
character encoding .....	128, 3511
state-dependent .....	133
character set .....	47, 3510
description file .....	3511
portable filename .....	3490
character special file .....	48
character string .....	48
CHARCLASS_NAME_MAX .....	275, 3516
charmap .....	
description .....	129
with localedef .....	2898
writing names with locale .....	2892
charmap file .....	2896, 3260
CHAR_BIT .....	280
CHAR_MAX .....	280, 1252, 1254, 3518
CHAR_MIN .....	280
chdir() .....	<b>662</b>
Checkpoint .....	2425
chgrp .....	<b>2558</b> , 3704, 3759-3760
child process .....	48, 3479
CHILD_MAX .....	271, 896, 2089, 3469, 3678, 3698, 3729, 3768
chmod .....	<b>2561</b> , 3704, 3759-3760
grammar .....	2564
chmod() .....	<b>664</b> , 3754
chown .....	<b>2568</b> , 3704, 3759-3760
chown() .....	<b>669</b> , 3754
chroot .....	
rationale for omission .....	3743

chroot()	3490
CHRTYPE	412
cimag()	<b>673</b>
cimagf()	673
cimagl()	673
circumflex	48
cksum	2572, 3704, 3759
CLD_	475
CLD_CONTINUED	336
CLD_DUMPED	336
CLD_EXITED	336
CLD_KILLED	336
CLD_STOPPED	336
CLD_TRAPPED	336
clearerr()	<b>674</b>
CLOCAL	417
clock	48, 3606
clock jump	48
clock tick	48, 594, 2093, 2150, 3479
per second	2089
rationale	3479
clock()	<b>675</b>
clock-resolution attribute	542, 1484
clockid_t	401
CLOCKRES_MIN	3768
clocks	3606
CLOCKS_PER_SEC	401, 424, 675
CLOCK_	476
clock_	476
clock_getcpuclockid()	<b>676</b> , 3613-3614
clock_getres()	<b>677</b>
clock_gettime()	677
CLOCK_MONOTONIC	425, 511, 682
clock_nanosleep()	<b>681</b> , 3611
CLOCK_PROCESS_CPUTIME_ID	425, 512, 3613-3614
CLOCK_REALTIME	275, 425, 511, 677, 682, 1382, 1668, 2142, 3606-3611
clock_settime()	677, 684
clock_t	401
CLOCK_THREAD_CPUTIME_ID	425, 512, 3613-3614
clog()	<b>685</b>
clogf()	685
clogl()	685
close a file	688
close()	<b>686</b> , 3591, 3754
closedir()	<b>690</b> , 3755
closelog()	<b>692</b> , 3760
cmp	<b>2577</b> , 3704, 3758
msg_	476
MSG_	477
MSG_DATA	386
MSG_FIRSTHDR	386
MSG_NXTHDR	386

coded character set.....	49
codes.....	3472
codeset .....	49
CODESET .....	267
codeset conversion.....	2842
tr.....	3299
col	
rationale for omission.....	3743
collating element .....	49
collating element order.....	3531
collation .....	49
collation sequence .....	49
COLL_ELEM_MULTI.....	192
COLL_ELEM_SINGLE.....	192
COLL_WEIGHTS_MAX .....	275, 2089, 2324, 3698
colon.....	<b>2379</b>
column position.....	50, 3480
COLUMNS.....	<b>177</b> , 3527
comm.....	<b>2581</b> , 3758
command.....	50, 2334, 2357, 2585, 3479, 3705, 3757
command execution.....	3727
command interpreter	
portable.....	2223
command language.....	3751, 3757
command language interpreter.....	50
command mode.....	2666
command search.....	2357, 3727
command substitution.....	2347, 3718
communications commands	
mailx.....	2932
talk .....	3270
uucp.....	3342
uudecode .....	3346
uuencode .....	3349
uustat .....	3354
uux.....	3357
write .....	3433
compare thread IDs.....	1630
compilation environment.....	<b>472</b> , 3556
compilers	
c99.....	2532
fort77 .....	2799
yacc.....	3443
complex.....	224
complex data manipulation.....	3752, 3758
composite graphic symbol .....	50
compound commands.....	2361, 3730
compound-list.....	2359
compress.....	<b>2591</b>
compression	
compress.....	2591
uncompress .....	3326

zcat .....	3460
concepts .....	3472
concurrent execution .....	107, 3500
of processes .....	2317
condition variable .....	50
conditional construct	
case .....	3731
if .....	3731
configurable limits .....	3763, 3768
configuration interrogation .....	3750, 3753
configuration options .....	3761
shell and utilities .....	3761
system interfaces .....	3763
configuration values .....	2820
conformance .....	15, 29, 3470, 3473-3474, 3477-3478, 3503, 3677
POSIX .....	15
POSIX system interfaces .....	17
XSI .....	15
XSI system interfaces .....	19
conformance document .....	16, 3470
rationale .....	3470
conforming application .....	16, 1990, 2457, 3477, 3566, 3701, 3703
conforming application, strictly .....	594, 789, 3474, 3477, 3572
conforming implementation options .....	20
confstr() .....	<b>696</b> , 3755
conj() .....	<b>699</b>
conjf() .....	699
conjl() .....	699
connect() .....	<b>700</b>
connected socket .....	51
connection .....	51
connection indication queue .....	3651
connection mode .....	51
connectionless mode .....	51
consequences of shell errors .....	2353
continue .....	<b>2381</b>
control character .....	51, 3257
control data .....	500
control mode .....	3542
control operator .....	51
control-normal .....	<b>1764</b>
controlling process .....	51
controlling terminal .....	52, 200, 2317, 3480, 3539, 3754
CONTTYPE .....	412
conversion descriptor .....	52, 783, 788, 1116-1117, 1119-1120
conversion specification .....	906, 945, 989, 999, 2029
modified .....	2037
conversion specifier	
modified .....	2055
Coordinated Universal Time (UTC) .....	2628
copy .....	155

copy files commands	
cp .....	2594
dd .....	2630
ln .....	2887
mv .....	3006
pax .....	3057
copysign() .....	<b>703</b>
copysignf() .....	703
copysignl() .....	703
core .....	2224, 3502
core file .....	52, 555
cos() .....	<b>704</b>
cosf() .....	704
cosh() .....	<b>706</b>
coshf() .....	706
coshl() .....	706
cosl() .....	704, 708
covert channel .....	1219, 3503
cp .....	<b>2594</b> , 3704, 3759
cpio	
rationale for omission .....	3743
cpio format .....	3078
cpow() .....	<b>709</b>
cpowf() .....	709
cpowl() .....	709
cpp	
rationale for omission .....	3743
cproj() .....	<b>710</b>
cprojf() .....	710
cprojl() .....	710
CPT .....	<b>7</b>
CPU .....	400
CPU time .....	52, 3288
clock .....	52
timer .....	52
CRDLY .....	415
CREAD .....	417
creal() .....	<b>711</b>
crealf() .....	711
creall() .....	711
creat() .....	<b>712</b> , 3591, 3704
create a per-process timer .....	2143
create an interprocess channel .....	1423
create session and set process group ID .....	1913
creation-time attribute .....	542, 1484
CRn .....	415
CRNCYSTR .....	267
cron daemon .....	2605
crontab .....	<b>2602</b> , 3758
CRYPT .....	714, 754, 1893
crypt() .....	<b>714</b>
csin() .....	<b>716</b>

csinf()	716
csinh()	717
csinhf()	717
csinhl()	717
csinl()	716, 718
CSIZE	417, 3542
CSn	417
csplit	2606, 3758
csqrt()	719
csqrtf()	719
csqrtl()	719
CSTOPB	417
CS_POSIX_V7_THREADS_LDFLAGS	442
ctags	2610, 3760
ctan()	720
ctanf()	720
ctanh()	721
ctanhf()	721
ctanhl()	721
ctanl()	720, 722
ctermid()	723
ctime()	725, 3757
ctime_r()	725
cu	
rationale for omission	3744
currency_symbol	155
current job	52
current working directory	53, 105, 2317
cursor position	53
cut	2615, 3758
CX	7
cxref	2620
c_	476
C_ constants in <cpio.h>	227
C_IRGRP	227
C_IROTH	227
C_IRUSR	227
C_ISBLK	227
C_ISCHR	227
C_ISCTG	227
C_ISDIR	227
C_ISFIFO	227
C_ISGID	227
C_ISLNK	227
C_ISREG	227
C_ISSOCK	227
C_ISUID	227
C_ISVTX	227
C_IWGRP	227
C_IWOTH	227
C_IWUSR	227
C_IXGRP	227

C_IXOTH.....	227
C_IXUSR.....	227
data access.....	3750, 3754
data key creation .....	1646
data keywords .....	3108
data messages .....	500
data segment.....	53
data structure	
dirent.....	231
entry .....	326
group.....	259
lconv .....	285
msqid_ds .....	375
stat .....	391
data type.....	547, 3677
ACTION .....	326
cc_t.....	414
DIR .....	231
div_t .....	358
ENTRY .....	326
FILE .....	354
fpos_t.....	354
glob_t .....	257
ldiv_t .....	358
lldiv_t.....	358
mbstate_t .....	457
msglen_t .....	375
msgqnum_t .....	375
nl_catd.....	311
nl_item .....	311
pid_t .....	331
ptrdiff_t.....	345
regex_t.....	322
regmatch_t.....	322
regoff_t.....	322
shmatt_t .....	383
sigset_t .....	331
sig_atomic_t .....	331
size_t .....	345
speed_t.....	414
tcflag_t.....	414
VISIT .....	326
wchar_t .....	345
wctrans_t .....	462
wctype_t .....	457
wint_t .....	457
data types	
defined in <fenv.h> .....	243
defined in <sys/types.h> .....	401
date.....	2623, 3759
conversion specifications .....	2623
modified conversion specifications .....	2624



DATEMSK.....	177, 1016
datum.....	298
daylight.....	727, 2176
DAY_.....	267
DBL_ constants	
defined in <float.h>.....	248
DBL_DIG.....	249
DBL_EPSILON.....	250
DBL_MANT_DIG.....	248
DBL_MAX.....	250
DBL_MAX_10_EXP.....	250
DBL_MAX_EXP.....	249
DBL_MIN.....	250, 602, 604, 608, 610, 614, 774, 796, 798, 800, 830, 882
DBL_MIN_10_EXP.....	249
DBL_MIN_EXP.....	249
DBM.....	298, 728-729
DBM_.....	475
dbm_.....	475
dbm_clearerr().....	728
dbm_close().....	728
dbm_delete().....	728
dbm_error().....	728
dbm_fetch().....	728
dbm_firstkey().....	728
DBM_INSERT.....	298, 729
dbm_nextkey().....	728
dbm_open().....	728
DBM_REPLACE.....	298, 729
dbm_store().....	728
dc	
rationale for omission.....	3744
dd.....	2630, 3704, 3758-3759
DEAD_PROCESS.....	455, 769-770
DECIMAL_DIG.....	248
default initialization.....	107
default queue.....	2435
DEFECHO.....	477
deferred batch service.....	53, 2423
deferred cancelability.....	1642
defined types.....	547, 3677
definitions.....	3472
delay process execution.....	1989
DELAYTIMER_MAX.....	271, 2089, 2147, 3768
delete batch job request.....	2432
delta.....	2639
dependency order.....	742
descriptive name.....	930
destination.....	2440
destroying a mutex.....	1654
destructor functions.....	1645
detaching a thread.....	1628
determinism.....	3750

device .....	53
output.....	198
device ID.....	53
device number .....	3481
device, logical .....	3487
DEV_BSIZE .....	394
dev_t.....	401
df.....	2643, 3704, 3759-3760
diff.....	2647, 3758-3759
binary output format .....	2649
default output format .....	2649
directory comparison format.....	2648
-c or -C output format.....	2650
-e output format.....	2650
-f output format.....	2650
-u or -U output format.....	2651
difftime() .....	732
DIR .....	231, 547, 690, 1770, 1772, 1814, 1835, 2132
dircmp	
rationale for omission.....	3744
direct I/O.....	3481
directive .....	906, 945, 989, 999
directory .....	53, 3481
device .....	3536
entry .....	53, 3481
files .....	3536
link.....	53
list .....	2912
operations.....	837
protection.....	108, 3501
root .....	3490
stream.....	54
structure.....	3536
directory commands	
cd .....	2550
pwd.....	3119
dirent.....	231, 837
dirfd() .....	733
dirname.....	2656, 3757, 3759
dirname() .....	734
DIRTYPE.....	412
dis	
rationale for omission.....	3744
disarm (a timer) .....	54
disk space commands	
df.....	2643
du.....	2659
ulimit.....	3315
display .....	54, 3481
display line.....	54
div() .....	736
dlclose() .....	737

dlderror()	739
dlopen()	741
dlsym()	744
documentation	16, 2981
dollar-sign	54
domain error	117
dot	54, 837, 1811, 2383, 3481
dot-dot	55, 837, 1811, 3481, 3489, 3506
double-quote	55, 2336, 3706
downshifting	55
dprintf()	746, 906
drand48()	747
driver	55
du	2659, 3704, 3759-3760
dup()	750, 3591, 3754
dup2()	750, 3591, 3754
duplicating an input file descriptor	2353
duplicating an output file descriptor	2353
duplocale()	752
DUP_COUNT	192
dynamic package initialization	1694
d_	475
D_FMT	267
D_T_FMT	267
E2BIG	234, 481
EACCES	234, 482
EADDRINUSE	234, 482
EADDRNOTAVAIL	234, 482
EAFNOSUPPORT	234, 482
EAGAIN	234, 482, 487
EAI_AGAIN	303, 1006
EAI_BADFLAGS	303, 1006
EAI_FAIL	303, 1006
EAI_FAMILY	303, 1006
EAI_MEMORY	303, 1006
EAI_NONAME	303, 1006
EAI_OVERFLOW	303, 1006
EAI_SERVICE	303, 1006
EAI_SOCKTYPE	303, 1006
EAI_SYSTEM	303, 1006
EALREADY	234, 482
EBADF	234, 482
EBADMSG	234, 482
EBUSY	234, 482, 3564, 3634
ECANCELED	234, 482, 3562
ECHILD	234, 482
ECHO	417
echo	2663, 3757
ECHOCTL	477
ECHOE	417, 3542
ECHOK	417, 3542
ECHOKE	477

ECHONL .....	417, 3542
ECHOPRT .....	477
ECONNABORTED .....	234, 483
ECONNREFUSED .....	234, 483
ECONNRESET .....	234, 483
ecvt() .....	3680
ed .....	2666, 3758-3759
addresses .....	2668
append command .....	2670
change command .....	2671
commands .....	2669
copy command .....	2676
delete command .....	2671
edit command .....	2671
edit without checking command .....	2671
filename command .....	2672
global command .....	2672
global non-matched command .....	2676
help command .....	2673
help-mode command .....	2673
insert command .....	2673
interactive global command .....	2672
interactive global not-matched command .....	2676
join command .....	2673
line number command .....	2677
list command .....	2673
mark command .....	2673
move command .....	2674
null command .....	2677
number command .....	2674
print command .....	2674
prompt command .....	2674
quit command .....	2674
quit without checking command .....	2674
read command .....	2675
regular expressions .....	2668
shell escape command .....	2677
substitute command .....	2675
undo command .....	2676
write command .....	2676
EDEADLK .....	234, 483
EDESTADDRREQ .....	234, 483
edit buffer .....	2686, 3364
edit line .....	3220
editors .....	
ed .....	2666
ex .....	2686
sed .....	3205
vi .....	3364
EDOM .....	234, 483, 3564
EDQUOT .....	234, 483
ED_FILE_MAX .....	2678

ED_LINE_MAX .....	2679
EEXIST .....	234, 483
EFAULT .....	235, 483, 3562
EFBIG .....	235, 483
effective group ID.....	55, 671, 790, 1037, 2317
effective user ID.....	55, 571, 790, 1219, 2317, 3501
EFTYPE.....	3562
EHOSTUNREACH .....	235, 483
EIDRM .....	235, 483
eight-bit transparency.....	55
Eighth Edition UNIX .....	2306, 2589
EILSEQ.....	235, 483, 499, 3564
EINPROGRESS.....	235, 483, 504, 3585
EINTR.....	235, 484, 520, 3562, 3565, 3574-3575
EINVAL .....	235, 484, 3562
EIO.....	235, 484
EISCONN .....	235, 484
EISDIR.....	235, 484
ELOOP .....	235, 484, 3562
ELSIZE .....	1282
emacs	
rationale for omission.....	3744
EMFILE.....	235, 484
EMLINK .....	235, 484
EMPTY .....	455, 770
empty directory .....	56
empty line.....	56
empty string (or null string).....	56
empty wide-character string .....	56
EMSGSIZE.....	235, 484
EMULTIHOP .....	235, 484
ENAMETOOLONG.....	235, 484, 3563
encoding	
character .....	128
encoding rule .....	56
encrypt().....	754
encryption .....	22
endgrent() .....	756, 3499
endhostent().....	758
endnetent() .....	760
endprotoent() .....	762
endpwent() .....	764, 3499
endservent().....	767
endutxent() .....	769
ENETDOWN .....	235, 484
ENETRESET .....	235, 485
ENETUNREACH.....	235, 485
ENFILE .....	235, 485
ENOBUFS.....	235, 485
ENODATA .....	235, 485
ENODEV .....	235, 485
ENOENT .....	235, 485

ENOEXEC .....	235, 485
ENOLCK .....	235, 485
ENOLINK.....	235, 485
ENOMEM.....	235, 485, 3563
ENOMSG.....	235, 485
ENOPROTOOPT.....	235, 485
ENOSPC .....	235, 485
ENOSR.....	235, 485
ENOSTR .....	236, 486
ENOSYS.....	236, 486, 3563, 3588
ENOTCONN.....	236, 486
ENOTDIR.....	236, 486
ENOTEMPTY .....	236, 486
ENOTRECOVERABLE.....	236, 486
ENOTSOCK .....	236, 486
ENOTSUP.....	236, 486, 3563
ENOTTY.....	236, 486, 3536, 3562-3563
entire regular expression.....	56, 181
ENTRY.....	1110
env .....	<b>2682</b> , 3757, 3760
environ.....	<b>772</b> , 790
environment access .....	3750, 3755
environment variable.....	3525
definition .....	3525
internationalization.....	174
envp.....	790
ENXIO.....	236, 486
EOF.....	355
EOPNOTSUPP.....	236, 486
EOVERFLOW .....	236, 486, 3563
EOWNERDEAD.....	236, 486
EPERM.....	236, 486, 2599, 3643
EPIPE.....	236, 486, 3564
Epoch .....	57, 3482, 3507, 3607
EPROTO .....	236, 487
EPROTONOSUPPORT.....	236, 487
EPROTOTYPE .....	236, 487
equivalence class .....	57
era .....	57
ERA .....	267
erand48().....	747, 773
ERANGE.....	236, 487, 3564
ERASE.....	3540
ERA_D_FMT .....	267
ERA_D_T_FMT .....	267
ERA_T_FMT .....	267
ERE.....	3533
alternation .....	3534
bracket expression.....	3534
expression anchoring.....	3534
grammar .....	3535
grammar lexical conventions .....	3535

matching a collating element.....	3534
matching a single character .....	3534
matching multiple characters.....	3534
ordinary character .....	3534
periods .....	3534
precedence.....	3534
special character .....	3534
erf() .....	774
erfc().....	776
erfcf() .....	776
erfcl() .....	776
erff() .....	774, 778
erfl() .....	774, 778
EROFS.....	236, 487, 3564
errno .....	779, 3561
per-thread .....	3564
error conditions .....	3509, 3725
mathematical functions .....	117
error descriptions .....	1006
error handling.....	3734
error numbers .....	481, 3561, 3565
additional .....	488
Error_Path .....	2426
escape character.....	3706
escape character (backslash) .....	2336
escape sequences	
awk .....	2482
gencat .....	2809
lex .....	2878
ESPIPE .....	236, 487
ESRCH .....	236, 487
EST5EDT.....	2176
establish cancellation handlers.....	1600
establish the locale .....	2321
ESTALE.....	236, 487
ETIME .....	236, 487
ETIMEDOUT .....	236, 487
ETXTBSY .....	236, 487
eval .....	2385
event management.....	57
EWOULDBLOCK.....	236, 487
ex .....	2686, 3758-3759
<backslash>.....	2699
<control>-D command .....	2722
<newline>.....	2699
abbreviate command .....	2702
addressing .....	2692
adjust window command .....	2720
append command .....	2702
args command .....	2703
autoindent option.....	2724
autoprint option .....	2725

autowrite option.....	2725
beautify option .....	2725
change command .....	2703
chdir command .....	2703
command descriptions .....	2699
copy command .....	2703
delete command .....	2704
directory option.....	2726
edcompatible option.....	2726
edit command .....	2704
edit options .....	2724
errorbells option .....	2726
escape command .....	2721
execute command .....	2723
exrc command.....	2726
file command .....	2705
global command.....	2705
ignorecase option .....	2726
initialization .....	2690
input editing .....	2697
insert command.....	2706
join command .....	2706
list command .....	2707, 2726
magic command .....	2727
map command.....	2707
mark command .....	2709
mesg command .....	2727
move command.....	2710
next command .....	2710
number command .....	2711
number option.....	2727
open command .....	2711
paragraphs option.....	2727
preserve command.....	2686, 2711
print command .....	2711
prompt command .....	2727
put command.....	2712
quit command.....	2712
read command.....	2712
readonly command .....	2728
recover command.....	2713
redraw command .....	2728
regular expressions .....	2723
remap command .....	2728
replacement strings.....	2724
report command .....	2728
rewind command .....	2713
scroll command .....	2698, 2729
sections command.....	2729
set command.....	2714
shell command.....	2714
shell option.....	2729



shift left command .....	2722
shift right command .....	2722
shiftwidth option.....	2729
showmatch option.....	2729
showmode command .....	2729
slowopen command.....	2730
source command .....	2714
substitute command .....	2714
suspend command.....	2716
tabstop option.....	2730
tag command .....	2716
taglength option .....	2730
tags command.....	2730
term command .....	2730
terse command .....	2730
unabbrev command.....	2717
undo command .....	2717
unmap command .....	2717
version command.....	2718
visual command .....	2718
warn command.....	2731
window command .....	2731
wrapmargin option.....	2731
wrapscreen option.....	2732
write command .....	2718
write line number command.....	2723
writeln option .....	2732
xit command .....	2719
yank command.....	2720
examine and change blocked signals .....	1728
examine and change signal action .....	1945
EXDEV .....	236, 488
exec .....	781, 2387, 3030
of shell scripts .....	789
exec family .....	571, 688, 824, 870, 897, 1552, 1901, 2223, 2334, 2589, 3014, 3441, 3484
.....	3589, 3699, 3732, 3754
exec().....	3670
execl() .....	781
execle() .....	781
execlp().....	781
executable file .....	57
execute .....	58
execute a file.....	789
execution time.....	52, 58
measurement.....	110, 3504
monitoring.....	58, 512, 3612
Execution_Time .....	2427
execv() .....	781
execve() .....	781
execvp().....	781
EXINIT .....	2686
exit .....	2389

exit status .....	3725
and errors .....	2353
for commands .....	2354
exit() .....	<b>794</b> , 3572-3573, 3754
EXIT_FAILURE .....	358, 553, 794, 3756
EXIT_SUCCESS .....	358, 553, 794, 3756
exp() .....	<b>796</b>
exp2() .....	<b>798</b>
exp2f() .....	798
exp2l() .....	798
expand .....	58, 2759, 3758
expf() .....	796
expl() .....	796
expm1() .....	<b>800</b>
expm1f() .....	800
expm1l() .....	800
export .....	<b>2391</b>
expr .....	<b>2762</b> , 3757-3758
matching expression .....	2764
expression argument .....	2485
expression list .....	2485
EXPR_NEST_MAX .....	275, 2089, 2324, 3698
EXTA .....	477
EXTB .....	477
extended regular expression .....	58, 188, 2481, 2598, 2789, 2831, 2877, 3008, 3067, 3188
.....	3438, 3533
extended security controls .....	58, 108, 3501
extension	
CX .....	7
OH .....	9
XSI .....	12
F-LOCK .....	<b>443</b>
fabs() .....	<b>802</b>
fabsf() .....	802
fabsl() .....	802
faccessat() .....	569, 804
false .....	2334, 2357, 2767, 3705, 3757
fattach() .....	<b>805</b>
fc .....	2334, 2357, 2769, 3705, 3758
fchdir() .....	<b>808</b>
fchmod() .....	<b>809</b> , 3754
fchmodat() .....	664, 811
fchown() .....	<b>812</b>
fchownat() .....	669, 814
fclose() .....	<b>815</b> , 3754
fcntl() .....	<b>818</b> , 3537, 3562, 3591, 3754
fcntl() locks .....	3649
fcvt() .....	3680
FD .....	7
fdatasync() .....	<b>827</b>
fdetach() .....	<b>828</b>
fdim() .....	<b>830</b>

fdimf()	830
fdiml()	830
fdopen()	832, 3591
fdopendir()	835
fds_	475
FD_	475
fd_	475
FD_CLOEXEC	238, 497, 648, 751, 782, 818, 837, 1120, 1400, 1445, 1452, 1924
FD_CLR	1545
FD_CLR()	552
FD_ISSET	552, 1545
fd_set	379, 398
FD_SET	552, 1545
FD_SETSIZE	379
FD_ZERO	552, 1545
feature test macro	59, 472, 1010, 3557, 3677
_POSIX_C_SOURCE	472
_XOPEN_SOURCE	473
feclearexcept()	839
fegetenv()	840
fegetexceptflag()	841
fegetround()	842
feholdexcept()	844
fenv_t	243
feof()	845
feraiseexcept()	846
error()	847
fesetenv()	840, 848
fesetexceptflag()	841, 849
fesetround()	842, 850
fetestexcept()	851
feupdateenv()	853
fexcept_t	243
fexecve	855
fexecve()	781
FE_	477
FE_constants	
defined in <fenv.h>	243
FE_ALL_EXCEPT	243
FE_DFL_ENV	244
FE_DIVBYZERO	243
FE_DOWNWARD	243
FE_INEXACT	243
FE_INVALID	243
FE_OVERFLOW	243
FE_TONEAREST	243
FE_TOWARDZERO	243
FE_UNDERFLOW	243
FE_UPWARD	243
FFDLY	416
fflush()	856
FFn	416

ffs()	859
fg	2334, 2357, 2775, 3705, 3758
fgetc()	860, 3755
fgetpos()	862
fgets()	864
fgetwc()	866
fgetws()	868
field	59
field splitting	2349, 3722
FIFO	59, 1315, 1317, 1406, 2304, 3482, 3490, 3600
FIFO special file	59, 2991, 3482
FIFOTYPE	412
file	59
FILE	355, 457, 547
file	2777, 3482
locking	824
file access permissions	108, 2318, 3501
file accessibility	571
file characteristics	
data structure	394
header	394
file classes	3482
file comparisons	
cmp	2577
comm	2581
diff	2647
uniq	3335
file contents	2320
file control	824
file conversion	
cut	2615
dd	2630
expand	2759
fold	2795
head	2839
join	2863
od	3033
paste	3041
patch	3045
sort	3236
strings	3248
tail	3266
tr	3299
tsort	3308
unexpand	3329
uniq	3335
uudecode	3346
uuencode	3349
file creation	2318, 3696
file description	59
file descriptor	60, 2317, 2350, 2358, 3575, 3679, 3723
file format notation	3510

file group class .....	60
file hierarchy .....	109, 3501
file hierarchy manipulation .....	3752, 3759
file mode .....	60
file mode bits .....	60
file mode creation mask .....	2317
FILE object .....	496
file offset .....	61
file other class .....	61
file owner class .....	61
file permission bits .....	61, 571
file permission commands	
chgrp .....	2558
chmod .....	2561
chown .....	2568
umask .....	3317
file permissions .....	571, 902, 964, 3501, 3539
file position indicator .....	495
file read .....	2318, 3696
file removal .....	2320, 3696
file searching	
grep .....	2831
file serial number .....	61
file size, arbitrary .....	3704
file system .....	61, 3482
file system, mounted .....	3488
file system, root .....	3490-3491
file time values .....	2320
file times update .....	109, 3503
file tree commands	
diff .....	2647
find .....	2785
ls .....	2912
mkdir .....	2988
rmdir .....	3194
file type .....	62
file write .....	2318, 3696
file, passwd .....	3489
filename .....	60, 109, 3482, 3502
filename portability .....	109, 3503
filename string .....	61
FILENAME_MAX .....	354
fileno() .....	870, 3488
FILESIZEBITS .....	273, 899
filter .....	62
filtering trace event types .....	3673
filters	
asa .....	2460
awk .....	2472
compress .....	2591
dd .....	2630
expand .....	2759

fold .....	2795
head .....	2839
iconv .....	2842
more .....	2994
nl .....	3020
paste .....	3041
pax .....	3057
pr .....	3096
read .....	3180
sed .....	3205
tail .....	3266
tee .....	3274
tr .....	3299
uncompress .....	3326
unexpand .....	3329
zcat .....	3460
FIND .....	1110
find .....	2785, 3758-3759
find string token .....	2069
FIPS .....	17
FIPS requirements .....	3469
first open (of a file) .....	62
flockfile() .....	871, 3565
floor() .....	873
floorf() .....	873
floorl() .....	873
flow control .....	62
FLT_ constants	
defined in <float.h> .....	248
FLT_DIG .....	249
FLT_EPSILON .....	250
FLT_EVAL_METHOD .....	247
FLT_MANT_DIG .....	248
FLT_MAX .....	250
FLT_MAX_10_EXP .....	250
FLT_MAX_EXP .....	249
FLT_MIN .....	250, 602, 604, 608, 610, 614, 774, 796, 798, 800, 830, 882
FLT_MIN_10_EXP .....	249
FLT_MIN_EXP .....	249
FLT_RADIX .....	248, 1272
FLT_ROUNDS .....	247, 875
FLUSH .....	475
FLUSHO .....	477
FLUSHR .....	367, 1140
FLUSHRW .....	367, 1140
FLUSHW .....	367, 1140
fma() .....	875
fmaf() .....	875
fmal() .....	875
fmax() .....	877
fmaxf() .....	877
fmaxl() .....	877

fmemopen()	878
fmin()	881
fminf()	881
fminl()	881
FMNAMESZ	366-367, 1139
fmod()	882
fmodf()	882
fmodl()	882
fmtmsg()	884
fnmatch()	887, 3760
FNM_	475
FNM_ constants	
in <fnmatch.h>	254
FNM_NOESCAPE	254, 887
FNM_NOMATCH	254, 887
FNM_PATHNAME	254, 887
FNM_PERIOD	254, 887
fold	2795, 3758
fopen()	889, 3483, 3704, 3754
FOPEN_MAX	272, 354, 832, 879, 891, 2153
for loop	2362, 3730
foreground	1900, 3484-3487, 3538-3540
foreground job	62
foreground process	62
group	62
group ID	62
fork()	894, 3484, 3538, 3580, 3589, 3591, 3670, 3677, 3682, 3754
forkall	897
form-feed character	63
format of entries	219, 469
fort77	2799, 3760
external symbols	2802
standard libraries	2801
fpathconf()	899, 3753, 3755
fpclassify()	905
FPE_	475
FPE_FLTDIV	336
FPE_FLTINV	336
FPE_FLTOVF	336
FPE_FLTRES	336
FPE_FLTSUB	336
FPE_FLTUND	336
FPE_INTDIV	336
FPE_INTOVF	336
fprintf()	906
fputc()	919, 3755
fputs()	921
fputwc()	923
fputws()	925
FP_ILOGB0	1126
FP_ILOGBNAN	1126
FQDN	1052

FR.....	7
frac_digits.....	156
fread().....	927, 3755
free().....	929, 3573, 3646
freeaddrinfo().....	930
freelocale().....	936
freopen().....	938
frexp().....	943
frexpf().....	943
frexpl().....	943
fsblkcnt_t.....	401
FSC.....	8
fscanf().....	945
fseek().....	953, 3755
fseeko().....	953
fsetpos().....	956, 3755
fsfilcnt_t.....	401
fstat().....	958, 3754
fstatat().....	961
fstatvfs().....	967
fsync().....	970, 3584
ftell().....	972
ftello(0).....	972
ftime().....	3680
ftok().....	974
ftruncate().....	976, 3590-3591, 3593, 3754
ftrylockfile().....	871, 978
FTW.....	255, 475, 1390-1391
ftw().....	979, 3704
FTW_ constants	
in <ftw.h>.....	255
FTW_CHDIR.....	255, 1390
FTW_D.....	255, 979, 1390
FTW_DEPTH.....	255, 1390
FTW_DNR.....	255, 979, 1390-1391
FTW_DP.....	255, 1390
FTW_F.....	255, 979, 1390
FTW_MOUNT.....	255, 1390
FTW_NS.....	255, 979, 1390-1391
FTW_PHYS.....	255, 1390
FTW_SL.....	255, 979, 1390
FTW_SLN.....	255, 1391
fully-qualified domain name.....	1052
function definition command.....	2364, 3731
function identifiers.....	2519
functions.....	471
implementation.....	471, 3555
use.....	471, 3555
funlockfile().....	871, 982
fuser.....	2805
futimens().....	983
fwide().....	987



fwprintf()	989
fwrite()	997, 3755
fwscanf()	999
f_	476
F_	477
F_DUPFD	238, 818, 821
F_DUPFD_CLOEXEC	238, 818, 821
F_GETFD	238, 818, 821
F_GETFL	238, 818, 821
F_GETLK	238, 819, 821
F_GETOWN	238, 819, 821
F_LOCK	518, 1261
F_OK	440
F_RDLCK	238, 821
F_SETFD	238, 818, 821
F_SETFL	238, 818, 821
F_SETLK	238, 819, 821
F_SETLKW	238, 518, 819, 821
F_SETOWN	238, 819, 821
F_TEST	443, 1261
F_TLOCK	443, 1261
F_ULOCK	443, 1261
F_UNLCK	238, 819, 821
F_WRLCK	238
g-file	2639
gai_strerror()	1006
gcvt()	3680
gencat	2808, 3759
escape sequences	2809
general terminal interface	3536
generated file	2639
generation-version attribute	542, 1484
get	2812
get configurable pathname variables	901
get configurable system variables	2093
get file status	964
get process times	2150
get supplementary group IDs	1037
get system time	2140
get thread ID	1718
get user name	1046
getaddrinfo()	930, 1007
GETALL	381, 1859
getc()	1008, 3637, 3755
getch()	3755
getchar()	1011
getchar_unlocked()	1009, 1012
getconf	2820, 3697, 3753, 3760
getcontext()	3680
getcwd()	1013
getc_unlocked()	1009
getdate()	1016

getdate_err .....	1016
getdelim() .....	<b>1021</b>
getegid() .....	<b>1023</b> , 3754
getenv() .....	790, 1024
geteuid() .....	<b>1027</b> , 3754
getgid() .....	<b>1028</b> , 3754
getgrent() .....	756, 1029, 3499
getgrgid() .....	<b>1030</b> , 3499, 3638, 3754
getgrgid_r() .....	1030
getgrnam() .....	<b>1034</b> , 3499, 3504, 3638, 3754
getgrnam_r() .....	1034
getgroups() .....	<b>1037</b> , 3491
gethostbyaddr() .....	3680
gethostbyname() .....	3680
gethostent() .....	758, 1039
gethostid() .....	<b>1040</b>
gethostname() .....	<b>1041</b>
getitimer() .....	<b>1042</b>
getline() .....	1021, 1044
getlogin() .....	<b>1045</b> , 3754
getlogin_r() .....	1045
getmsg() .....	<b>1048</b>
getnameinfo() .....	<b>1052</b>
GETNCNT .....	381, 1859-1860
getnetbyaddr() .....	760, 1055
getnetbyname() .....	760, 1055
getnetent() .....	760, 1055
getopt() .....	<b>1056</b> , 3545, 3759-3760
getopts .....	2334, 2357, 2826, 3705, 3760
getpeername() .....	<b>1060</b>
getpgid() .....	<b>1062</b>
getpgrp() .....	<b>1063</b> , 3486
GETPID .....	381, 1859-1860
getpid() .....	<b>1064</b> , 3575, 3754
getpmsg() .....	1048, 1065
getppid() .....	<b>1066</b> , 3754
getpriority() .....	<b>1067</b> , 3601
getprotent() .....	1070
getprotobyname() .....	762, 1070
getprotobynumber() .....	762, 1070
getprotoent() .....	762
getpwent() .....	764, 1071, 3499
getpwnam() .....	<b>1072</b> , 3499, 3504, 3638, 3754
getpwnam_r() .....	1072
getpwuid() .....	<b>1076</b> , 3499, 3638, 3754
getpwuid_r() .....	1076
getrlimit() .....	<b>1080</b> , 3704
getrusage() .....	<b>1083</b> , 3614
gets() .....	<b>1085</b>
getservbyname() .....	767, 1087
getservbyport() .....	767, 1087
getservent() .....	767, 1087

getsid()	1088
getsockname()	1089
getsockopt()	1091
getsubopt()	1093
gettimeofday()	1097
getty	3539
getuid()	1098, 3575, 3678, 3754
getutxent()	769, 1099
getutxid()	769, 1099
getutxline()	769, 1099
GETVAL	381, 1859-1860
getwc()	1100
getwchar()	1101
getwd()	3680
GETZCNT	381, 1859-1860
gid_t	401, 3499
glob()	1102, 3760
global storage class	2523
globfree()	1102, 3760
GLOB_	475
GLOB_ constants	
defined in <glob.h>	257
error returns of glob	1104
used in glob	1102
GLOB_ABORTED	257, 1104
GLOB_APPEND	257, 1102-1103
GLOB_DOOFFS	257, 1102-1103
GLOB_ERR	257, 1102, 1104
GLOB_MARK	257, 1102
GLOB_NOCHECK	257, 1103-1104
GLOB_NOESCAPE	257, 1103
GLOB_NOMATCH	257, 1104
GLOB_NOSORT	257, 1103
GLOB_NOSPACE	257, 1104
gl_	475
GMT0	2176
gmtime()	1106, 3508
gmtime_r()	1106
GNU make	2976
grammar	
conventions	3700
locale	166
regular expression	192
grantpt()	1108
graphic character	63
grep	2831, 3758-3759
group database	63, 3483
group database access	3500
group file	3483
group ID	63
group name	63
grouping commands	2361, 3730

HALT.....	885
hard limit.....	64
hard link .....	64
hash .....	2334, 2357, 2836
hcreate().....	<b>1110</b>
hdestroy() .....	1110
head.....	<b>2839</b> , 3758
headers.....	<b>219</b> , 3546
here-document.....	2352, 3724
high resolution sleep.....	1382
historical implementations .....	3483
history command	
fc .....	2769
hold batch job request.....	2433
Hold_Types .....	2427
HOME.....	<b>177</b> , 2703, 3469
home directory.....	64
host byte order.....	64, 110, 3504
host name .....	930
hosted implementation .....	3483
hostent.....	<b>301</b>
HOST_NAME_MAX .....	271
hsearch() .....	1110
htonl() .....	<b>1113</b>
htons() .....	1113
HUGE_VAL.....	289, 2261
HUGE_VALF .....	289
HUGE_VALL .....	289
hunk .....	3047
HUPCL .....	417
hypot() .....	<b>1114</b>
hypotf().....	1114
hypotl() .....	1114
h_ .....	475
h_errno.....	3681
I .....	224
ICANON .....	417, 3540, 3542
iconv .....	<b>2842</b>
iconv().....	<b>1116</b> , 3759
iconv_close() .....	<b>1119</b> , 3759
iconv_open().....	<b>1120</b> , 3759
ICRNL .....	415
ic_.....	475
id.....	<b>2846</b> , 3760
idtype_t.....	<b>408</b>
id_t.....	401
IEEE Std 754-1985.....	469
IEEE Std 854-1987.....	469
IEXTEN.....	417
if.....	3731
if conditional construct.....	2363
ifc_ .....	476

ifra_.....	476
ifru_.....	476
IF_.....	475
if_.....	475-476
if_freenameindex().....	<b>1122</b>
if_indextoname().....	<b>1123</b>
if_nameindex.....	<b>300</b>
if_nameindex().....	<b>1124</b>
IF_NAMESIZE.....	300
if_nametoindex().....	<b>1125</b>
IGNBRK.....	415
IGNCR.....	415
IGNPAR.....	415
ILL_.....	475
ILL_BADSTK.....	336
ILL_COPROC.....	336
ILL_ILLADR.....	336
ILL_ILLOPC.....	336
ILL_ILLOPN.....	336
ILL_ILLTRP.....	336
ILL_PRVOPC.....	336
ILL_PRVREG.....	336
ilogb().....	<b>1126</b>
ilogbf().....	1126
ilogbl().....	1126
imaginary.....	224
imaxabs().....	<b>1128</b>
imaxdiv().....	<b>1129</b>
implementation.....	3483
historical.....	3483
hosted.....	3483
native.....	3488
specific.....	3483
implementation-defined.....	5, 3470-3471
rationale.....	3470
IMPLINK_.....	477
in6_.....	475
IN6_.....	477
IN6_IS_ADDR_LINKLOCAL.....	307
IN6_IS_ADDR_LOOPBACK.....	307
IN6_IS_ADDR_MC_GLOBAL.....	308
IN6_IS_ADDR_MC_LINKLOCAL.....	308
IN6_IS_ADDR_MC_NODELOCAL.....	307
IN6_IS_ADDR_MC_ORGLOCAL.....	308
IN6_IS_ADDR_MC_SITELOCAL.....	308
IN6_IS_ADDR_MULTICAST.....	307
IN6_IS_ADDR_SITELOCAL.....	307
IN6_IS_ADDR_UNSPECIFIED.....	307
IN6_IS_ADDR_V4COMPAT.....	307
IN6_IS_ADDR_V4MAPPED.....	307
INADDR_.....	475
include line.....	2962

incomplete line .....	64
incomplete pathname .....	3483
index() .....	3681
INET6_ADDRSTRLEN .....	307
inet_ .....	475
inet_addr() .....	<b>1130</b>
INET_ADDRSTRLEN .....	307
inet_ntoa() .....	1130
inet_ntop() .....	<b>1132</b>
inet_pton() .....	1132
Inf .....	64, 602, 604, 608, 614
INF .....	909, 992
inference rule .....	2958
INFINITY .....	289, 909, 992
INFO .....	885
infu_ .....	476
inheritance attribute .....	542, 1486
init .....	556, 1219
initialize a named semaphore .....	1847
initializing a mutex .....	1654
initstate() .....	<b>1134</b>
INIT_PROCESS .....	455, 769-770
INLCR .....	415
ino_t .....	401
INPCK .....	415
input and output rationale .....	1766
input file descriptor	
duplication .....	3724
input mode .....	2666, 3541
input processing .....	3540
canonical mode .....	3540
non-canonical mode .....	3540
insque() .....	<b>1136</b>
instrumented application .....	64
INT .....	477
inter-user communication .....	3752, 3759
interactive facilities .....	3751, 3758
interactive shell .....	65
interface .....	3650
characteristics .....	3537
international environment .....	1895
internationalization .....	65
internationalization variable .....	3526
Internet Protocols .....	532
interprocess communication .....	65, 3576
INTMAX_MAX .....	351
INTMAX_MIN .....	351
INTN_MAX .....	350
INTN_MIN .....	350
INTPTR_MAX .....	351
INTPTR_MIN .....	350
int_curr_symbol .....	155

INT_FASTN_MAX .....	350
INT_FASTN_MIN .....	350
int_frac_digits .....	156
INT_LEASTN_MAX .....	350
INT_LEASTN_MIN .....	350
INT_MAX .....	280, 1126
INT_MIN .....	280, 566
int_n_cs_precedes .....	156
int_n_sep_by_space .....	157
int_n_sign_posn .....	157
int_p_cs_precedes .....	156
int_p_sep_by_space .....	157
int_p_sign_posn .....	157
invalid .....	182
use in RE .....	3530
invariant values .....	281
invoke .....	65
in_ .....	475
IN_ .....	477
in_addr .....	305
ioctl() .....	1139, 3537, 3563
iovec .....	405
iov_ .....	476
IOV_ .....	477
IOV_MAX .....	271, 405, 1778, 2089, 2309
IP6 .....	8
IPC .....	370, 501, 1365, 1367, 1370, 1372, 1863, 1868, 1935, 1938, 3576
ipcrm .....	2850
ipcs .....	2852
IPC_ .....	475
ipc_ .....	475
IPC_constants	
defined in <sys/ipc.h> .....	370
used in semctl .....	1860
used in shmctl .....	1933
IPC_CREAT .....	370, 1366, 1862, 1937
IPC_EXCL .....	370, 1366, 1862
IPC_NOWAIT .....	370, 1368-1369, 1371-1372, 1864
IPC_PRIVATE .....	370, 1366, 1862, 1937
IPC_RMID .....	370, 1364, 1860, 1933
IPC_SET .....	370, 1364, 1860, 1933
IPC_STAT .....	370, 1364, 1860, 1933
IPPORT_ .....	477
IPPROTO_ .....	475
IPPROTO_ICMP .....	306
IPPROTO_IP .....	306
IPPROTO_IPV6 .....	306
IPPROTO_RAW .....	306
IPPROTO_TCP .....	306
IPPROTO_UDP .....	306
IPv4 .....	532
IPv4-compatible address .....	533

IPv4-mapped address.....	533
IPv6.....	532
compatibility with IPv4.....	533
interface identification.....	534
options .....	534
IPv6 address	
anycast .....	533
loopback .....	533
multicast .....	533
unicast.....	532
unspecified .....	533
IPV6_.....	475
IPV6_JOIN_GROUP .....	307, 534
IPV6_LEAVE_GROUP .....	307, 534
ipv6_mreq.....	<b>306</b>
IPV6_MULTICAST_HOPS.....	307, 534
IPV6_MULTICAST_IF.....	307, 534
IPV6_MULTICAST_LOOP .....	307, 534
IPV6_UNICAST_HOPS.....	307, 535
IPV6_V6ONLY .....	307, 535
ip_.....	475
IP_.....	477
isalnum() .....	<b>1151</b>
isalnum_l() .....	1151
isalpha().....	<b>1153</b>
isalpha_l().....	1153
isascii().....	<b>1155</b>
isastream().....	<b>1156</b>
isatty().....	<b>1157</b>
isblank().....	<b>1158</b>
isblank_l().....	1158
iscntrl() .....	<b>1160</b>
iscntrl_l() .....	1160
isdigit() .....	<b>1162</b>
isdigit_l() .....	1162
isfinite() .....	<b>1164</b>
isgraph() .....	<b>1165</b>
isgraph_l().....	1165
isgreater() .....	<b>1167</b>
isgreaterequal() .....	<b>1168</b>
ISIG.....	417
isinf().....	<b>1169</b>
isless() .....	<b>1170</b>
islessequal() .....	<b>1171</b>
islessgreater().....	<b>1172</b>
islower() .....	<b>1173</b>
islower_l() .....	1173
isnan().....	<b>1176</b>
isnormal().....	<b>1177</b>
ISO/IEC 646: 1991 standard.....	3490
ISO C standard.....	224, 469, 594, 789, 824, 1010, 1284, 1760, 1811, 1895, 1946, 1962, 1973
.....	2140, 3477, 3479, 3508, 3537, 3555



isprint()	1178
isprint_l()	1178
ispunct()	1180
ispunct_l()	1180
isspace()	1182
isspace_l()	1182
ISTRIP	415, 3541
isunordered()	1184
isupper()	1185
isupper_l()	1185
iswalnum()	1187
iswalnum_l()	1187
iswalpha()	1189
iswalpha_l()	1189
iswblank()	1191
iswblank_l()	1191
iswcntrl()	1193
iswcntrl_l()	1193
iswctype()	1195
iswctype_l()	1195
iswdigit()	1197
iswdigit_l()	1197
iswgraph()	1199
iswgraph_l()	1199
iswlower()	1201
iswlower_l()	1201
iswprint()	1203
iswprint_l()	1203
iswpunct()	1205
iswpunct_l()	1205
iswspace()	1207
iswspace_l()	1207
iswupper()	1209
iswupper_l()	1209
iswxdigit()	1211
iswxdigit_l()	1211
isxdigit()	1213
isxdigit_l()	1213
itimerspec	424
itimerval	398
ITIMER_	476
ITIMER_PROF	398, 1042
ITIMER_REAL	398, 1042
ITIMER_VIRTUAL	398, 1042
it_	476
IXANY	415
IXOFF	415
IXON	415
I_	475
I_ATMARK	366, 1146
I_CANPUT	366, 1146
I_CKBAND	366, 1146

I_FDINSERT .....	366, 1142
I_FIND .....	366, 1141
I_FLUSH .....	366, 1139
I_FLUSHBAND .....	366, 1140
I_GETBAND .....	366, 1146
I_GETCLTIME .....	366, 1147
I_GETSIG .....	366, 1141
I_GRDOPT .....	366, 1142, 1764
I_GWROPT .....	366, 1144
I_ISVTX .....	2563
I_LINK .....	366, 1147
I_LIST .....	366, 1145
I_LOOK .....	366, 1139
I_NREAD .....	366, 1142
I_PEEK .....	366, 1141
I_PLINK .....	366, 1148
I_POP .....	366, 1139
I_PUNLINK .....	366, 1148
I_PUSH .....	366, 1139
I_RECVFD .....	366, 482, 1145
I_SENDFD .....	366, 1144-1145
I_SETCLTIME .....	366, 686, 1146
I_SETSIG .....	366, 1140-1141
I_SRDOPT .....	366, 1142, 1764
I_STR .....	366, 1143
I_SWROPT .....	367, 1144, 2302
I_UNLINK .....	367, 1147
j0() .....	<b>1215</b>
j1() .....	1215
jn() .....	1215
job .....	65
job control .....	65, 556, 1063, 1219, 1900, 1913, 2093, 2223, 3484-3487, 3489, 3538-3539
.....	3566, 3572, 3754
implementing applications .....	3486
implementing shells .....	3484
implementing systems .....	3486
job control job ID .....	66
jobs .....	2334, 2357, 2859, 3705, 3758
Job_Owner .....	2427
join .....	<b>2863</b> , 3758
Join_Path .....	2427
jr48() .....	747, 1217
JST-9 .....	2176
Keep_Files .....	2427
kernel .....	3487
kernel entity .....	3640
keyword-value pairs .....	2441
key_t .....	401
kill .....	2334, 2357, 2868, 3705, 3758
kill() .....	<b>1218</b> , 3565-3567, 3570, 3572, 3677
killpg() .....	<b>1221</b>
l64a() .....	562, 1223

labs()	1224
LANG	174, 648
last close	1929, 3591
last close (of a file)	66
LASTMARK	368, 1146
lchown()	1225
lcong48()	747, 1228
LC_ALL	175, 285, 783, 1254, 1396, 1894, 1896
LC_COLLATE	175, 275, 285, 1102-1103, 1894, 1896, 2018, 2080, 2239, 2280, 3516
description	147
LC_CTYPE	175, 267, 285, 462, 1195, 1289, 1298, 1300, 1894, 1896, 2158, 2164, 3515
description	139
LC_GLOBAL_LOCALE	752, 936
LC_MESSAGES	175, 267, 285, 311, 648, 1894-1896, 2026, 3521
description	165
LC_MONETARY	175, 267, 285, 1254, 1894, 1896, 2031, 3518
description	155
LC_NUMERIC	175, 267, 285, 907, 945, 989, 999, 1254, 1894, 1896, 2031, 2063, 2261
description	3520, 3545
LC_TIME	175, 267, 285, 1017, 1397, 1894, 1896, 3520
description	159
ld	
rationale for omission	3744
LDBL_constants	
defined in <float.h>	248
LDBL_DIG	249
LDBL_EPSILON	250
LDBL_MANT_DIG	248
LDBL_MAX	250
LDBL_MAX_10_EXP	250
LDBL_MAX_EXP	249
LDBL_MIN	250, 602, 604, 608, 610, 614, 774, 796, 798, 800, 830, 882
LDBL_MIN_10_EXP	249
LDBL_MIN_EXP	249
ldexp()	1229
ldexpf()	1229
ldexpl()	1229
ldiv()	1231
legacy	5, 3471
rationale	3471
lex	2873, 3760, 3762
actions	2879
definitions	2876
escape sequences	2878
regular expressions	2877
rules	2877
table sizes	2876
translation table	2884
user subroutines	2877
lfind()	1232, 1282
lgamma()	1233

lgammaf()	1233
lgammal()	1233
libraries	
ar command	2452
library routine	3487
LIMIT	2323
limit	
numerical	280
limits	3697
line	66
rationale for omission	3744
line counting	3423
LINES	177, 3527
LINE_MAX	275, 2089, 2324, 2473, 2679, 2687, 2954, 3213, 3338, 3488, 3498, 3698
linger	66
link	66, 2885
link count	66
link to a file	1238
link()	1235, 3481, 3755
linkat()	1235
LINK_MAX	273, 484, 899, 1236, 1809, 3698, 3769
lint	
rationale for omission	3744
LIO_	475
lio_	475
lio_listio()	1240, 3569, 3585
LIO_NOP	220, 1240
LIO_NOWAIT	220, 1240
LIO_READ	220, 1240
LIO_WAIT	220, 1240
LIO_WRITE	220, 1240
list directed I/O	1242
listen()	1244
lists	2359, 3729
AND-OR	2359
compound-list	2359
live process	66
llabs()	1224, 1246
lldiv()	1231, 1247
LLONG_MAX	280, 2072, 2268
LLONG_MIN	280, 2072, 2268
llrint()	1248
llrintf()	1248
llrintl()	1248
llround()	1250
llroundf()	1250
llroundl()	1250
ln	2887, 3704, 3759
LNKTYPE	412
load order	742
LOBLK	477
local customs	67

local IPC .....	67
local mode .....	3542
locale .....	67, 135, 2892, 3514, 3759
configuration .....	3752, 3759
definition .....	136, 3514
definition example .....	3522
definition grammar .....	3521
grammar .....	166, 3522
lexical conventions .....	3522
POSIX .....	136
localeconv() .....	<b>1252</b>
localedef .....	<b>2898</b> , 3759-3760
localization .....	67
localtime() .....	<b>1257</b> , 3508, 3636
localtime_r() .....	1257
locate batch job request .....	2434
lockf() .....	<b>1261</b>
locking .....	824
advisory .....	824
mandatory .....	824
locking and unlocking a mutex .....	1665
locking file .....	2565
log() .....	<b>1264</b>
log-full-policy attribute .....	540, 542, 1486, 1489, 1506
log-max-size attribute .....	542, 1487, 1489
log10() .....	<b>1266</b>
log10f() .....	1266
log10l() .....	1266
log1p() .....	<b>1268</b>
log1pf() .....	1268
log1pl() .....	1268
log2() .....	<b>1270</b>
log2f() .....	1270
log2l() .....	1270
logb() .....	<b>1272</b>
logbf() .....	1272
logbl() .....	1272
logf() .....	1264, 1274
logger .....	<b>2902</b> , 3760
logical device .....	3487
login .....	67
rationale for omission .....	3744
login name .....	67
login shell .....	789
LOGIN_NAME_MAX .....	271, 1045, 2089, 3769
LOGIN_PROCESS .....	455, 769-770
logl() .....	1264, 1274
LOGNAME .....	<b>178</b>
logname .....	<b>2905</b>
LOGNAME .....	3469, 3527
logname .....	3760
LOG_ .....	476

LOG_constants in syslog.....	692
LOG_ALERT .....	411, 692
LOG_AUTH .....	410
LOG_CONS.....	410, 693
LOG_CRIT .....	411, 692
LOG_CRON .....	410
LOG_DAEMON .....	410
LOG_DEBUG .....	411, 692
LOG_EMERG.....	411, 692
LOG_ERR .....	411, 692
LOG_INFO .....	411, 692
LOG_KERN.....	410
LOG_LOCAL .....	410, 692
LOG_LPR .....	410
LOG_MAIL .....	410
LOG_MASK .....	410
LOG_NDELAY .....	410, 693
LOG_NEWS .....	410
LOG_NOTICE.....	411, 692
LOG_NOWAIT .....	410, 693
LOG_ODELAY .....	410, 693
LOG_PID .....	410, 693
LOG_USER .....	410, 692-693
LOG_UUCP .....	410
LOG_WARNING.....	411, 692
longjmp() .....	1275, 3562, 3573, 3644, 3646, 3757
LONG_BIT .....	280
LONG_MAX .....	280, 2072, 2268, 3543
LONG_MIN .....	280, 2072, 2268, 3543
lorder	
rationale for omission.....	3744
lower multiplexing.....	94
lp.....	2907, 3760
lpstat	
rationale for omission.....	3744
LR(1) grammars.....	3457
lrand48() .....	747, 1277
lrint() .....	1278
lrintf() .....	1278
lrintl() .....	1278
lround() .....	1280
lroundf() .....	1280
lroundl() .....	1280
ls.....	2912, 3704, 3759
lsearch() .....	1282
lseek() .....	1284, 3584-3585, 3591, 3635, 3755
lstat() .....	961, 1286, 3704, 3754
l_ .....	475-476
L_ANCHOR.....	192
L_ctermid .....	354, 723
l_sysid .....	824
L_tmpnam .....	354

m4 .....	2922
macro .....	3472
macro processor .....	2922
MAGIC .....	227
magic file .....	2782
mail	
rationale for omission .....	3744
mailx .....	2932, 3758-3759
change current directory .....	2943
change folder .....	2945
command escapes .....	2951
commands .....	2942
copy messages .....	2943
declare aliases .....	2943
declare alternatives .....	2943
delete aliases .....	2949
delete messages .....	2944
delete messages and display .....	2944
direct messages to mbox .....	2947
discard header fields .....	2944
display beginning of messages .....	2949
display current message number .....	2951
display header summary .....	2946
display list of folders .....	2945
display message .....	2947
display message size .....	2949
echo a string .....	2944
edit message .....	2944, 2950
execute commands conditionally .....	2946
exit .....	2945
follow up specified messages .....	2945
help .....	2946
hold messages .....	2946
internal variables .....	2939
invoke a shell .....	2949
invoke shell command .....	2950
list available commands .....	2946
mail a message .....	2947
null command .....	2951
pipe message .....	2947
process next specified message .....	2947
quit .....	2948
read mailx commands from a file .....	2949
receive mode .....	2932
reply to a message .....	2948
reply to a message list .....	2948
retain header fields .....	2948
save messages .....	2948
scroll header display .....	2950
send mode .....	2932
set variables .....	2949
start-up .....	2939

touch messages.....	2949
undelete messages.....	2950
unset variables.....	2950
write messages to a file.....	2950
Mail_Points .....	2428
Mail_Users .....	2428
main() .....	3573
make.....	<b>2958</b> , 3760-3761
default rules .....	2969
inference rules.....	2966
internal macros .....	2968
libraries .....	2967
macros.....	2965
makefile execution .....	2963
makefile syntax.....	2961
target rules.....	2963
make, GNU version .....	2976
makecontext() .....	3680-3681
malloc() .....	<b>1287</b> , 3573, 3594, 3596, 3624, 3636-3637, 3646, 3648
man.....	<b>2981</b> , 3758
manipulate signal sets .....	1952
map.....	67, 3487
mapped.....	3487
mappings.....	1336
MAP_ .....	475
MAP_FAILED .....	1337
MAP_FIXED .....	372, 1332
MAP_PRIVATE.....	372, 894, 1332, 1336, 1341, 1374
MAP_SHARED .....	372, 897, 1332-1333
margin code.....	3472
notation.....	13, 3473
marked message.....	68
matched .....	68, 181
mathematical functions .....	2323
domain error .....	117
error conditions .....	117, 3509
NaN arguments .....	118, 3509
pole error .....	117
range error.....	118
max-data-size attribute.....	542, 1489
MAXARGS .....	343
MAXFLOAT .....	289
maximum values.....	275
MAX_CANON .....	273, 899, 3540, 3698, 3769
MAX_INPUT .....	273, 899, 3698, 3769
may.....	5, 3470
rationale.....	3470
mblen() .....	<b>1289</b>
mbrlen().....	<b>1291</b>
mbrtowc() .....	<b>1293</b>
mbsinit().....	<b>1295</b>
mbsnrtowcs().....	1296



mbsrtowcs()	1296
mbstowcs()	1298
mbtowc()	1300
MB_CUR_MAX	358, 1289, 1291, 1293, 1300, 2232, 2282
MB_LEN_MAX	280
MC1	8
MCL	475
MCL_CURRENT	372, 1329
MCL_FUTURE	372, 1329, 1337, 3588
MCL_INHERIT	3589
mcontext_t	334
memccpy()	1302
memchr()	1303
memcmp()	1304
memcpy()	1305
memmove()	1306
memory locking	3587
memory management	505, 3586, 3754
memory management unit	3587
memory mapped files	68
memory object	68, 3487
memory synchronization	111, 3504
memory-resident	68, 3487
memset()	1307
mesg	2985, 3759-3760
message	69
message catalog	69
descriptor	69, 553, 783, 788
generation	2808
message parts	501
message passing	3578, 3754, 3756
message priority	500
high-priority	500
normal	500
priority	500
message queue	69, 3578
message-discard mode	1764
message-nondiscard mode	1764
MET-1MEST	2176
META_CHAR	192
MIL-STD-1753	2803
minimum values	276
Minimum_Cpu_Interval	2425
MINSIGSTKSZ	334, 1949
mkdir	2988, 3759
mkdir()	1308, 3755
mkdirat()	1308
mkdtemp()	1312
mkfifo	2991, 3759
mkfifo()	1315, 3483
mkfifoat()	1315

mknod	
rationale for omission.....	3745
mknod().....	<b>1319</b> , 3483
mknodat().....	1319
mkstemp().....	1312, 1323
mktemp().....	3681
mktime().....	<b>1324</b> , 3508
ML.....	<b>8</b>
mlock().....	<b>1327</b>
mlockall().....	<b>1329</b> , 3588
MLR.....	<b>8</b>
mmap().....	<b>1331</b> , 3590-3592, 3594
MMU.....	3587
MM_.....	475
MM_ macros.....	<b>252</b>
MM_APPL.....	252, 884
MM_CONSOLE.....	252, 884
MM_ERROR.....	252, 885-886
MM_FIRM.....	252
mm_FIRM.....	884
MM_HALT.....	252, 885
MM_HARD.....	252, 884
MM_INFO.....	252, 885
MM_NOCON.....	253, 885
MM_NOMSG.....	252, 885
MM_NOSEV.....	252, 885
MM_NOTOK.....	252, 885
MM_NRECOV.....	252, 884
MM_NULLACT.....	252
MM_NULLLBL.....	252
MM_NULLMC.....	252, 884
MM_NULLSEV.....	252
MM_NULLTAG.....	252
MM_NULLTXT.....	252
MM_OK.....	252, 885
MM_OPSYS.....	252, 884
MM_PRINT.....	252, 884, 886
MM_RECOVER.....	252, 884
MM_SOFT.....	252, 884
MM_UTIL.....	252, 884
MM_WARNING.....	252, 885
mode.....	69
modem disconnect.....	3541
mode_t.....	401
modf().....	<b>1339</b>
modff().....	1339
modfl().....	1339
modify batch job request.....	2434
MON.....	<b>8</b>
monotonic clock.....	69, 3610
MON_.....	267
mon_decimal_point.....	155

mon_grouping .....	155
mon_thousands_sep .....	155
more .....	<b>2994</b> , 3758-3759
discard and refresh .....	3000
display position .....	3003
examine new file .....	3002
examine next file .....	3002
examine previous file .....	3002
go to beginning of file .....	3000
go to end-of-file .....	3000
go to tag .....	3002
help .....	2999
invoke editor .....	3002
mark position .....	3000
quit .....	3003
refresh the screen .....	3000
repeat search .....	3001
repeat search in reverse .....	3001
return to mark .....	3001
return to previous position .....	3001
scroll backward one half screenful .....	3000
scroll backward one line .....	2999
scroll backward one screenful .....	2999
scroll forward one half screenful .....	2999
scroll forward one line .....	2999
scroll forward one screenful .....	2999
search backward for pattern .....	3001
search forward for pattern .....	3001
skip forward one line .....	3000
MORECTL .....	368, 1049
MOREDATA .....	368, 1049
motion command .....	3221
mount point .....	70, 3488
mount() .....	3488
mounted file system .....	3488
move batch job request .....	2435
mprotect() .....	<b>1341</b> , 3591
MQ .....	475
mq .....	475
mq_close() .....	<b>1343</b>
mq_getattr() .....	<b>1344</b>
mq_notify() .....	<b>1346</b>
mq_open() .....	<b>1349</b> , 3579
MQ_OPEN_MAX .....	271, 2089, 3769
MQ_PRIO_MAX .....	271, 1355-1356, 2089, 3769
mq_receive() .....	<b>1352</b> , 3580
mq_send() .....	<b>1355</b> , 3580
mq_setattr() .....	<b>1357</b>
mq_timedreceive() .....	1352, 1359, 3611
mq_timedsend() .....	1355, 1360, 3611
mq_unlink() .....	<b>1361</b>
mrnd48() .....	747, 1363

MSG.....	8
msg.....	475
msg*().....	3576
msgctl().....	1364, 3576
msgget().....	1366, 3576
msgrcv().....	1368, 3576
msgsnd().....	1371, 3576
MSGVERB.....	178, 885-886
MSG_.....	475-476
msg_.....	476
MSG_ANY.....	368, 1048
MSG_BAND.....	368, 1048, 1745
MSG_CTRUNC.....	387
MSG_DONTROUTE.....	387
MSG_EOR.....	387, 1869, 1872, 1876, 1995, 1997
MSG_HIPRI.....	368, 1048, 1745
MSG_NOERROR.....	375, 1368-1369
MSG_NOSIGNAL.....	387, 1869, 1872, 1876
MSG_OOB.....	387, 1869, 1872, 1876
MSG_PEEK.....	387
msg_perm.....	502
MSG_TRUNC.....	387
MSG_WAITALL.....	387
msqid.....	502
MST7MDT.....	2176
msync().....	1374, 3591
MS.....	475
MS_ASYNC.....	372, 1333, 1374
MS_INVALIDATE.....	372, 1374-1375
MS_SYNC.....	372, 1333, 1374
multi-byte character.....	3540, 3542
multi-character collating element.....	70
multi-threaded library.....	70
multi-threaded process.....	70
multi-threaded program.....	70
multicast.....	533
multiple tasks.....	3751, 3758
munlock().....	1327, 1377
munlockall().....	1329, 1378
munmap().....	1379, 3590-3592, 3594, 3597
mutex.....	70, 3628
attributes.....	1673
extended attributes.....	3631
initialization.....	3643
initialization attributes.....	1672
performance.....	1673
MUXID_ALL.....	368, 1147-1148
MUXID_R.....	477
mv.....	3006, 3704, 3759
MX.....	9
MXX.....	9
M_.....	288, 477

M_E .....	288
M_LN .....	288
M_LOG10E .....	288
M_LOG2E .....	288
M_PI .....	288
M_SQRT1_2 .....	289
M_SQRT2 .....	289
name .....	71
name information .....	1052
name space .....	473, 3557
name space pollution .....	3557-3558
named STREAM .....	71
NAME_MAX .....	112, 231, 274, 484, 899, 2457, 3086, 3698, 3769
NaN .....	71, 247
NAN .....	289
NaN .....	602, 604, 608, 614
NAN .....	909
NaN .....	909
NAN .....	992
NaN .....	992
NaN arguments .....	3509
mathematical functions .....	118
nan() .....	1381
nanf() .....	1381
nanl() .....	1381
nanosleep() .....	1382, 3608, 3610-3611, 3756
native implementation .....	3488
native language .....	71
NCCS .....	414
NDEBUG .....	223, 480, 607
nearbyint() .....	1384
nearbyintf() .....	1384
nearbyintl() .....	1384
negative response .....	71
negative_sign .....	155
netent .....	301
network .....	71
network address .....	71
network byte order .....	72, 110, 3504
network interfaces .....	524
newgrp .....	2334, 2357, 3012, 3760
newline character .....	72
newlocale() .....	1385
news .....	
rationale for omission .....	3745
NEW_TIME .....	455, 769-770
nextafter() .....	1388
nextafterf() .....	1388
nextafterl() .....	1388
nexttoward() .....	1388
nexttowardf() .....	1388
nexttowardl() .....	1388

nftw()	1390
NGROUPS_MAX	275, 1038, 2089, 3015, 3469, 3491, 3698, 3763, 3769
nice	3016, 3758
nice value	72, 3488
nice()	1394, 3602
Ninth Edition UNIX	2526, 2665, 3105
NI_DGRAM	302
NI_NAMEREQD	302
NI_NOFQDN	302
NI_NUMERICHOST	302
NI_NUMERICSCOPE	302
NI_NUMERICSERV	302
nl	3020
NLDLY	415
nlink_t	401
NLn	415
NLSPATH	175, 648
NL	475
NL_ARGMAX	281, 906, 945, 989, 999
NL_CAT_LOCALE	311, 648
nl_langinfo()	1396, 3757
nl_langinfo_l()	1396
NL_LANGMAX	281
NL_MSGMAX	281
NL_SETD	311
NL_SETMAX	282
NL_TEXTMAX	282
nm	3024, 3760
noclobber option	3055, 3723
NOEXPR	267
NOFLSH	417
nohup	790, 3029, 3758
non-blocking	72
non-canonical mode input processing	202, 3540
non-local jumps	1973
non-printable	2679, 3212, 3273, 3498
non-spacing characters	72
non-volatile storage	970
normative references	3470
NOSTR	267
NQS	3737
rand48()	747, 1398
ntohl()	1113, 1399
ntohs()	1113, 1399
NUL	73
NULL	345, 424, 697, 730, 739, 1336, 1772
null byte	73
null pointer	73
null string	73
null wide-character code	73
number-sign	73
numerical limits	280

NUM_EMPL .....	1111
NZERO .....	282, 1067, 1394
n_ .....	475
n_cs_precedes .....	156
n_sep_by_space .....	156
n_sign_posn .....	156
OB .....	9
object file .....	73, 3024
obsolescent .....	3471
rationale .....	3471
OCRNL .....	415
octet .....	73
od .....	3033, 3758, 3760
OF .....	9
OFDEL .....	415
offset maximum .....	74
off_t .....	401
OFILL .....	415
OH .....	9
OLD_TIME .....	455, 769-770
ONLCR .....	415
ONLRET .....	415
ONOCR .....	415
opaque address .....	74
open a file .....	1406
open a named semaphore .....	1847
open a shared memory object .....	1926
open file .....	74
open file description .....	74, 3488
open file descriptors .....	3724
for reading and writing .....	2353
open mode .....	2686
open() .....	1400, 3483, 3539, 3591-3594, 3704, 3754
openat() .....	1400, 1412
opendir() .....	835, 1413, 3755
openlog() .....	692, 1414, 3760
OPEN_MAX .....	271, 328, 765, 822, 835, 979, 1349, 1452, 1455, 2089, 2154, 3469, 3698
.....	3769-3770
open_memstream() .....	1410
open_wmemstream() .....	1410
operand .....	74
operator .....	74
OPOST .....	415
optarg .....	1056, 1415
opterr .....	1056, 1415
optind .....	1056, 1415
option .....	75
ADV .....	7
BE .....	7
CD .....	7
CPT .....	7
FD .....	7

FR.....	7
FSC .....	8
IP6.....	8
MC1 .....	8
ML .....	8
MLR.....	8
MON .....	8
MSG.....	8
MX .....	9
MXX .....	9
PIO.....	9
PS .....	9
RPI .....	9
RPP .....	10
RS.....	10
SD .....	10
SHM .....	10
SIO .....	10
SPN.....	10
SS .....	10
TCT.....	10
TEF.....	11
TPL.....	11
TPP.....	11
TPS.....	11
TRC.....	11
TRI .....	11
TRL .....	11
TSA.....	12
TSH.....	12
TSP.....	12
TSS.....	12
TYM.....	12
UP .....	12
UU .....	12
XSR.....	13
option definitions .....	3737
option-argument .....	75
optional behavior .....	3771
options .....	3652
shell and utilities .....	27
system interfaces .....	27
optopt.....	1056, 1059, 1415
optstring .....	1059
OR lists.....	2361, 3730
ordinary identifiers .....	2519
ORD_CHAR.....	192
orientation .....	75
orphaned process group .....	75, 556, 3489, 3572
output device .....	198, 3536
output file descriptor	
duplication .....	3724



output mode .....	3542
output processing.....	3541
Output_Path.....	2428
overflow conditions.....	3676
overflow in dumping trace streams .....	3676
overflow in trace streams.....	3676
O_ .....	477
O_ constants	
defined in <fcntl.h> .....	238-239
used in open() .....	1400
used in posix_openpt() .....	1442
O_ACCMODE .....	239, 818
O_APPEND.....	239, 503, 591, 728, 833, 1400, 2300
O_CLOEXEC.....	238, 1400, 3594
O_CREAT.....	238, 712, 1349-1350, 1400, 1838, 1845, 1924-1925, 1927
O_DIRECTORY .....	238, 1401
O_DSYNC .....	239, 582, 1401, 1764, 2301
O_EXCL.....	239, 1350, 1401, 1845, 1924-1925
O_EXEC .....	239, 1400
O_NDELAY .....	2306
O_NOCTTY.....	239, 1401, 1442
O_NOFOLLOW.....	239, 1401
O_NONBLOCK.....	239, 483, 818, 1350, 1401
O_RDONLY .....	239, 735, 1349, 1400, 1924, 1927
O_RDWR.....	239, 808, 1261, 1349, 1400, 1442, 1924, 1927
O_RSYNC.....	239, 1401, 1764
O_SEARCH.....	239, 1235, 1308, 1315, 1320, 1400, 1403, 1775, 1809, 2085, 2188
O_SYNC.....	239, 582, 1402, 1764, 2301
O_TRUNC.....	239, 712, 1402, 1925, 1927
O_TTY_INIT .....	199, 239, 1402
O_WRONLY.....	239, 712, 808, 1261, 1349, 1400
pack	
rationale for omission.....	3745
page.....	75, 3489, 3590, 3594
page size .....	75
PAGESIZE .....	271, 505, 1327, 1560, 2089, 3590, 3635, 3769
PAGE_SIZE .....	272, 2089
paginators	
more .....	2994
parallel I/O .....	3635
parameter .....	76, 3541, 3709
expansion .....	2344, 3717
positional.....	3709
special .....	3709
parameters and variables.....	2339
PARENB.....	417
parent directory .....	76, 3489
parent process .....	76
parent process ID .....	76
PARMRK .....	415
PARODD .....	417

passwd	
rationale for omission.....	3745
passwd file.....	3489
paste .....	3041, 3758
patch .....	3045, 3759-3760
application.....	3048
file format .....	3047
filename determination .....	3048
PATH.....	178, 697, 791, 3527
path prefix .....	77
pathchk .....	3052, 3759
pathconf().....	899, 1416, 3483, 3697, 3753, 3755
pathname.....	76, 3489
component.....	77
expansion .....	2350, 3722
incomplete .....	3483
resolution .....	111, 2321, 3506
variable values.....	273
pathname manipulation	
basename.....	2508
dirname.....	2656
pathchk .....	3052
PATH_MAX .....	274, 282, 484, 899, 2324, 3091, 3190, 3698, 3769
pattern.....	77
filename expansion .....	3736
for filename expansion .....	2373
scanning and processing language.....	2472
pattern matching .....	2786, 3067, 3343, 3359
definition .....	2372
in case statements.....	2362
in shell variables.....	2345
multiple character .....	3735
multiple characters.....	2373
notation.....	2372, 2792, 3084, 3735
single character.....	2372, 3735
pause() .....	1417, 3571, 3574, 3754
pax.....	3057, 3759-3760
archive character set encoding/decoding .....	3090
cpio file data.....	3080
cpio filename.....	3080
cpio header .....	3079
cpio interchange format .....	3078
cpio special entries .....	3081
extended header .....	3071
extended header file times.....	3074
extended header keyword precedence .....	3074
list mode format specifications .....	3065
ustar format.....	3075
ustar interchange format.....	3075
pcat	
rationale for omission.....	3745
pclose() .....	1418, 3760

pd_.....	475
PENDIN.....	477
pending error .....	3651
per-thread errno .....	3564
performance enhancements.....	3750
period.....	77
permissions .....	78
perror() .....	<b>1420</b>
persistence.....	78
persistent connection (I_PLINK).....	<b>1148</b>
PF_.....	476
pg	
rationale for omission.....	3745
physical write.....	970
ph_.....	475
PID_MAX .....	3678
pid_t .....	401
PIO.....	<b>9</b>
pipe.....	78, 896, 1406, 2304, 3484, 3490
pipe() .....	<b>1422</b> , 3571, 3679, 3754
pipelines .....	2358, 3728
PIPE_BUF .....	274, 899, 2301, 2304, 3698, 3769
PIPE_MAX .....	2306
plain characters.....	2029
PM_STR .....	267
pointer to a function .....	492
pole error .....	117
POLL .....	475
poll() .....	<b>1425</b>
POLLERR .....	312, 1425
pollfd .....	<b>312</b>
POLLHUP .....	312, 1425
POLLIN .....	312, 1425
polling.....	78
POLLNVAL.....	312, 1426
POLLOUT .....	312, 1425
POLLPRI.....	312, 1425
POLLRDBAND .....	312, 1425
POLLRDNORM .....	312, 1425
POLLWRBAND.....	312, 1425
POLLWRNORM.....	312, 1425
POLL_.....	475
POLL_ERR .....	336
POLL_HUP .....	336
POLL_IN .....	336
POLL_MSG .....	336
POLL_OUT .....	336
POLL_PRI.....	336
popen().....	<b>1429</b> , 3757, 3760-3761
portability.....	3472
portability codes.....	3472
portable character set.....	78, 125, 2965, 3510

portable filename.....	79
portable filename character set .....	79, 3490
positional parameter .....	79, 2339, 3709
positive_sign .....	155
POSIX conformance .....	15
POSIX locale.....	136, 3514
POSIX shell and utilities.....	19
POSIX system interfaces	
conformance.....	17
POSIX.1 symbols .....	472, 3556
POSIX.13.....	3594
POSIX2_BC_BASE_MAX.....	2323-2324, 3763
POSIX2_BC_DIM_MAX .....	2323-2324, 3763
POSIX2_BC_SCALE_MAX.....	2323-2324, 3763
POSIX2_BC_STRING_MAX.....	2323-2324, 3763
POSIX2_CHAR_TERM.....	19, 27, 3762
POSIX2_COLL_WEIGHTS_MAX .....	2323-2324, 3763
POSIX2_C_BIND .....	3699, 3761
POSIX2_C_DEV .....	19, 27, 3699, 3761-3762
POSIX2_EXPR_NEST_MAX .....	2323-2324, 3763
POSIX2_FORT_DEV .....	19, 27, 3699, 3762
POSIX2_FORT_RUN .....	19, 27, 3699, 3762
POSIX2_LINE_MAX.....	2323, 2325, 3763
POSIX2_LOCALEDEF.....	19, 27, 3699, 3759, 3762
POSIX2_PBS.....	19, 28, 3762
POSIX2_PBS_ACCOUNTING .....	19, 28, 3762
POSIX2_PBS_CHECKPOINT .....	28, 3762
POSIX2_PBS_LOCATE.....	19, 28, 3762
POSIX2_PBS_MESSAGE.....	19, 28, 3762
POSIX2_PBS_TRACK.....	19, 28, 3762
POSIX2_RE_DUP_MAX.....	3763
POSIX2_SW_DEV .....	19, 28, 3699, 3761
POSIX2_SYMLINKS .....	899, 2325, 3699
POSIX2_UPE.....	19, 28, 3699, 3761-3762
POSIX2_VERSION .....	3763
POSIX_.....	474
posix_.....	474
POSIX_ALLOC_SIZE_MIN .....	274, 899, 3577
POSIX_ASYNCHRONOUS_IO.....	3775
POSIX_BARRIERS .....	3775
POSIX_CLOCK_SELECTION .....	3776
POSIX_C_LANG_JUMP .....	3775
POSIX_C_LANG_MATH.....	3775
POSIX_C_LANG_SUPPORT .....	3776
POSIX_C_LANG_SUPPORT_R .....	3776
POSIX_C_LANG_WIDE_CHAR .....	3776
POSIX_C_LANG_WIDE_CHAR_EXT.....	3776
POSIX_C_LIB_EXT .....	3776
POSIX_DEVICE_IO .....	3776
POSIX_DEVICE_IO_EXT.....	3776
POSIX_DEVICE_SPECIFIC .....	3776
POSIX_DEVICE_SPECIFIC_R.....	3777

POSIX_DYNAMIC_LINKING .....	3777
posix_fadvise() .....	<b>1432</b> , 3577
POSIX_FADV_DONTNEED .....	240, 1432, 3577
POSIX_FADV_NOREUSE .....	240, 1432, 3577
POSIX_FADV_NORMAL .....	240, 1432
POSIX_FADV_RANDOM .....	240, 1432, 3577
POSIX_FADV_SEQUENTIAL .....	240, 1432, 3577
POSIX_FADV_WILLNEED .....	240, 1432, 3577
posix_fallocate() .....	<b>1434</b>
POSIX_FD_MGMT .....	3777
POSIX_FIFO .....	3777
POSIX_FIFO_FD .....	3777
POSIX_FILE_ATTRIBUTES .....	3777
POSIX_FILE_ATTRIBUTES_FD .....	3777
POSIX_FILE_LOCKING .....	3777
POSIX_FILE_SYSTEM .....	3777
POSIX_FILE_SYSTEM_EXT .....	3777
POSIX_FILE_SYSTEM_FD .....	3777
POSIX_FILE_SYSTEM_GLOB .....	3777
POSIX_FILE_SYSTEM_R .....	3777
POSIX_I18N .....	3777
POSIX_JOB_CONTROL .....	3777
posix_madvise() .....	<b>1436</b> , 3577
POSIX_MADV_DONTNEED .....	372, 1436, 3577
POSIX_MADV_NORMAL .....	372, 1436
POSIX_MADV_RANDOM .....	372, 1436, 3577
POSIX_MADV_SEQUENTIAL .....	372, 1436, 3577
POSIX_MADV_WILLNEED .....	373, 1436, 3577
POSIX_MAPPED_FILES .....	3777
posix_memalign() .....	<b>1440</b>
POSIX_MEMORY_PROTECTION .....	3777
posix_mem_offset() .....	<b>1438</b> , 3594-3595
POSIX_MULTI_CONCURRENT_LOCALES .....	3777
POSIX_MULTI_PROCESS .....	3778
POSIX_MULTI_PROCESS_FD .....	3778
POSIX_NETWORKING .....	3778
posix_openpt() .....	<b>1442</b>
POSIX_PIPE .....	3778
POSIX_REALTIME_SIGNALS .....	3778
POSIX_REC_INCR_XFER_SIZE .....	274, 899, 3578
POSIX_REC_MAX_XFER_SIZE .....	274, 899, 3578
POSIX_REC_MIN_XFER_SIZE .....	274, 899, 3578
POSIX_REC_XFER_ALIGN .....	274, 899, 3577
POSIX_REGEX .....	3778
POSIX_RE_DUP_MAX .....	2323, 2325
POSIX_ROBUST_MUTEXES .....	3778
POSIX_RW_LOCKS .....	3778
POSIX_SEMAPHORES .....	3778
POSIX_SHELL_FUNC .....	3778
POSIX_SIGNALS .....	3778
POSIX_SIGNALS_EXT .....	3778
POSIX_SIGNAL_JUMP .....	3778

POSIX_SINGLE_PROCESS .....	3779
posix_spawn() .....	<b>1444</b> , 3682, 3754
posix_spawnattr_destroy() .....	<b>1460</b>
posix_spawnattr_getflags() .....	<b>1462</b>
posix_spawnattr_getpgroup() .....	<b>1464</b>
posix_spawnattr_getschedparam() .....	<b>1466</b>
posix_spawnattr_getschedpolicy() .....	<b>1468</b>
posix_spawnattr_getsigdefault() .....	<b>1470</b>
posix_spawnattr_getsigmask() .....	<b>1472</b>
posix_spawnattr_init() .....	1460, 1474
posix_spawnattr_setflags() .....	1462, 1475
posix_spawnattr_setpgroup() .....	1464, 1476
posix_spawnattr_setschedparam() .....	1466, 1477
posix_spawnattr_setschedpolicy() .....	1468, 1478
posix_spawnattr_setsigdefault() .....	1470, 1479
posix_spawnattr_setsigmask() .....	1472, 1480
posix_spawnp() .....	1444, 1481, 3682, 3754
posix_spawn_file_actions_addclose() .....	<b>1452</b>
posix_spawn_file_actions_adddup2() .....	<b>1455</b>
posix_spawn_file_actions_addopen() .....	1452, 1457
posix_spawn_file_actions_destroy() .....	<b>1458</b>
posix_spawn_file_actions_init() .....	1458
POSIX_SPAWN_RESETIDS .....	1445, 1462
POSIX_SPAWN_SETPGROUP .....	1445, 1462, 1464
POSIX_SPAWN_SETSCHEDPARAM .....	1462, 1466
POSIX_SPAWN_SETSCHEDULER .....	1445, 1462, 1466, 1468
POSIX_SPAWN_SETSIGDEF .....	1446, 1462, 1470
POSIX_SPAWN_SETSIGMASK .....	1462, 1472
POSIX_SPIN_LOCKS .....	3779
POSIX_SYMBOLIC_LINKS .....	3779
POSIX_SYMBOLIC_LINKS_FD .....	3779
POSIX_SYSTEM_DATABASE .....	3779
POSIX_SYSTEM_DATABASE_R .....	3779
POSIX_THREADS_BASE .....	3779
POSIX_THREADS_EXT .....	3779
POSIX_TIMERS .....	3779
POSIX_TRACE_ADD_EVENTSET .....	1521
POSIX_TRACE_ALL_EVENTS .....	1514
POSIX_TRACE_APPEND .....	1487, 1507
posix_trace_attr_destroy() .....	<b>1482</b>
posix_trace_attr_getclockres() .....	<b>1484</b>
posix_trace_attr_getcreatetime() .....	1484
posix_trace_attr_getgenversion() .....	1484
posix_trace_attr_getinherited() .....	<b>1486</b>
posix_trace_attr_getlogfullpolicy() .....	1486
posix_trace_attr_getlogsize() .....	<b>1489</b>
posix_trace_attr_getmaxdatasize() .....	1489
posix_trace_attr_getmaxsystemeventsz() .....	1489
posix_trace_attr_getmaxusereventsiz() .....	1489
posix_trace_attr_getname() .....	1484, 1492
posix_trace_attr_getstreamfullpolicy() .....	1486, 1493
posix_trace_attr_getstreamsize() .....	1489, 1494

posix_trace_attr_init()	1482, 1495
posix_trace_attr_setinherited()	1486, 1496
posix_trace_attr_setlogfullpolicy()	1486, 1496
posix_trace_attr_setlogsize()	1489, 1497
posix_trace_attr_setmaxdatasize()	1489, 1497
posix_trace_attr_setname()	1484, 1498
posix_trace_attr_setstreamfullpolicy()	1486, 1499
posix_trace_attr_setstreamsize()	1489, 1500
posix_trace_clear()	<b>1501</b>
posix_trace_close()	<b>1503</b>
POSIX_TRACE_CLOSE_FOR_CHILD	1486
posix_trace_create()	<b>1505</b>
posix_trace_create_withlog()	1505
POSIX_TRACE_ERROR trace event	543
posix_trace_event()	<b>1509</b>
posix_trace_eventid_equal()	<b>1511</b>
posix_trace_eventid_get_name()	1511
posix_trace_eventid_open()	1509, 1513, 3671
posix_trace_eventset_add()	<b>1514</b>
posix_trace_eventset_del()	1514
posix_trace_eventset_empty()	1514
posix_trace_eventset_fill()	1514
posix_trace_eventset_ismember()	1514
posix_trace_eventtypelist_getnext_id()	<b>1516</b>
posix_trace_eventtypelist_rewind()	1516
posix_trace_event_info	540
POSIX_TRACE_FILTER trace event	543, 1521
POSIX_TRACE_FLUSH	1487
posix_trace_flush()	1505, 1518
POSIX_TRACE_FLUSHING	539
POSIX_TRACE_FULL	538-540
posix_trace_getnext_event()	<b>1524</b>
posix_trace_get_attr()	<b>1519</b>
posix_trace_get_filter()	<b>1521</b>
posix_trace_get_status()	1519, 1523
POSIX_TRACE_INHERITED	1486
POSIX_TRACE_LOOP	539, 1486-1487, 1507, 3676
POSIX_TRACE_NOT_FLUSHING	540
POSIX_TRACE_NOT_FULL	538-540
POSIX_TRACE_NOT_FULL	1501
POSIX_TRACE_NOT_TRUNCATED	541, 1525
POSIX_TRACE_NO_OVERRUN	539-540, 1519
posix_trace_open()	1503, 1527
POSIX_TRACE_OVERFLOW trace event	543
POSIX_TRACE_OVERRUN	539-540
POSIX_TRACE_RESUME trace event	543
posix_trace_rewind()	1503, 1527
POSIX_TRACE_RUNNING	538-539, 1530
POSIX_TRACE_SET_EVENTSET	1521
posix_trace_set_filter()	1521, 1528
posix_trace_shutdown()	1505, 1529
POSIX_TRACE_START trace event	543, 1530

posix_trace_start()	1530
posix_trace_status_info	538
POSIX_TRACE_STOP trace event	543, 1530
posix_trace_stop()	1530
POSIX_TRACE_SUB_EVENTSET	1521
POSIX_TRACE_SUSPENDED	538-539, 1530
POSIX_TRACE_SYSTEM_EVENTS	1514
posix_trace_timedgetnext_event()	1524, 1532
posix_trace_trid_eventid_open()	1511, 1533
POSIX_TRACE_TRUNCATED_READ	541, 1525
POSIX_TRACE_TRUNCATED_RECORD	541, 1525
posix_trace_trygetnext_event()	1524, 1534
POSIX_TRACE_UNTIL_FULL	539, 1486-1487, 1506
POSIX_TRACE_USER_EVENT_MAX	1509
POSIX_TRACE_WOPID_EVENTS	1514
POSIX_TYPED_MEM_ALLOCATE	373, 1331-1332, 1438, 1535, 1537
POSIX_TYPED_MEM_ALLOCATE_CONTIG	373, 1331-1332, 1438, 1535, 1537
posix_typed_mem_get_info()	1535, 3594
posix_typed_mem_info	373
POSIX_TYPED_MEM_MAP_ALLOCATABLE	373, 1379, 1537
posix_typed_mem_open()	1537, 3594
POSIX_USER_GROUPS	3779
POSIX_USER_GROUPS_R	3779
POSIX_VERSION	3769
POSIX_WIDE_CHAR_DEVICE_IO	3779
post-mortem filtering of trace event types	3673
pow()	1540
powf()	1540
powl()	1540
pr	3096, 3758, 3760
pread()	1543, 1763, 3636
preallocation	79
predefined stream	
standard error	498
standard input	498
standard output	498
preempted process (or thread)	79
preempted thread	1612
previous job	79
PRI	477
print-related commands	
fold	2795
lp	2907
pr	3096
printable character	80
printable file	80
printf	3101, 3757-3758
printf()	906, 1544
printing	3752
priority	80, 500
Priority	2429



priority	
band.....	80
inversion .....	80
scheduling .....	80
priority-based scheduling.....	80
PRIO_.....	475
PRIO_ constants	
defined in <sys/resource.h> .....	377
PRIO_INHERIT .....	1668
PRIO_PGRP .....	377, 1067
PRIO_PROCESS .....	377, 1067
PRIO_USER.....	377, 1067
privilege.....	81, 2986, 3018, 3501
process .....	81
attributes.....	2317
concurrent execution .....	896
ID .....	82, 2317
ID reuse .....	113, 3507
ID, 1 .....	556
ID, rationale .....	3678
lifetime .....	82, 3490
memory locking.....	82
scheduling .....	506, 3600, 3754
setting real and effective user IDs.....	1909
single-threaded .....	896
termination.....	82, 3490
virtual time.....	83
process creation .....	896
process group.....	81, 3538
concepts in job control.....	3484
ID .....	81, 2317, 3484, 3538
leader.....	81
lifetime .....	81, 3538
orphaned .....	556, 3489, 3572
termios .....	200
process group ID .....	1063, 1901, 1913
process lifetime .....	1220
process management .....	3750, 3754
process shared memory.....	1673
process status report .....	3112
process synchronization.....	1673
process termination.....	555
process-to-process communication .....	82
prof	
rationale for omission.....	3745
profiling .....	3761
program .....	83
programming manipulation.....	3667
prompting.....	3715
protocol.....	83, 3650
protoent .....	301
PROT_.....	475

PROT_EXEC.....	372, 1332, 1341
PROT_NONE .....	372, 505, 1331-1332, 1341
PROT_READ.....	372, 1332, 1341
PROT_READ constants	
in <sys/mman.h> .....	372
PROT_WRITE.....	372, 1332-1333, 1336, 1341
prs .....	3107
PS .....	9
ps.....	3112, 3758, 3760
pselect() .....	1545
pseudo-random sequence generation functions .....	1760
pseudo-terminal .....	83
psiginfo() .....	1550
psignal() .....	1550
PST8PDT .....	2176
ps_ .....	475
pthread.....	3674
PTHREAD_ .....	475
pthread_ .....	475
pthread_atfork() .....	1552
pthread_attr_destroy() .....	1555
pthread_attr_getdetachstate() .....	1558
pthread_attr_getguardsize() .....	1560, 3635
pthread_attr_getinheritsched() .....	1563
pthread_attr_getschedparam() .....	1565
pthread_attr_getschedpolicy() .....	1567
pthread_attr_getscope() .....	1569
pthread_attr_getstack() .....	1571
pthread_attr_getstackaddr() .....	3681
pthread_attr_getstacksize() .....	1574
pthread_attr_init() .....	1555, 1576
pthread_attr_setdetachstate() .....	1558, 1577
pthread_attr_setguardsize() .....	1560, 1578, 3635
pthread_attr_setinheritsched() .....	1563, 1579
pthread_attr_setschedparam() .....	1565, 1580
pthread_attr_setschedpolicy() .....	1567, 1581
pthread_attr_setscope() .....	1569, 1582
pthread_attr_setstack() .....	1571, 1583
pthread_attr_setstackaddr() .....	3681
pthread_attr_setstacksize() .....	1574, 1584
pthread_barrierattr_destroy() .....	1589
pthread_barrierattr_getpshared() .....	1591
pthread_barrierattr_init() .....	1589, 1593
pthread_barrierattr_setpshared() .....	1591, 1594
pthread_barrier_destroy() .....	1585
pthread_barrier_init() .....	1585
PTHREAD_BARRIER_SERIAL_THREAD .....	314, 1587, 3626
pthread_barrier_wait() .....	1587, 3627, 3647
pthread_cancel() .....	1595
PTHREAD_CANCELED.....	314, 521, 1632
PTHREAD_CANCEL_ASYNCHRONOUS .....	314, 517, 1719
PTHREAD_CANCEL_DEFERRED .....	314, 517, 521, 784, 1609, 1719

PTHREAD_CANCEL_DISABLE .....	314, 517, 521, 1719
PTHREAD_CANCEL_ENABLE .....	314, 517, 521, 1719
PTHREAD_CANCEL_ENABLED .....	784
pthread_cleanup_pop() .....	<b>1597</b>
pthread_cleanup_push() .....	1597
pthread_condattr_destroy() .....	<b>1616</b>
pthread_condattr_getclock() .....	<b>1618</b>
pthread_condattr_getpshared() .....	<b>1620</b>
pthread_condattr_init() .....	1616, 1622
pthread_condattr_setclock() .....	1618, 1623
pthread_condattr_setpshared() .....	1620, 1624
pthread_cond_broadcast() .....	<b>1602</b>
pthread_cond_destroy() .....	<b>1605</b>
pthread_cond_init() .....	1605, 3623
PTHREAD_COND_INITIALIZER .....	314, 1605
pthread_cond_signal() .....	1602, 1608
pthread_cond_timedwait() .....	<b>1609</b> , 3565, 3610, 3632, 3764
pthread_cond_wait() .....	1609, 3565, 3582, 3632
pthread_create() .....	<b>1625</b> , 3623-3624
PTHREAD_CREATE_DETACHED .....	314, 490, 1558, 3645
PTHREAD_CREATE_JOINABLE .....	314, 490, 784, 1558, 1642
PTHREAD_DESTRUCTOR_ITERATIONS .....	272, 1639, 1644, 2089, 3770
pthread_detach() .....	<b>1628</b> , 3645
pthread_equal() .....	<b>1630</b>
pthread_exit() .....	<b>1631</b>
PTHREAD_EXPLICIT_SCHED .....	314, 1563
pthread_getconcurrency() .....	<b>1633</b> , 3635
pthread_getcpuclockid() .....	<b>1635</b> , 3613-3614
pthread_getschedparam() .....	<b>1636</b>
pthread_getspecific() .....	<b>1639</b>
PTHREAD_INHERIT_SCHED .....	314, 1563
pthread_join() .....	<b>1641</b> , 3565, 3645
PTHREAD_KEYS_MAX .....	272, 1644, 2089, 3770
pthread_key_create() .....	<b>1644</b> , 3626
pthread_key_delete() .....	<b>1647</b>
pthread_kill() .....	<b>1649</b>
pthread_mutexattr_destroy() .....	<b>1672</b>
pthread_mutexattr_getprioceiling() .....	<b>1677</b>
pthread_mutexattr_getprotocol() .....	<b>1679</b>
pthread_mutexattr_getpshared() .....	<b>1682</b>
pthread_mutexattr_getrobust() .....	<b>1684</b>
pthread_mutexattr_gettype() .....	<b>1686</b> , 3633
pthread_mutexattr_init() .....	1672, 1688
pthread_mutexattr_setprioceiling() .....	1677, 1689
pthread_mutexattr_setprotocol() .....	1679, 1690
pthread_mutexattr_setpshared() .....	1682, 1691
pthread_mutexattr_setrobust() .....	1684, 1692
pthread_mutexattr_settype() .....	1686, 1693, 3633
pthread_mutex_consistent() .....	<b>1651</b>
PTHREAD_MUTEX_DEFAULT .....	314, 1663, 1686, 3632
pthread_mutex_destroy() .....	<b>1653</b>
PTHREAD_MUTEX_ERRORCHECK .....	314, 1659, 1663, 1686, 3632

pthread_mutex_getprioceiling()	1659
pthread_mutex_init()	1653, 1662, 3623
PTHREAD_MUTEX_INITIALIZER	314, 1653
pthread_mutex_lock()	1663, 3565, 3632, 3646-3647
PTHREAD_MUTEX_NORMAL	314, 1663, 1686, 3632
PTHREAD_MUTEX_RECURSIVE	111, 314, 1663, 1686, 3632
PTHREAD_MUTEX_ROBUST	1684
pthread_mutex_setprioceiling()	1659, 1667
PTHREAD_MUTEX_STALLED	1684
pthread_mutex_timedlock()	1668, 3611
pthread_mutex_trylock()	1663, 1671, 3632
pthread_mutex_unlock()	1663, 1671, 3632
pthread_once()	1694
PTHREAD_ONCE_INIT	314, 1694
PTHREAD_PRIO_INHERIT	314, 1679
PTHREAD_PRIO_NONE	314, 1659, 1679
PTHREAD_PRIO_PROTECT	314, 1664, 1679
PTHREAD_PROCESS_PRIVATE	314, 1673, 3634
PTHREAD_PROCESS_SHARED	314, 1591, 1620, 1673, 1682, 1714, 1730, 3634
pthread_rwlockattr_destroy()	1712, 3634
pthread_rwlockattr_getpshared()	1714, 3634
pthread_rwlockattr_init()	1712, 1716, 3633
pthread_rwlockattr_setpshared()	1714, 1717, 3634
pthread_rwlock_destroy()	1696
pthread_rwlock_init()	1696, 3634
PTHREAD_RWLOCK_INITIALIZER	314, 3634
pthread_rwlock_rdlock()	1699, 3634
pthread_rwlock_t	3633
pthread_rwlock_timedrdlock()	1702
pthread_rwlock_timedwrlock()	1704
pthread_rwlock_tryrdlock()	1699, 1706, 3634
pthread_rwlock_trywrlock()	1707, 3634
pthread_rwlock_unlock()	1709, 3634, 3648
pthread_rwlock_wrlock()	1707, 1711, 3634
PTHREAD_SCOPE_PROCESS	314, 515, 1569
PTHREAD_SCOPE_SYSTEM	314, 515, 1569
pthread_self()	1718, 3625
pthread_setcancelstate()	1719
pthread_setcanceltype()	1719
pthread_setconcurrency()	1633, 1721, 3634-3635
pthread_setprio()	3643
pthread_setschedparam()	1636, 1722, 3643
pthread_setschedprio()	1723
pthread_setspecific()	1639, 1725, 3625
pthread_sigmask()	1726
pthread_spin_destroy()	1730
pthread_spin_init()	1730
pthread_spin_lock()	1732, 3627, 3647
pthread_spin_trylock()	1732, 3627
pthread_spin_unlock()	1734
PTHREAD_STACK_MIN	272, 1571, 1574, 2089, 3770
pthread_testcancel()	1719, 1736

PTHREAD_THREADS_MAX .....	272, 1625, 2089, 3770
PTRDIFF_MAX .....	351
PTRDIFF_MIN .....	351
ptsname() .....	<b>1737</b>
public locale .....	2892
putc() .....	<b>1739</b> , 3637, 3755
putchar() .....	<b>1741</b> , 3755
putchar_unlocked() .....	1009, 1742
putc_unlocked() .....	1009, 1740
putenv() .....	<b>1743</b>
putmsg() .....	<b>1745</b>
putpmsg() .....	1745
puts() .....	<b>1749</b>
pututxline() .....	769, 1751
putwc() .....	<b>1752</b>
putwchar() .....	<b>1753</b>
PWD .....	<b>178</b>
pwd .....	2334, 2357, 3119, 3759
pwrite() .....	<b>1754</b> , 2300, 3636
pw_ .....	475
p_ .....	475
P_ .....	476
P_ALL .....	408, 2226
p_cs_precedes .....	156
P_PGID .....	408, 2226
P_PID .....	408, 2226
p_sep_by_space .....	156
p_sign_posn .....	156
P_tmpdir .....	355
qalter .....	<b>3122</b>
qdel .....	<b>3132</b>
qhold .....	<b>3135</b>
qmove .....	<b>3138</b>
qmsg .....	<b>3141</b>
qrerun .....	<b>3144</b>
qrls .....	<b>3147</b>
qselect .....	<b>3150</b>
qsig .....	<b>3159</b>
qsort() .....	<b>1755</b>
qstat .....	<b>3162</b>
qsub .....	<b>3167</b>
queue a signal to a process .....	1971
queue batch job request .....	2435
queuing of waiting threads .....	3648
quiet NaN .....	247
quote removal .....	2350, 3722
QUOTED_CHAR .....	192
quoting .....	2336, 3706
radix character .....	83
RADIXCHAR .....	267
raise() .....	<b>1757</b>
rand() .....	<b>1759</b> , 3646

random()	1134, 1762
RAND_MAX	358, 1759
rand_r()	1759
range error	118
result overflows	118
result underflows	118
RCS	
rationale for omission	3745
RE	
bracket expression	3531
grammar	3535
read	2334, 2357, 3180, 3705, 3757
read lock	3633
read()	1763, 3484, 3539-3540, 3562, 3571-3572, 3574, 3584-3586, 3590-3591, 3635
	3646, 3678, 3755
read-only file system	83
read-write attribute	3633
read-write lock	83, 3633
readdir()	1770, 3755
readdir_r()	1770
reading an active trace stream	3676
reading data	3540
readlink()	1775, 3754
readlinkat()	1775
readonly	2394
readv()	1778
real group ID	84, 2317
real time	84
real user ID	84, 571, 1219, 2317
realloc()	1780
realpath()	1782, 3754
realtime	22
REALTIME	296, 827, 1327, 1329, 1343-1344, 1346, 1349, 1352, 1355, 1357, 1361, 1825-1829
	1831, 1924, 1929
realtime	3577
realtime signal delivery	3568
realtime signal extension	84
realtime signal generation	3568
realtime signals	3582
REALTIME THREADS	24
realtime threads	24
REALTIME THREADS	1563, 1567, 1569, 1636, 1659, 1677, 1679, 1723
record	84
recv()	1785
recvfrom()	1788
recvmsg()	1791
red	
rationale for omission	3745
redirect input	2351, 3724
redirect output	2351, 3724
redirection	84, 2350, 3723
redirection operator	85

referenced shared memory object .....	85
references .....	3470
refresh .....	85
regcomp() .....	<b>1794</b> , 3760
regerror() .....	1794, 3760
regexexec() .....	1794, 3760
regfree() .....	1794, 3760
region .....	85
register fork handlers .....	1552
REGTYPE .....	412
regular expressions .....	85, 2481, 2598, 2668, 2723, 2764, 2789, 2831, 2877, 2998, 3008
.....	3020, 3064, 3188, 3207, 3381, 3438, 3528
basic .....	183
definitions .....	3529
extended .....	188
general requirements .....	3529
grammar .....	192, 3535
related to shell patterns .....	2372
regular file .....	85, 3490
REG_ .....	475
REG_ constants	
defined in <regex.h> .....	<b>322</b>
error return values of regcomp .....	1796
used in regcomp .....	1794
REG_BADBR .....	323, 1796
REG_BADPAT .....	322, 1796
REG_BADRPT .....	323, 1796
REG_EBRACE .....	323, 1796
REG_EBRACK .....	323, 1796
REG_ECOLLATE .....	322, 1796
REG_ECTYPE .....	322, 1796
REG_EESCAPE .....	322, 1796
REG_EPAREN .....	323, 1796
REG_ERANGE .....	323, 1796
REG_ESPACE .....	323, 1796
REG_ESUBREG .....	322, 1796
REG_EXTENDED .....	322, 1794
REG_ICASE .....	322, 1794
REG_NEWLINE .....	322, 1794
REG_NOMATCH .....	322, 1796
REG_NOSUB .....	322, 1794
REG_NOTBOL .....	322, 1795
REG_NOTEOL .....	322, 1795
rejected utilities .....	3743
relational database operator .....	2863
relative pathname .....	86, 111
release batch job request .....	2436
relocatable file .....	86
relocation .....	86
remainder() .....	<b>1801</b>
remainderf() .....	1801
remainderl() .....	1801

remove a directory .....	1818, 3194
remove a directory entry .....	2191
remove a file .....	3187
remove() .....	<b>1803</b>
remque() .....	1136, 1805
remquo() .....	<b>1806</b>
remquof() .....	1806
remquol() .....	1806
rename a file .....	1811
rename() .....	<b>1808</b> , 3755
renameat() .....	1808
renice .....	<b>3183</b> , 3758
replenishment period .....	3603
requested batch service .....	86, 2432
requirements .....	15
rerun batch job request .....	2437
Rerunable .....	2429
reserved words .....	2339, 3709
Resource_List .....	2429
result overflows .....	118
result underflows .....	118
return .....	<b>2397</b>
rewind() .....	<b>1813</b>
rewinddir() .....	<b>1814</b> , 3755
re_ .....	475
RE_DUP_MAX .....	275, 2089, 2324, 3699
rindex() .....	3681
rint() .....	<b>1815</b>
rintf() .....	1815
rintl() .....	1815
rlimit .....	<b>377</b>
RLIMIT_ .....	475
RLIMIT_AS .....	378, 1081
RLIMIT_CORE .....	377, 1080
RLIMIT_CPU .....	377, 1080
RLIMIT_DATA .....	378, 1080
RLIMIT_FSIZE .....	378, 1080
RLIMIT_NOFILE .....	378, 1080, 1082
RLIMIT_STACK .....	378, 1081
rlim_ .....	475
RLIM_ .....	477
RLIM_INFINITY .....	377, 1080-1081
RLIM_SAVED_CUR .....	377, 1081
RLIM_SAVED_MAX .....	377, 1081
rm .....	<b>3187</b> , 3704, 3759
rm del .....	<b>3192</b>
rmdir .....	<b>3194</b> , 3759
rmdir() .....	<b>1817</b> , 3564, 3755
RMSGD .....	367, 1142
RMSGN .....	368, 1142
RNORM .....	368, 1142
robust mutex .....	86, 514, 1657, 3629



root directory .....	86, 2317, 3490, 3506-3507
root file system.....	3490
root of a file system .....	3491
round robin .....	508
round() .....	<b>1820</b>
roundf() .....	1820
roundl() .....	1820
routing .....	524, 3650
RPI .....	<b>9</b>
RPP .....	<b>10</b>
RPROTDAT .....	368, 1142
RPROTDIS .....	368, 1142
RPROTNORM.....	368, 1142
RS .....	<b>10</b>
rsh	
rationale for omission.....	3745
RS_HIPRI .....	367, 1048, 1141, 1745
RTLD_ .....	475
RTLD_DEFAULT .....	744
RTLD_GLOBAL .....	233, 742
RTLD_LAZY .....	233, 741
RTLD_LOCAL .....	233, 742
RTLD_NEXT .....	744
RTLD_NOW.....	233, 742
RTSIG_MAX.....	272, 332, 2089, 3770
runnable process (or thread).....	86
running process (or thread) .....	87
runtime values	
increasable .....	274
invariant .....	270
rusage.....	<b>377</b>
RUSAGE_ .....	475
RUSAGE_CHILDREN.....	377, 1083
RUSAGE_SELF .....	377, 1083
ru_.....	475
R_ANCHOR .....	192
R_OK.....	440
s6_.....	475
sact.....	<b>3197</b>
samefile().....	3677
saved resource limits .....	87
saved set-group-ID.....	87, 2317
saved set-user-ID.....	87, 2317
SA_ .....	475
sa_.....	475-476
SA_ macros	
declared in <signal.h> .....	334
SA_NOCLDSTOP .....	334, 491, 1941, 1945-1946, 3485
SA_NOCLDWAIT .....	334, 553-554, 1083, 1943
SA_NODEFER .....	334, 1943
SA_ONSTACK.....	334, 783, 1942
SA_RESETHAND .....	334, 1942-1943, 3680

SA_RESTART .....	334, 1548, 1942, 1958, 3680
SA_SIGINFO .....	334, 1941-1942, 1945, 1970, 3570
scalb() .....	3681
scalbln() .....	<b>1821</b>
scalblnf() .....	1821
scalblnl() .....	1821
scalbn() .....	1821
scalbnf() .....	1821
scalbnl() .....	1821
scandir() .....	596, 1823
scanf() .....	945, 1824
scs .....	<b>3200</b>
SCCS commands	
admin .....	2444
delta .....	2639
get .....	2812
prs .....	3107
rmdel .....	3192
sact .....	3197
scs .....	3200
unget .....	3332
val .....	3361
what .....	3426
SCHAR_MAX .....	280
SCHAR_MIN .....	280-281
schedule alarm .....	594
scheduling .....	87
scheduling allocation domain .....	87, 3641
scheduling contention scope .....	87, 3641-3642
scheduling documentation .....	517, 3642
scheduling policy .....	88, 113, 3507
round robin .....	508
SCHED_ .....	475
sched .....	475
SCHED_FIFO .....	324, 504, 507, 516, 783, 894, 1067, 1394, 1565, 1567, 1636, 1677, 1699, 1848
.....	3756
sched_getparam() .....	<b>1826</b>
sched_getscheduler() .....	<b>1827</b>
sched_get_priority_max() .....	<b>1825</b>
sched_get_priority_min() .....	1825
SCHED_OTHER .....	324, 507, 510, 1067, 1567, 1636
SCHED_RR .....	324, 504, 507-508, 516, 783, 894, 1067, 1394, 1565, 1567, 1636, 1699, 1848
.....	3756
sched_rr_get_interval() .....	<b>1828</b>
sched_setparam() .....	<b>1829</b>
sched_setscheduler() .....	<b>1831</b>
SCHED_SPORADIC .....	324, 504, 507, 509, 783, 1699, 1848, 3756
sched_yield() .....	<b>1833</b>
SCM_ .....	476
SCM_RIGHTS .....	386
SCN .....	477
scope .....	3467

screen .....	88
scroll .....	88
SD .....	<b>10</b>
sdb	
rationale for omission .....	3745
sdiff	
rationale for omission .....	3745
search pattern .....	2610
seconds since the Epoch .....	113, 3507-3508
security considerations .....	555, 671, 964, 1219, 1900, 3478, 3482, 3486, 3499, 3501, 3539
security, monolithic privileges .....	3478
sed .....	<b>3205</b> , 3758-3759
addresses .....	3207
editing commands .....	3207
regular expressions .....	3207
seed48() .....	747, 1834
seekdir() .....	<b>1835</b>
SEEK_ .....	477
SEEK_CUR .....	238, 354, 442, 820, 953, 1284
SEEK_END .....	238, 354, 442, 820, 878, 953, 1284
SEEK_GET .....	1813
SEEK_SET .....	238, 354, 442, 503, 584, 591, 820, 953, 1284
SEGV_ .....	475
SEGV_ACCERR .....	336
SEGV_MAPERR .....	336
select batch jobs request .....	2437
select() .....	1545, 1837
sem .....	476
sem*() .....	3576
semaphore .....	88, 114, 3509, 3580, 3756
lock operation .....	114
unlock operation .....	114
semctl() .....	<b>1859</b> , 3576
semget() .....	<b>1862</b> , 3576
semid .....	502
semop() .....	<b>1864</b> , 3576
SEM_ .....	475
sem_ .....	475
SEM_ .....	476
sem_close() .....	<b>1838</b>
sem_destroy() .....	<b>1840</b>
SEM_FAILED .....	328, 1846-1847
sem_getvalue() .....	<b>1841</b>
sem_init() .....	<b>1843</b> , 3580
SEM_NSEMS_MAX .....	272, 1843, 2090, 3770
sem_open() .....	<b>1845</b> , 3580
sem_perm .....	502
sem_post() .....	<b>1848</b>
sem_timedwait() .....	<b>1850</b> , 3611
sem_trywait() .....	<b>1854</b> , 3565, 3582
SEM_UNDO .....	381, 1864
sem_unlink() .....	<b>1856</b>

SEM_VALUE_MAX .....	272, 1843, 1845, 2090, 3770
sem_wait() .....	1854, 1858, 3565, 3582
send() .....	<b>1869</b>
sendmsg() .....	<b>1872</b>
sendto() .....	<b>1876</b>
sequential lists .....	2360, 3730
servent .....	<b>301</b>
server shutdown request .....	2438
server status request .....	2438
service name .....	930
session .....	89, 556, 1219, 1901, 1913, 3485, 3489, 3539
session leader .....	89
session lifetime .....	89
session membership .....	2317
set .....	<b>2399</b> , 3715
set cancelability state .....	1720
set file creation mask .....	2180
set process group ID for job control .....	1900
set-group-ID .....	555, 667, 790, 825, 2317, 2600
set-user-ID .....	555, 790, 1015, 1219, 2317, 2566, 2600
set-user-ID scripts .....	3229
SETALL .....	381, 1859, 1862
setbuf() .....	<b>1880</b>
setegid() .....	<b>1882</b>
setenv() .....	<b>1883</b>
seteuid() .....	<b>1885</b>
setgid() .....	<b>1886</b> , 3491
setgrent() .....	756, 1888, 3499
sethostent() .....	758, 1889
setitimer() .....	1042, 1890
setjmp() .....	<b>1891</b> , 3757
setkey() .....	<b>1893</b>
setlocale() .....	<b>1894</b> , 3757
extensions to .....	1895
setlogmask() .....	692, 1898, 3760
setnetent() .....	760, 1899
setpgid() .....	<b>1900</b> , 3484, 3486, 3538-3539
setpgrp() .....	<b>1903</b>
setpriority() .....	1067, 1904, 3601
setprotoent() .....	762, 1905
setpwent() .....	764, 1906, 3499
setregid() .....	<b>1907</b>
setreuid() .....	<b>1909</b>
setrlimit() .....	1080, 1911, 3704
setservent() .....	767, 1912
setsid() .....	<b>1913</b> , 3538
setsockopt() .....	<b>1915</b>
setstate() .....	1134, 1917
setuid() .....	<b>1918</b> , 3491
setutxent() .....	769, 1921
SETVAL .....	381, 1859, 1862
setvbuf() .....	<b>1922</b>

sh.....	3215, 3759, 3766
command history list.....	3219
command line editing.....	3219
vi line editing command mode.....	3221
vi line editing insert mode.....	3220
vi-mode command line editing.....	3220
shall.....	6, 3470
rationale.....	3470
shar	
rationale for omission.....	3745
shared memory.....	3592
shared memory object.....	89
shell.....	89
SHELL.....	178
shell.....	556, 789, 1046, 1063, 1219, 1901, 2223, 3484-3486
SHELL.....	3527
shell.....	3538-3539, 3566, 3571-3572
commands.....	2355, 3725
errors.....	3725
execution environment.....	2371, 2450, 3182, 3319, 3709, 3734
grammar.....	2365, 3733
grammar rules.....	2365, 3734
grammar, lexical conventions.....	2365, 3734
introduction.....	2335
job control.....	1219, 3484, 3566, 3572
login.....	789, 1046
variables.....	2341, 3714
shell command language.....	2335
alias substitution.....	2338
appending redirected output.....	2351
arithmetic expansion.....	2348
command substitution.....	2347
compound commands.....	2361
consequences of shell errors.....	2353
double-quote.....	2336
duplicating an input file descriptor.....	2353
duplicating an output file descriptor.....	2353
escape character (backslash).....	2336
exit status and errors.....	2353
exit status for commands.....	2354
field splitting.....	2349
function definition command.....	2364
grammar.....	2365
here-document.....	2352
introduction.....	2335
lists.....	2359
open file descriptors for reading and writing.....	2353
parameter expansion.....	2344
parameters and variables.....	2339
pathname expansion.....	2350
pattern matching notation.....	2372
patterns matching a single character.....	2372

patterns matching multiple characters .....	2373
patterns used for filename expansion .....	2373
pipelines .....	2358
positional parameters .....	2339
quote removal .....	2350
quoting .....	2336
redirecting input .....	2351
redirecting output .....	2351
redirection .....	2350
reserved words .....	2339
shell commands .....	2355
shell execution environment .....	2371
shell grammar lexical conventions .....	2365
shell grammar rules .....	2365
shell variables .....	2341
signals and error handling .....	2371
simple commands .....	2355
single-quote .....	2336
special built-in utilities .....	2374
special parameter .....	2340
tilde expansion .....	2344
token recognition .....	2337
word expansions .....	2343
shell script .....	90
exec .....	789
shell, the .....	89
Shell_Path_List .....	2429
shift .....	<b>2406</b>
shl	
rationale for omission .....	3745
SHM .....	<b>10</b> , 476
shm .....	476
shm*() .....	3576
shmat() .....	<b>1931</b>
shmctl() .....	<b>1933</b> , 3576
shmdt() .....	<b>1935</b> , 3576
shmget() .....	<b>1937</b>
shmid .....	502
SHMLBA .....	383, 1931
shm_ .....	475
SHM_ .....	476
shm_open() .....	<b>1924</b> , 3591-3594
shm_perm .....	502
SHM_RDONLY .....	383, 1931
SHM_RND .....	383, 1931
shm_unlink() .....	<b>1929</b> , 3592-3594
should .....	6, 3470
rationale .....	3470
SHRT_MAX .....	281
SHRT_MIN .....	281
shutdown() .....	<b>1939</b>
SHUT_ .....	476

SHUT_RD .....	388
SHUT_RDWR .....	388
SHUT_WR .....	388
SIGABRT .....	332, 564, 3497, 3565
sigaction() .....	<b>1941</b> , 3568, 3570
sigaddset() .....	<b>1948</b>
SIGALRM .....	332, 594, 1042, 1989
sigaltstack() .....	<b>1949</b>
SIGBUS .....	332, 336, 506, 1333, 1336, 1726, 3497, 3566
SIGCANCEL .....	1595
SIGCHLD .....	332, 336, 553-554, 693, 1083, 1108, 1941, 1946, 1955, 2098, 2227, 3485, 3567
.....	3571
SIGCLD .....	1946, 3571
SIGCONT .....	332, 495, 554, 556, 1218-1219, 2689, 3485, 3568, 3571-3572
sigdelset() .....	<b>1951</b>
sigemptyset() .....	<b>1952</b>
SIGEMT .....	3566
SIGEV_ .....	475
sigev_ .....	475
SIGEV_NONE .....	331, 489, 504, 3568
SIGEV_SIGNAL .....	331, 489, 2142, 3568-3569
SIGEV_THREAD .....	331, 490, 1241, 3568
sigfillset() .....	<b>1954</b>
SIGFPE .....	332, 336, 1726, 1962, 3497, 3566, 3568
sighold() .....	<b>1955</b>
SIGHUP .....	332, 553-554, 556, 686, 2667, 2689, 3364, 3405, 3572
sigignore() .....	1955
SIGILL .....	332, 336, 1726, 1962, 3497, 3566
siginfo_t .....	<b>335</b>
SIGINT .....	332, 896, 2098, 2371, 2641, 2667, 2688, 3417, 3486, 3639, 3734
siginterrupt() .....	<b>1958</b>
SIGIOT .....	3565-3566
sigismember() .....	<b>1960</b>
SIGKILL .....	332, 1219, 1941, 1945-1946, 1955, 3565, 3568, 3572
siglongjmp() .....	<b>1961</b> , 3562, 3573, 3757
signal .....	90, 488, 3491, 3651, 3734
acceptance .....	3567
actions .....	3571
concepts .....	3565
delivery .....	488, 3567
error handling .....	2371
generation .....	488, 3567
names .....	3565
realtime delivery .....	489
realtime generation .....	489
stack .....	90
signal batch job request .....	2438
signal handler .....	1962
signal processes .....	2868
signal() .....	<b>1962</b> , 3565, 3567
signaling NaN .....	247
signbit() .....	<b>1965</b>

signgam .....	1233
signgam() .....	<b>1966</b>
sigpause() .....	1955, 1967
sigpending() .....	<b>1968</b>
SIGPIPE .....	332, 816, 857, 919, 924, 954, 957, 1746, 2303, 3497, 3564
SIGPOLL .....	332, 336, 686, 1140-1141
sigprocmask() .....	1726, 1969, 3567
SIGPROF .....	332, 1042
sigqueue() .....	<b>1970</b>
SIGQUEUE_MAX .....	272, 1970, 2090
SIGQUIT .....	332, 2098, 2371, 2667, 3734
sigrlse() .....	1955, 1972
SIGRTMAX .....	332, 489-490, 1944, 1970, 1977, 1981, 3569-3570
SIGRTMIN .....	332, 489-490, 1944, 1970, 1977, 1981, 3569-3570
SIGSEGV .....	332, 336, 505, 1081, 1379, 1560, 1726, 1962, 3497, 3566
sigset() .....	1955, 1972
sigsetjmp() .....	<b>1973</b> , 3757
sigset_t .....	3565
SIGSTKSZ .....	334, 1949
SIGSTOP .....	332, 489, 1941, 1946, 1955, 3572
sigsuspend() .....	<b>1975</b> , 3571, 3574
SIGSYS .....	332, 3566
SIGTERM .....	332, 2689, 3565
sigtimedwait() .....	<b>1977</b> , 3565, 3585, 3611
SIGTRAP .....	332, 336, 3566
SIGTSTP .....	332, 489, 2734, 3486, 3572
SIGTTIN .....	332, 489, 860, 866, 1765, 3486, 3539, 3572
SIGTTOU .....	332, 489, 815, 856, 919, 923, 954, 956, 2109, 2111, 2113, 2120, 2122-2123, 2125
.....	2303, 3485, 3539, 3572
SIGURG .....	332, 1141
SIGUSR1 .....	332, 3565
SIGUSR2 .....	332, 3565
SIGVTALRM .....	332, 1042
sigwait() .....	<b>1981</b> , 3565, 3646
sigwaitinfo() .....	1977, 1983, 3565, 3585
sigwait_multiple() .....	3567
SIGXCPU .....	332, 1080
SIGXFSZ .....	332, 1080, 2169
SIG_ .....	477
SIG_ATOMIC_MAX .....	351
SIG_ATOMIC_MIN .....	351
SIG_BLOCK .....	334, 1726
SIG_DFL .....	331, 491, 783, 1081, 1941, 1943, 1962-1963, 3567-3568, 3571
SIG_ERR .....	331, 1963
SIG_HOLD .....	331, 1955
SIG_IGN .....	331, 491, 553-554, 783, 790, 1083, 1941, 1962, 2371, 3485, 3567-3568, 3571, 3574
SIG_SETMASK .....	334, 1726
SIG_UNBLOCK .....	334, 1726
simple commands .....	2355, 3726
sin() .....	<b>1984</b>
sin6_ .....	475
sinf() .....	1984



single-quote.....	90, 2336, 3706
single-threaded process.....	90
single-threaded program .....	90
sinh().....	<b>1986</b>
sinhf() .....	1986
sinhl().....	1986
sinl().....	1984, 1988
sin_ .....	475
SIO .....	<b>10</b>
SIOCATMARK .....	1993
sival_ .....	475
size	
rationale for omission.....	3746
SIZE_MAX .....	351
size_t .....	401
SI_ .....	475
si_.....	475
SI_ASYNCIO.....	336, 492
SI_MSGQ.....	336, 492
SI_QUEUE.....	336, 492
SI_TIMER .....	336, 492
SI_USER.....	336, 492, 3570
slash.....	91
sleep .....	<b>3233</b> , 3757
sleep().....	<b>1989</b> , 3573-3574, 3754, 3756
SLR(1) grammars.....	3457
sl_.....	475
SND .....	475
SNDTIMEO .....	529
SNDZERO .....	368, 1144
snprintf() .....	906, 1992
SO .....	476
sockaddr_in.....	<b>305</b>
sockaddr_in6.....	<b>305</b>
socketmark().....	<b>1993</b>
socket .....	91, 523, 3650
address.....	91
address families.....	523
addressing .....	524
asynchronous errors .....	527
connection indication queue.....	527
I/O mode.....	525, 3651
Internet Protocols .....	532, 3652
IPv4.....	532, 3652
IPv6.....	532, 3652
local UNIX connection.....	3652
local UNIX connections.....	531
options .....	528
out-of-band data.....	527
out-of-band data state.....	3651
owner .....	526, 3651
pending error .....	526

protocols .....	524
queue limit .....	3651
queue limits.....	526
receive queue .....	526, 3651
signals .....	527
types .....	524, 3651
socket() .....	<b>1995</b>
socketpair() .....	<b>1997</b> , 3679
SOCK_.....	477
SOCK_DGRAM.....	386, 532, 1995, 1997
SOCK_RAW .....	386, 532
SOCK_SEQPACKET .....	386, 532, 1995, 1997
SOCK_STREAM.....	386, 532, 1995, 1997
soft limit.....	91
software development.....	3752, 3760
SOL_SOCKET .....	386
SOMAXCONN .....	387
sort .....	<b>3236</b> , 3758-3759
source code.....	91
SO_ACCEPTCONN.....	387, 529, 1915
SO_BROADCAST .....	387, 529
SO_DEBUG .....	387, 529
SO_DONTROUTE .....	387, 529
SO_ERROR.....	387, 529, 1915
SO_KEEPAIVE .....	387, 529
SO_LINGER.....	387, 529
SO_OOBINLINE .....	387, 529
SO_RCVBUF .....	387, 529
SO_RCVLOWAT .....	387, 529
SO_RCVTIMEO.....	387, 529, 1915
SO_REUSEADDR.....	387, 529
SO_SNDBUF .....	387, 529
SO_SNDLOWAT .....	387, 529
SO_SNDTIMEO.....	387, 1915
SO_TYPE .....	387, 529, 1915
space character.....	92
spawn.....	92
spawn example.....	3682
special built-in .....	92, 2589, 3018, 3031, 3120, 3229, 3288, 3307, 3731
special built-in utilities .....	2374, 3736
break.....	2376, 2381
characteristics.....	2374
colon.....	2379
dot.....	2383
eval .....	2385
exec.....	2387
exit .....	2389
export .....	2391
readonly.....	2394
return.....	2397
set.....	2399
shift.....	2406

times .....	2408
trap .....	2410
unset .....	2414
special characters.....	3541
special control character.....	3542
special parameter .....	92, 2340, 3709
special targets.....	2963
specific implementation .....	3483
SPEC_CHAR.....	193
spell	
rationale for omission.....	3746
spin lock .....	92, 3627-3628
split.....	<b>3244</b> , 3758
split files	
csplit .....	2606
split.....	3244
SPN.....	<b>10</b>
spoofing .....	2395
sporadic server .....	92
sporadic server policy	
execution capacity .....	509
replenishment period .....	509
scheduling .....	3603
sprintf() .....	906, 1999
spurious wakeup.....	1603
sqrt() .....	<b>2000</b>
sqrtf() .....	2000
sqrtl() .....	2000
rand() .....	1759, 2002
rand48() .....	747, 2003
random().....	1134, 2004
SS .....	<b>10</b>
sscanf() .....	945, 2005
SSIZE_MAX .....	281, 403, 1352, 1368, 1763, 1775, 2031, 2300, 3678, 3770
ssize_t.....	401
SS_ .....	475
ss_ .....	475-476
SS_DISABLE .....	334, 1949-1950
SS_ONSTACK.....	334, 1949
SS_REPL_MAX.....	272, 3606
stack size.....	1555
stack_t .....	<b>334</b>
standard error .....	92, 2350
standard I/O stream.....	495, 3575
standard input .....	92, 2350
standard output.....	93, 2350
standard utilities.....	93
START.....	2111
stat .....	3704
stat data structure.....	<b>391</b>
stat() .....	961, 2006, 3481, 3591, 3704, 3754
state-dependent character encoding .....	3512

status information .....	3679
statvfs() .....	967, 2007, 3704
stderr .....	355, 2008, 3575
STDERR_FILENO .....	446, 2008
stdin .....	355, 2008, 3575
STDIN_FILENO .....	446, 1429, 2008
stdio	
locking functions .....	872
with explicit client locking .....	1009
stdout .....	355, 2008, 3575
STDOUT_FILENO .....	446, 1429, 2008
STOP .....	2111
stpcpy() .....	2010, 2020
stpncpy() .....	2011, 2048
STR .....	477
strbuf .....	365
strcasecmp() .....	2012
strcasecmp_l() .....	2012
strcat() .....	2014
strchr() .....	2015
strcmp() .....	2016
strcoll() .....	2018
strcoll_l() .....	2018
strcpy() .....	2020
strcspn() .....	2023
strdup() .....	2024
stream .....	93
byte-oriented .....	498
interaction with file descriptors .....	497
stream orientation .....	498
wide-oriented .....	498
stream-full-policy attribute .....	539-540, 542, 1487
stream-min-size attribute .....	542, 1490
STREAMS .....	25, 93, 236, 365, 482, 686, 805, 828, 1048, 1139, 1156, 1402, 1425, 1545, 1745
.....	1764, 2302, 3575
access .....	501
end .....	93
head .....	93
head/tail .....	500
multiplexed .....	1147
multiplexor .....	94
overview .....	500
STREAM_MAX .....	272, 832, 890, 1429, 2090, 2153, 3770
strerror() .....	2026
strerror_l() .....	2026
strerror_r() .....	2026
strfdinsert .....	365
strfmon() .....	2029
strfmon_l() .....	2029
strftime() .....	2034
strftime_l() .....	2034
string .....	94

strings.....	3248, 3760
strioctl .....	365
strip .....	3251, 3760
strlen() .....	2043
strncasecmp() .....	2012, 2045
strncasecmp_l() .....	2012, 2045
strncat() .....	2046
strncmp() .....	2047
strncpy() .....	2048
strndup() .....	2024, 2050
strnlen() .....	2051
strpbrk() .....	2052
strpeek.....	365
strptime() .....	2053
strchr() .....	2058
strrecvfd.....	365
strsignal() .....	2059
strspn() .....	2060
strstr() .....	2061
strtod() .....	2062
strtof() .....	2062
strtoimax() .....	2066
strtok() .....	2068
strtok_r() .....	2068
strtol() .....	2071
strtold() .....	2062, 2074
strtoll() .....	2071, 2075
strtoul() .....	2076
strtoull() .....	2076
strtoimax() .....	2066, 2079
structures, additions to.....	3558
strxfrm() .....	2080
strxfrm_l() .....	2080
str_.....	475
str_list.....	365
str_mlist .....	366
stty .....	3253, 3527, 3760
combination modes.....	3258
control modes .....	3253
input modes .....	3254
local modes.....	3256
output modes.....	3255
special control character assignments.....	3257
ST_.....	476
st_.....	476
st_gid.....	2457
st_mode.....	2457
st_mtime .....	2457
ST_NOSUID.....	396, 783, 967
ST_RDONLY .....	396, 967
st_size.....	2457
st_uid.....	2457

su	
rationale for omission	3746
subprofiling	21, 3475
subprofiling option groups	3775
subshell	94, 3486
successfully completed	3498
successfully transferred	94
sum	3704
rationale for omission	3746
sun_	476
superuser	571, 671, 1238, 2191, 2773, 2919, 3083, 3478, 3491, 3501, 3715, 3743
supplementary group ID	94, 2317, 3491
supplementary groups	671, 1037, 3501
supported threads functions	3630
suseconds_t	401
suspended job	94
SVID	1973
SVR4	1335, 1382
sv_	475
SV_	477
swab()	2082
swapcontext()	3680
swprintf()	989, 2083
swscanf()	999, 2084
SWTCH	477
symbolic constant	95, 3472, 3492
symbolic link	95, 3493
symbolic name	3472
symbols	3556
POSIX.1	472
symlink()	2085
symlinkat()	2085
SYMLINK_MAX	274, 282, 899, 2086
SYMLOOP_MAX	272, 2090, 3562
SYMTYPE	412
sync()	2088
synchronized I/O	3585, 3755
completion	95
data integrity completion	95, 3498, 3585
file integrity completion	96, 3498, 3585
synchronized I/O operation	96
synchronized input and output	95
synchronous I/O operation	96
synchronously accept a signal	1978
synchronously-generated signal	96, 3497
sysconf()	2089, 3483, 3588, 3590, 3592, 3639, 3697, 3753, 3755
syslog()	692, 2097, 3760
system	96
boot	96
call	3497
clock	96
configuration values	2820

console .....	97, 3498
crash .....	97, 970
database .....	3498
databases .....	97
documentation .....	97
name .....	2182, 3323
process .....	97, 3498
reboot .....	97, 3498
trace event .....	97
trace event type definitions .....	542
system documentation .....	3471
system environment .....	3752, 3760
System III .....	671, 2182, 3490, 3677
system interfaces .....	551, 3679
System V .....	556, 594, 671, 791, 824, 901, 1063, 1219, 1310, 1818, 1913, 1946, 1973, 2115
.....	2182, 3481, 3486, 3566
system() .....	2098, 3757, 3760-3761
system-wide .....	98
S_ .....	475
s_ .....	475
S_ .....	477
S_ constants	
defined in <sys/stat.h> .....	392
S_ macros	
defined in <sys/stat.h> .....	392
S_BANDURG .....	367, 1141
S_ERROR .....	367, 1141
S_HANGUP .....	367, 1141
S_HIPRI .....	367, 1140
S_IFBLK .....	392, 1319
S_IFCHR .....	392, 1319
S_IFDIR .....	392, 1319
S_IFIFO .....	392, 1319
S_IFLNK .....	392
S_IFMT .....	392
S_IFREG .....	392, 1319
S_IFSOCK .....	392
S_INPUT .....	367, 1140
S_IRGRP .....	809, 958, 961, 1319
S_IROTH .....	809, 958, 961, 1319
S_IRUSR .....	809, 958, 961, 1319
S_IRWXG .....	1319
S_IRWXO .....	1319
S_IRWXU .....	1319
S_ISBLK .....	392
S_ISCHR .....	392
S_ISDIR .....	392
S_ISFIFO .....	392
S_ISGID .....	394, 664, 666-667, 1319, 2169, 2301
S_ISLNK .....	393
S_ISREG .....	392
S_ISSOCK .....	393

S_ISUID .....	394, 664, 666, 1319, 2169, 2301
S_ISVTX .....	664, 1319
S_IWGRP .....	809, 958, 961, 1319
S_IWOTH .....	809, 958, 961, 1319
S_IWUSR .....	809, 958, 961, 1319
S_IXGRP .....	1319
S_IXOTH .....	1319
S_IXUSR .....	1319
S_MSG .....	367, 1140
S_OUTPUT .....	367, 1140
S_RDBAND .....	367, 1140-1141
S_RDNORM .....	367, 1140
S_TYPEISMQ .....	393
S_TYPEISSEM .....	393
S_TYPEISSHM .....	393
S_TYPEISTMO .....	393
S_WRBAND .....	367, 1140
S_WRNORM .....	367, 1140
tab character .....	98
TABDLY .....	416
TABn .....	416
tabs .....	<b>3262</b> , 3758
TABSIZE .....	628, 1282
tag file creation .....	2610
tail .....	<b>3266</b> , 3758
talk .....	<b>3270</b> , 3759
tan() .....	<b>2103</b>
tanf() .....	2103
tanh() .....	<b>2106</b>
tanhf() .....	2106
tanh1() .....	2106
tanl() .....	2103, 2108
tar	
rationale for omission .....	3746
tar format .....	3075
target queue .....	2435
target rule .....	2958
tcdrain() .....	<b>2109</b>
tcflow() .....	<b>2111</b>
tcflush() .....	<b>2113</b>
tcgetattr() .....	<b>2115</b> , 3485
tcgetpgrp() .....	<b>2117</b> , 3486, 3538
tcgetsid() .....	<b>2119</b>
TCIFLUSH .....	418, 2113
TCIOFF .....	418, 2111
TCIOFLUSH .....	418, 2113
TCION .....	418, 2111
TCOFLUSH .....	2113
TCOOFF .....	418, 2111
TCOON .....	418, 2111
TCP_ .....	475
TCP_NODELAY .....	310



TCSADRAIN.....	418, 2122
TCSAFLUSH.....	418, 2122
TCSANOW.....	418, 2122
tcsendbreak().....	<b>2120</b>
tcsetattr().....	<b>2122</b> , 3485, 3537
tcsetpgrp().....	<b>2125</b> , 3485-3486
TCT.....	<b>10</b>
tdelete().....	<b>2127</b>
tee.....	<b>3274</b> , 3757
TEF.....	<b>11</b>
telldir().....	<b>2132</b>
tempnam().....	<b>2133</b>
TERM.....	<b>178</b> , 3527
terminal.....	98
access control.....	2115, 2123, 3539
controlling.....	200
device file.....	3537
device file, closing.....	3541
device name.....	2173
type.....	198, 3536
terminal characteristics	
stty.....	3253
tabs.....	3262
tput.....	3296
tty.....	3311
terminal device.....	98
terminate a process.....	555, 2868
terminology.....	3470
termios.....	199
canonical mode input processing.....	202
control modes.....	209
controlling terminal.....	200
input modes.....	206
local modes.....	210
non-canonical mode input processing.....	202
output modes.....	207
process group.....	200
special control characters.....	212
termios structure.....	2115, 3541
test.....	<b>3277</b> , 3757, 3759
TeX.....	3759
text column.....	98
text file.....	98, 3498
tfind().....	2127, 2135
tgamma().....	<b>2136</b>
tgammaf().....	2136
tgammal().....	2136
TGEXEC.....	412
TGREAD.....	412
TGWRITE.....	412
THOUSEP.....	267
thread.....	99, 3499

thread cancelability	
states.....	3646
type.....	3646
thread cancellation.....	3644, 3646
cleanup handlers .....	520
thread cancellation points.....	3646
thread concurrency level.....	3634
thread creation.....	1626
thread creation attributes .....	1555, 3623
thread ID.....	99, 513, 1630, 3499, 3639
thread interactions .....	3650
thread list.....	99
thread mutex.....	514, 3640
thread read-write lock .....	3648
thread scheduling.....	515, 3640
thread stack guard size.....	3635
thread termination .....	1631
thread-safe function .....	3499
thread-safety .....	99, 114, 513, 872, 3499, 3509, 3636
rationale.....	3509
thread-specific data .....	3625
thread-specific data key .....	99
creation .....	1645
deletion .....	1647
thread-specific data management.....	1639
threads .....	512, 3622
extensions .....	3631
implementation models .....	3624
regular file operations.....	522
tilde.....	99
tilde expansion.....	2344, 3716
time.....	3286, 3757, 3760
time() .....	2139, 3562
timeouts .....	100, 3615
timer .....	100
ID .....	2144
overflow .....	100
timers .....	3607
TIMER_.....	476
timer_.....	476
TIMER_ABSTIME .....	425, 511, 681, 2146, 3607-3609
timer_create() .....	2142
timer_delete() .....	2145
timer_getoverrun() .....	2146
timer_gettime() .....	2146
TIMER_MAX .....	272, 2090, 3770
timer_settime().....	2146, 3607-3609
timer_t.....	401
times.....	2408
times().....	2149, 3562, 3614, 3754
timespec.....	424, 3508
timestamp clock.....	3675

timeval .....	379, 398
timezone() .....	2152
time_t .....	401, 3508
tm .....	424
TMAGIC .....	412
TMAGLEN .....	412
TMPDIR .....	178, 3061
tmpfile() .....	2153
tmpnam() .....	2155
TMP_MAX .....	354, 2133, 2154-2155
tms .....	400
tms_ .....	476
tm_ .....	476
toascii() .....	2157
TOEXEC .....	412
token .....	100
token recognition .....	2337, 3708
tolower() .....	2158
tolower_l() .....	2158
TOREAD .....	412
TOSTOP .....	417, 815, 856, 919, 923, 954, 956, 2303, 3485
touch .....	3290, 3704, 3759
toupper() .....	2160
toupper_l() .....	2160
towctrans() .....	2162
towctrans_l() .....	2162
towlower() .....	2164
towlower_l() .....	2164
TOWRITE .....	412
towupper() .....	2166
towupper_l() .....	2166
TPI .....	11
TPP .....	11
TPS .....	11
tput .....	3296, 3760
tr. ....	3299, 3758-3759
trace analyzer .....	3664
trace analyzer process .....	100
trace controller process .....	100
trace event .....	100
POSIX_TRACE_ERROR .....	543
POSIX_TRACE_FILTER .....	543, 1521
POSIX_TRACE_OVERFLOW .....	543
POSIX_TRACE_RESUME .....	543
POSIX_TRACE_START .....	543, 1530
POSIX_TRACE_STOP .....	543, 1530
trace event type .....	100, 3673
filtering .....	3673
mapping .....	101
trace examples .....	3662
trace filter .....	101
trace functions .....	546

trace generation version.....	101
trace log .....	101
trace model.....	3657
trace operation control .....	3662
trace point.....	101
trace storage .....	3661
trace stream .....	101
attribute .....	3667
identifier .....	101
states.....	3660
trace system.....	101
trace-name attribute.....	542, 1484
traced process .....	102
TRACE_EVENT_NAME_MAX.....	273, 1509, 1511
TRACE_NAME_MAX.....	273
TRACE_SYS_MAX.....	273, 1506
TRACE_USER_EVENT_MAX.....	273, 1509, 1511
tracing .....	25, 115
TRACING .....	1482, 1484, 1486, 1489, 1501, 1503, 1505, 1509, 1511, 1514, 1516, 1519, 1521
.....	1524, 1530
tracing .....	3509, 3652
all processes .....	3661
detailed objectives.....	3653
status of a trace stream .....	102
track batch job request.....	2439
trap .....	<b>2410</b> , 3734
TRAP .....	475
TRAP_BRKPT .....	336
TRAP_TRACE .....	336
TRC.....	<b>11</b>
TRI .....	<b>11</b>
triggering.....	3675
TRL .....	<b>11</b>
troff.....	3759
trojan horse.....	2920, 3478
true .....	2334, 2357, 3306, 3705, 3757
trunc().....	<b>2168</b>
truncate().....	<b>2169</b>
truncation-status attribute .....	1509
truncf().....	2168, 2171
truncl().....	2168, 2171
TSA.....	<b>12</b>
tsearch().....	2127, 2172
TSGID.....	412
TSH.....	<b>12</b>
tsort.....	<b>3308</b>
TSP.....	<b>12</b>
TSS.....	<b>12</b>
TSUID .....	412
TSVTX.....	412
tty.....	<b>3311</b> , 3760
ttyname().....	<b>2173</b> , 3636

ttynam_r()	2173
TTY_NAME_MAX	273, 2090, 2173, 3770
TUEXEC	412
TUREAD	412
TUWRITE	412
TVERSION	412
TVERSLEN	412
tv_	476
twalk()	2127, 2175
TYM	12
type	2334, 2357, 3313
typed memory	3594
name space	102
object	102
pool	102
port	102
TZ	179, 3527
tzname	2176
TZNAME_MAX	273, 2090, 3770
tzset	2176
tzset()	2176, 3757
T_FMT	267
T_FMT_AMPM	267
t_scalar_t	365
t_uscalar_t	365, 1142
ualarm()	3682, 3754
UCHAR_MAX	280-281
ucontext_t	334
uc_	475
UID_MAX	3678
uid_t	401, 3499
UINT	477
UINTMAX_MAX	351
UINTN_MAX	350
UINTPTR_MAX	351
UINT_FASTN_MAX	350
UINT_LEASTN_MAX	350
UINT_MAX	281, 594, 1990
UIO_MAXIOV	476
ulimit	2334, 2357, 3315
ulimit()	2178
ULLONG_MAX	281, 2077
ULONG_MAX	281, 2077, 2275, 3697
UL_	476
UL_GETFSIZE	432, 2178
UL_SETFSIZE	432, 2178
umask	2334, 2357, 3317, 3705, 3760
umask()	2180, 3754
umount()	3488
unalias	2334, 2357, 3321, 3705, 3758
uname	3323, 3760
uname()	2182, 3753

unary primaries .....	3279
unbind .....	102
unbounded priority inversion .....	3643
uncompress .....	<b>3326</b>
undefined .....	6, 3471
rationale .....	3471
underlying function .....	498
unexpand .....	<b>3329</b> , 3758
unget .....	<b>3332</b>
ungetc() .....	<b>2184</b>
ungetwc() .....	<b>2186</b>
unicast .....	532
uniq .....	<b>3335</b> , 3758-3759
unlink .....	<b>3340</b>
unlink() .....	<b>2188</b> , 3564, 3591-3592, 3594, 3755
unlinkat() .....	2188
unlockpt() .....	<b>2194</b>
unpack .....	
rationale for omission .....	3746
unsafe functions .....	3572
unset .....	<b>2414</b>
unsetenv() .....	<b>2195</b>
unspecified .....	6, 3471
rationale .....	3471
until loop .....	2364, 3731
UP .....	<b>12</b>
upper multiplexing .....	94
upshifting .....	103
US-ASCII .....	1155
uselocale() .....	<b>2196</b>
user database .....	103, 3499
access .....	3500
user ID .....	103, 2846
logname .....	2905
newgrp .....	3012
real and effective .....	1909
setting real and effective .....	1909
who .....	3429
user name .....	103
user requirements .....	3749
user trace event .....	103
user trace event type definitions .....	545
User_List .....	2430
USER_PROCESS .....	455, 769-770
USHRT_MAX .....	281
usleep() .....	3682, 3754
ustar format .....	3075
UTC .....	2176
utility .....	104, 119, 3509
argument syntax .....	3543
conventions .....	3543
description defaults .....	3700

limits.....	3697
option parsing.....	2826
syntax guidelines.....	216, 3544
utimbuf.....	<b>454</b>
utime().....	<b>2198</b> , 3754
utimensat().....	983, 2201
utimes().....	983, 2201
UTIME_NOW.....	393, 983
UTIME_OMIT.....	393, 983
utim_.....	476
utmpx.....	<b>455</b>
uts_.....	476
ut_.....	476
UU.....	<b>12</b>
uucp.....	<b>3342</b> , 3759
uudecode.....	<b>3346</b> , 3758-3759
uuencode.....	<b>3349</b> , 3758-3759
uustat.....	<b>3354</b>
uux.....	<b>3357</b>
val.....	<b>3361</b>
variable.....	104, 3709
variable assignment.....	119, 3509
Variable_List.....	2430
va_arg().....	<b>2202</b>
va_copy().....	2202
va_end().....	2202
va_start().....	2202
VDISCARD.....	477
vdprintf().....	2203
VDSUSP.....	477
VEOF.....	414, 3542
VEOL.....	414, 3542
VERASE.....	414
Version 7.....	594, 1219, 2182, 3506, 3706
vertical-tab character.....	104
vfork().....	3682
vfprintf().....	<b>2203</b>
VFS.....	396
vfscanf().....	<b>2205</b>
vfwprintf().....	<b>2206</b>
vfwscanf().....	<b>2208</b>
vhangup().....	3487
vi.....	<b>3364</b> , 3758-3759
<ESC>.....	3404
append.....	3385
change.....	3386
change to end-of-line.....	3387
clear and redisplay.....	3372
command descriptions.....	3365
control-D.....	3401
control-H.....	3401
control-T.....	3403

control-U .....	3403
control-V .....	3403
current and line above .....	3379
delete .....	3387
delete character .....	3397
delete to end-of-line .....	3388
display information .....	3371
edit the alternate file .....	3373
enter ex mode .....	3393
execute .....	3384
execute an ex command .....	3383
exit .....	3399
find character .....	3388-3389
find regular expression .....	3381
Initialization .....	3365
input mode commands .....	3399
insert .....	3390
insert empty line .....	3392
join .....	3390
mark position .....	3391
move back .....	3379-3380, 3385-3386
move cursor .....	3371, 3374-3375, 3394-3395
move down .....	3372
move forward .....	3379-3380
move to bigword .....	3388, 3396
move to bottom of screen .....	3390
move to first character in line .....	3383
move to first non-<blank> .....	3379
move to line .....	3389
move to matching character .....	3376
move to middle of screen .....	3391
move to next section .....	3378
move to specific column .....	3380
move to top of screen .....	3389
move to word .....	3388, 3396
move up .....	3372
newline .....	3402
nul .....	3401, 3404
page backwards .....	3370
page forward .....	3371
put from buffer .....	3392-3393
redraw screen .....	3373
redraw window .....	3398
regular expression .....	3384
repeat .....	3381
repeat find .....	3383, 3391
repeat substitution .....	3377
replace character .....	3393-3394
replace text with command .....	3375
return to previous context .....	3377
return to previous section .....	3378
reverse case .....	3385



reverse find character .....	3380
scroll backward.....	3373
scroll backward by line.....	3373
scroll forward.....	3370
scroll forward by line.....	3370
search for tagstring .....	3374
shift left .....	3383
shift right .....	3384
substitute character .....	3394
substitute lines .....	3394
terminate command or input mode .....	3374
undo .....	3395
undo current line .....	3396
yank.....	3397
yank current line .....	3398
VINTR.....	414
virtual processor.....	3500
VISIT .....	2127, 2175
visual mode.....	2686
VKILL.....	414
VLNEXT .....	477
VMIN .....	3542
vprintf().....	2203, 2209
VQUIT.....	414
VREPRINT .....	477
vscanf().....	2205, 2210
vsnprintf().....	2203, 2211
vsprintf() .....	2203, 2211
vsscanf() .....	2205, 2212
VSTART .....	414
VSTATUS.....	477
VSTOP.....	414
VSUSP .....	414
vswprintf().....	2206, 2213
vswscanf().....	2208, 2214
VTDLY .....	416
VTIME.....	3542
VTn.....	416
VWERASE.....	477
vwprintf().....	2206, 2215
vwscanf().....	2208, 2216
wait.....	2334, 2357, 3419, 3705, 3757
for process termination .....	2223
for thread termination .....	1642
wait() .....	2217, 3562, 3566, 3571-3572, 3574, 3754
waitid().....	2226, 3572, 3679, 3754
waiting on a condition.....	1611
waitpid() .....	2217, 2229, 3485, 3489, 3572, 3677, 3754
wall .....	
rationale for omission.....	3746
WARNING .....	885

warning	
OB .....	9
OF .....	9
wc .....	3423, 3758
WCHAR_MAX .....	351, 457
WCHAR_MIN .....	351, 457
WCONTINUED .....	408, 2217, 2226
wcpcpy() .....	2230, 2241
wcpncpy() .....	2231, 2250
wcrtomb() .....	2232
wscasecmp() .....	2234
wscasecmp_l() .....	2234
wscat() .....	2236
wcschr() .....	2237
wscmp() .....	2238
wscoll() .....	2239
wscoll_l() .....	2239
wscpy() .....	2241
wscspn() .....	2242
wcsdup() .....	2243
wcsftime() .....	2244
wcslen() .....	2246
wcsncasecmp() .....	2234, 2247
wcsncasecmp_l() .....	2234, 2247
wcsncat() .....	2248
wcsncmp() .....	2249
wcsncpy() .....	2250
wcsnlen() .....	2246, 2252
wcsnrtombs() .....	2253, 2256
wcspbrk() .....	2254
wcsrchr() .....	2255
wcsrtombs() .....	2256
wcsspn() .....	2258
wcsstr() .....	2259
wcstod() .....	2260
wcstof() .....	2260
wcstoimax() .....	2264
wcstok() .....	2265
wcstol() .....	2267
wcstold() .....	2260, 2270
wcstoll() .....	2267, 2271
wcstombs() .....	2272
wcstoul() .....	2274
wcstoull() .....	2274
wcstoumax() .....	2264, 2277
wcswcs() .....	3682
wcswidth() .....	2278
wcsxfrm() .....	2279
wcsxfrm_l() .....	2279
wctob() .....	2281
wctomb() .....	2282
wctrans() .....	2284

wctrans_l()	2284
wctype()	<b>2286</b>
wctype_l()	2286
wcwidth()	<b>2288</b>
WEOF	457, 462, 548, 1187, 1189, 1193, 1195, 1197, 1199, 1201, 1203, 1205, 1207, 1209 1211, 2164, 2166, 2186
WERASE	3540
WEXITED	408, 2226
WEXITSTATUS	358, 408, 2218
we_	476
what	<b>3426</b>
while loop	2363, 3731
white space	104
who	<b>3429</b> , 3759-3760
wide characters	129
wide-character code	3511
wide-character code (C language)	104
wide-character input/output functions	105
wide-character string	105
wide-oriented stream	498
WIFCONTINUED	408, 2218
WIFEXITED	358, 408, 2218
WIFSIGNALED	358, 408, 2218
WIFSTOPPED	358, 408, 2218, 2224
WINT_MAX	351
WINT_MIN	351
wmemchr()	<b>2289</b>
wmemcmp()	<b>2290</b>
wmemcpy()	<b>2291</b>
wmemmove()	<b>2292</b>
wmemset()	<b>2293</b>
WNOHANG	358, 408, 1946, 2217, 2226
WNOWAIT	408, 2226, 3679
word	105
word counting	3423
word expansions	2343, 3715
wordexp()	<b>2294</b> , 3760
wordfree()	2294, 3760
WORD_BIT	280-281
working directory	105
worldwide portability interface	105
wprintf()	989, 2299
WRDE_	476
WRDE_APPEND	464, 2295
WRDE_BADCHAR	464, 2296
WRDE_BADVAL	464, 2296
WRDE_CMDSUB	464, 2296
WRDE_DOOFFS	464, 2295
WRDE_NOCMD	464, 2295
WRDE_NOSPACE	464, 2296
WRDE_REUSE	464, 2295
WRDE_SHOWERR	464, 2295

WRDE_SYNTAX .....	464, 2296
WRDE_UNDEF .....	464, 2295
write .....	105, 3433, 3758-3759
write lock .....	3633
write to a file .....	2304
write() .....	<b>2300</b> , 3484-3485, 3539, 3562, 3571-3572, 3574, 3584-3586, 3590-3591
.....	3678, 3755
writew() .....	<b>2309</b>
writing data .....	3541
wscanf() .....	999, 2311
WSTOPPED .....	408, 2226
WSTOPSIG .....	358, 408, 2218
WTERMSIG .....	358, 408, 2218
WUNTRACED .....	358, 408, 2218, 2223, 3485
W_OK .....	440
xargs .....	<b>3436</b> , 3757
XOPEN_UNIX .....	19, 29
XOPEN_UUCP .....	19, 29
XSI .....	<b>12</b> , 105, 3500
conformance .....	15, 19, 106
XSI interprocess communication .....	<b>501</b>
XSI IPC .....	3576
XSI options groups .....	22
XSI STREAMS .....	25
XSI system interfaces	
conformance .....	19
XSI_C_LANG_SUPPORT .....	3779
XSI_DBM .....	3780
XSI_DEVICE_IO .....	3780
XSI_DEVICE_SPECIFIC .....	3780
XSI_FILE_SYSTEM .....	3780
XSI_IPC .....	3780
XSI_JUMP .....	3780
XSI_MATH .....	3780
XSI_MULTI_PROCESS .....	3780
XSI_SIGNALS .....	3780
XSI_SINGLE_PROCESS .....	3780
XSI_SYSTEM_DATABASE .....	3780
XSI_SYSTEM_LOGGING .....	3780
XSI_THREADS_EXT .....	3780
XSI_TIMERS .....	3780
XSI_USER_GROUPS .....	3780
XSI_WIDE_CHAR .....	3780
XSR .....	<b>13</b>
X_OK .....	440, 572
y0() .....	<b>2312</b>
y1() .....	2312
yacc .....	<b>3443</b> , 3760, 3762
algorithms .....	3454
code file .....	3445
completing the program .....	3454
conflicts .....	3452

debugging the parser .....	3454
declarations section .....	3446
description file .....	3445
error handling .....	3452
grammar rules .....	3448
header file .....	3445
input grammar .....	3450
input language .....	3445
interface to the lexical analyzer .....	3453
lexical structure of the grammar .....	3446
library .....	3454
limits .....	3455
programs section .....	3449
YESEXPR .....	267
yn() .....	2312
zcat .....	<b>3460</b>
zombie process .....	106, 553

